Ain Shams University
Faculty of Computer & Information Sciences
Scientific Computing Department

Arabic books search engine using NLP techniques

July 2025

Ain Shams University
Faculty of Computer & Information Sciences
Scientific Computing Department

Arabic books search engine

This documentation submitted as required for the degree of bachelors in Computer and Information Sciences.

By

Mohamed Sayed Zakaria Seddik	[Scientific Computing]
Mohamed Ahmed Elsayed Elsayed	[Scientific Computing]
Yousef Nasser Mohamed Ali	[Scientific Computing]
Mahmoud Mohamed Mostafa Ibrahim	[Scientific Computing]
Abd Elrahman Mohamed Ahmed Awwad	[Scientific Computing]
keroles ashraf fouad gheprial	[Scientific Computing]

Under Supervision of

Prof. Dr. Howida Abdelfattah Saber Shedeed
Head of the Scientific Computing Department, Faculty of Computer and Information Sciences,
Faculty of Computer and Information Sciences,
Ain Shams University.

TA. Nouran Elsayed
Scientific Computing Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

July 2025

Acknowledgements

All praise and thanks to ALLAH, who provided us the ability to complete this work. We hope to accept this work from me.

We are grateful of *our parents* and *our family* who are always providing help and support throughout the whole years of study. We hope we can give that back to them.

We also offer my sincerest gratitude to our supervisors, *Prof. Dr. Howida Shedeed*,
and T.A. Nouran Elsayed. who have supported us throughout my thesis with their patience, knowledge and experience.

Finally, We would thank our friends and all people who gave us support and encouragement.

Abstract

In today's digital age, the exponential growth of Arabic textual content has created a pressing need for intelligent and efficient search systems tailored to the unique characteristics of the Arabic language. This project presents an Arabic Book Search Engine that leverages Natural Language Processing (NLP) techniques to enhance information retrieval accuracy and relevance for Arabic texts.

Unlike traditional keyword-based search engines, our system applies advanced preprocessing, including normalization, tokenization, stemming, and vectorization methods to understand the semantic context of user queries. By using TF-IDF and other vector-based models, the engine retrieves and ranks Arabic book descriptions based on linguistic relevance.

With an average similarity score of **85%** between user queries and retrieved results, the system demonstrates high accuracy in matching user intent. The application is developed as a user-friendly mobile platform, providing users with fast and relevant access to books. This project aims to bridge the gap between Arabic language complexities and modern search technologies, ultimately improving access to Arabic literary resources.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	**Stands for**
NLP	Natural Language Processing
NLU	Natural Language Understanding
NLG	Natural Language Generation
ML	Machine Learning
GUI	Graphical User Interface
BOW	Bag of Words

Chapter 1: Introduction

1.1 Problem Definition

As the volume of Arabic digital content continues to grow, accessing relevant information efficiently has become increasingly challenging. Traditional search engines often fail to accurately retrieve Arabic content due to the complex morphology and rich semantics of the language. Furthermore, many available search systems rely on basic keyword matching, which overlooks the deeper linguistic structures and meanings in user queries and text data.

To address these issues, there is a critical need for a search engine tailored specifically to Arabic, one that understands the nuances of the language and applies advanced Natural Language Processing (NLP) techniques to deliver semantically relevant results. This project aims to bridge this gap by developing an Arabic book search engine that processes, understands, and retrieves Arabic book data effectively.

Problem Statement:

- Most search engines are optimized for English and perform poorly on morphologically rich languages like Arabic.
- Arabic has unique linguistic challenges such as diacritics, root-based morphology, and varied word forms that are not handled well by standard keyword-based searches.
- There is a lack of effective tools that allow Arabic-speaking users to search for books based on semantic understanding rather than exact word matching.
- Current Arabic book repositories and apps often provide limited or irrelevant search results due to inadequate text processing techniques.
- Users need a system that offers meaningful, accurate search results from large Arabic text datasets using context-aware NLP methods.

1.2 Motivation

The Arabic language, spoken by over 400 million people worldwide, holds a vast and rich body of literature, yet remains underrepresented in modern digital search technologies. As university students and native Arabic speakers, we personally experienced the difficulties of searching for Arabic books using conventional platforms. These platforms often return irrelevant results, suffer from poor language handling, and do not reflect the true intent behind user queries—especially in academic or literary contexts. This motivated us to build a solution that goes beyond simple keyword matching and embraces the power of Natural Language Processing (NLP) to improve how Arabic text is understood and retrieved. Our goal is to empower users—students, researchers, and casual readers alike—to find Arabic books quickly and accurately, even if their search terms differ from the exact words in the dataset. Additionally, this project allowed us to apply what we've learned in NLP, machine learning, and mobile development to solve a real-world problem faced by Arabic-speaking communities. It gave us the opportunity to innovate, contribute to our culture's digital transformation, and bridge the gap between Arabic language challenges and intelligent search solutions.

1.3 Objectives

The primary objective of this project is to design and implement an intelligent Arabic book search engine that leverages Natural Language Processing (NLP) techniques to enhance keyword-based search functionality. The goal is to provide users with accurate and relevant results by addressing the unique linguistic challenges of the Arabic language.

Specific Objectives:

- To build a searchable database of Arabic books with structured metadata and descriptions.
- To implement an efficient keyword-based search system enhanced by NLP preprocessing techniques (e.g., normalization, tokenization, stemming).
- To improve the relevance of search results by handling the morphological richness and variability of Arabic words.
- To design a user-friendly mobile application interface that allows users to easily search and browse Arabic books.
- To evaluate the effectiveness of the search engine in retrieving relevant results and improving user satisfaction.
- To promote the accessibility of Arabic literary resources through intelligent search technology.

1.4 Documentation outlines

Chapter 1: Introduction.
Chapter 2: Background.
Chapter 3: Dataset.
Chapter 4: System Architecture.
Chapter 5: system implementation & experimental results.
Chapter 6: GUI (Graphical User Interface).
Chapter 7: Conclustion & Future work.

Chapter 2: Background

2.1 Introduction to NLP

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. By "natural language" we mean a language that is used for everyday communication by humans, such as Arabic, English, Spanish....etc.

NLP is an Intersection of several fields Computer Science Artificial Intelligence Linguistics NLP is a difficult task because it involves a lot of unstructured data. Understanding context is an issue in NLP– that requires semantic analysis and machine learning to get a handle on it.

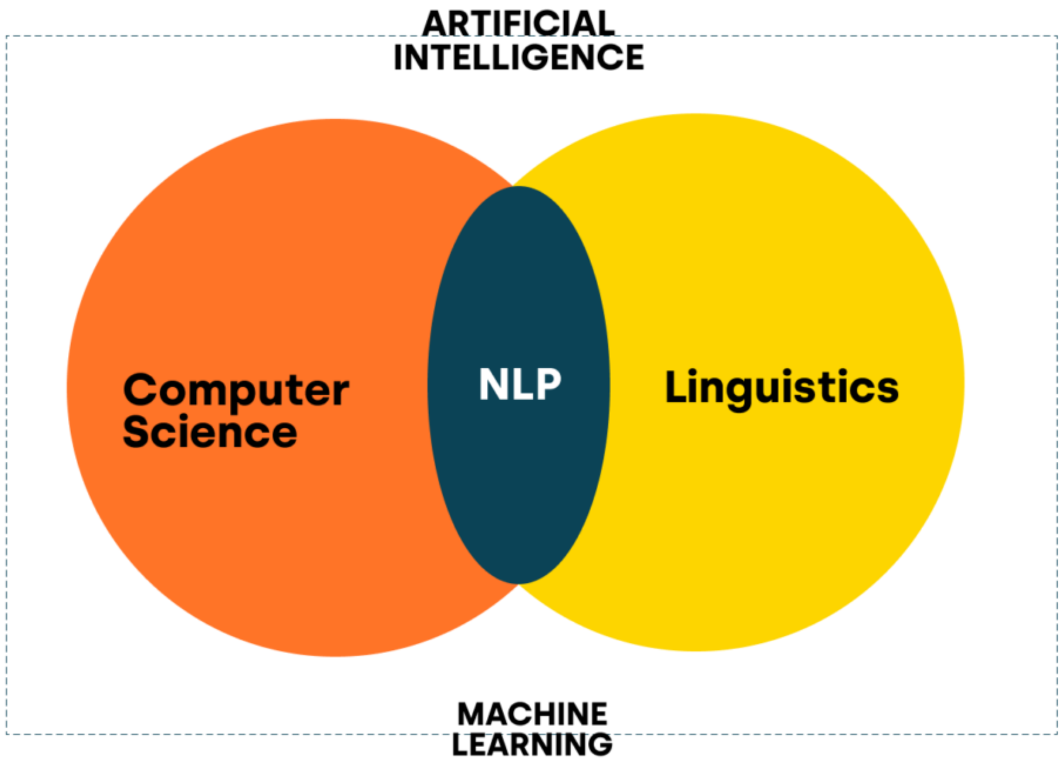


Figure 1: What is NLP

2.1.1 NLP Components

Two main components:

- Natural Language Understanding
- (NLU) Natural Language Generation (NLG)

Description:

NLG (Natural Language Generation) and NLU (Natural Language Understanding) are two core components of Natural Language Processing (NLP)—the field of AI that focuses on enabling machines to process and understand human language.

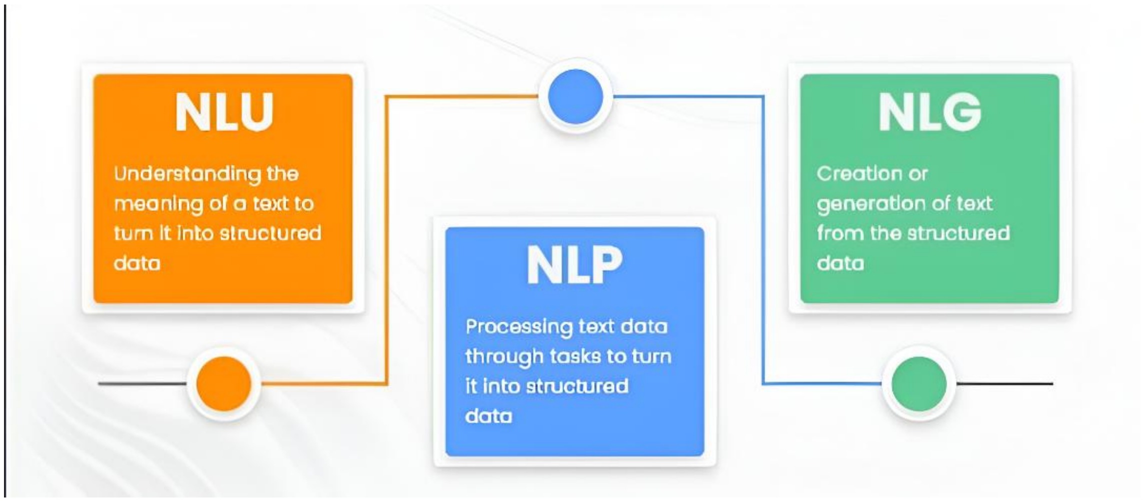

NLU
Understanding the meaning of a text to turn it into structured data
NLP
Processing text data through tasks to turn it into structured data
NLG
Creation or generation of text from the structured data

Figure 2: NLP Components

1. Natural Language Understanding (NLU)
   Goal: Understand and interpret human language.
   Functions of NLU:
   - Intent recognition: Understanding what the user wants (e.g., booking a flight).
   - Entity recognition: Identifying key details (e.g., location, date, time).
   - Sentiment analysis: Detecting emotions or opinions.
   - Disambiguation: Understanding context when words have multiple meanings.

Applications of NLU:

- ML on Text (Classification, Regression, Clustering)
- Document Recommendation
- Language Identification
- Natural Language Search
- Sentiment Analysis
- Text Summarization
- Extracting Word/Document Meaning (vectors)
- Relationship Extraction
- Topic Modeling

2. Natural Language Generation (NLG)
   Goal: Produce natural-sounding language from structured data or machine-readable formats.
   Functions of NLG:
   - Text planning: Deciding what information to convey.
   - Sentence generation: Structuring coherent and grammatically correct sentences.
   - Language style/tone adjustment: Adapting tone for audience or purpose.

Applications of NLG:

- Image Captioning.
- (Better) Text Summarization.
- Machine Translation.
- Question Answering/Chatbots

2.1.2 Challenges and limitations of NLP

1. Contextual words and phrases
   Some words in Arabic change meaning based on context, and literal translations may fail to capture the correct meaning.

2. Synonyms
   Arabic is rich in synonyms, which can confuse NLP models if not trained to recognize that different words can have the same meaning.

3. Irony and sarcasm
   Sarcasm is difficult for machines to detect, especially when the literal meaning is the opposite of the intended meaning.

4. Ambiguity
   Many Arabic words are ambiguous without diacritics (tashkeel), which are often omitted in written Arabic.

5. Errors in text or speech
   Spelling mistakes, mispronunciations, or auto-correct errors can lead to misinterpretation.

6. Colloquialisms and slang
   Arabic dialects vary widely by region, and slang terms may be unknown to standard NLP models trained on Modern Standard Arabic (MSA).

7. Low-resource languages
   While Modern Standard Arabic is somewhat supported, many Arabic dialects (e.g., Yemeni, Sudanese, Hassaniya) are low-resource with little training data.

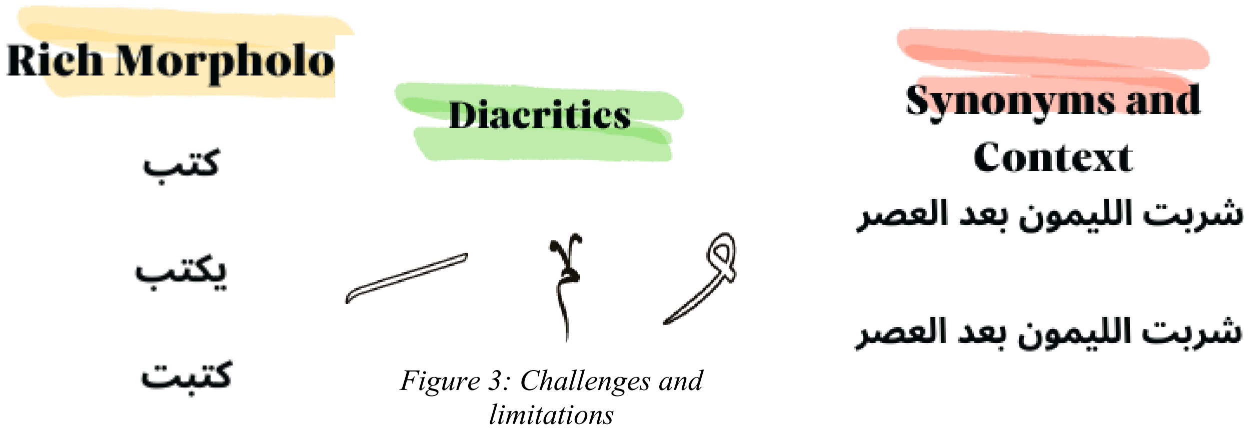


Figure 3: Challenges and limitations

2.2 Preprocessing

Proposed NLP-Techniques For Text Preprocessing

- sentence segmentation.
- Tokenization.
- Normalization.
- Stemming or Lemmatization.
- Stop-word.
- part of speech.

2.2.1 Sentence Segmentation

Description:
It is the first step for building the NLP pipeline. It breaks the paragraph into separate sentences. It involves identifying and splitting the text on sentence boundaries. Complex rules have been developed to ensure that sentence splitting is as accurate as possible to account for real-world usage—e.g., abbreviations ending with periods may not always be at the end of sentences, and misplaced decimal places should not needlessly split sentences.
Importance:

- Splits text into individual sentences.
- Enables sentence-level analysis (e.g., sentiment, question answering).
- Improves model input structure for downstream tasks like translation or summarization.

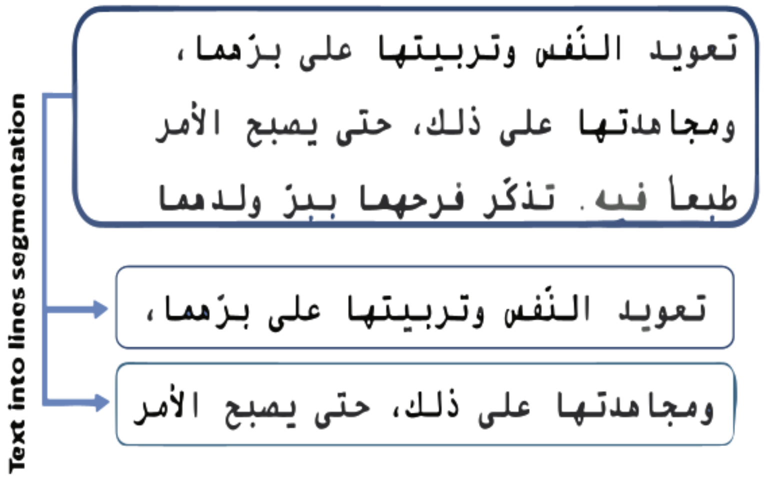

تعويد النّفس وتربيتها على بزّهما،
ومجاهدتها على ذلك، حتى يصبح الأمر
طبعاً فيه. تذكّر فرحهما ببزّ ولدهما
تعويد النّفس وتربيتها على بزّهما،
ومجاهدتها على ذلك، حتى يصبح الأمر
Text into lines segmentation

Figure 4: Sentence Segmentation

2.2.2 Word Tokenization

Description:
Word tokenization is the process of breaking up the given text into units called *tokens*. These tokens may be words, numbers, or punctuation marks. Tokenization does this task by locating word boundaries—i.e., the ending point of one word and the beginning of the next. This process is also known as word segmentation, especially in languages like Arabic where words may be tightly connected or inflected.

Importance:

- Converts text into individual, manageable units (tokens).
- Essential for downstream tasks such as parsing, text classification, and embedding generation.
- Enables frequency analysis, vectorization, and input preparation for neural models.

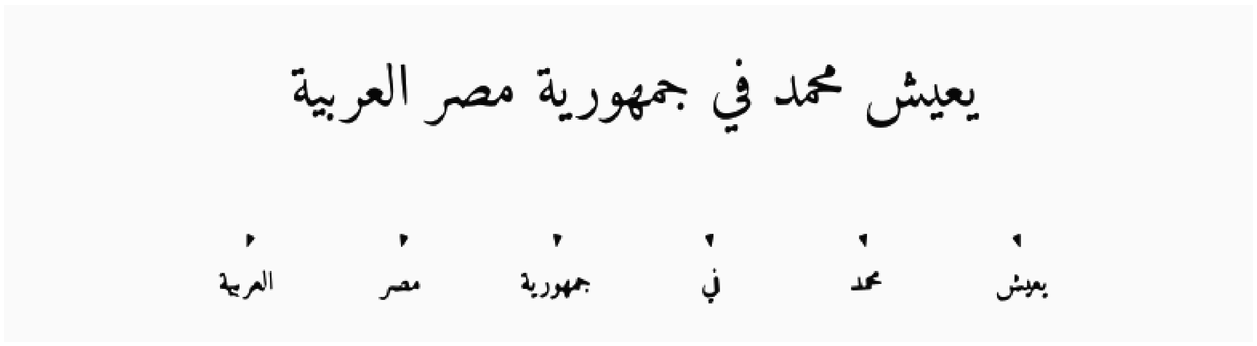


Figure 5: Word Tokenization

2.2.3 Normalization:

Description:
Normalization converts text into a standardized form to reduce variation in how words are written. This is especially important in Arabic due to its rich morphology and script variation. Common normalization steps include:

- Removing diacritics (e.g., ً, َ, ُ) to avoid confusion caused by optional pronunciation markers.
- Unifying different forms of "Alif" (e.g., ا → أ ,إ) to treat all forms of "Alif" as the same letter.
- Handling "Taa Marbuta" (e.g., ه → ة or vice versa depending on context) to ensure consistent representation.

Importance:

- Reduces inconsistencies in the text.
- Improves model generalization by treating variants of the same word as identical.
- Prepares text for more accurate tokenization, matching, and embedding.

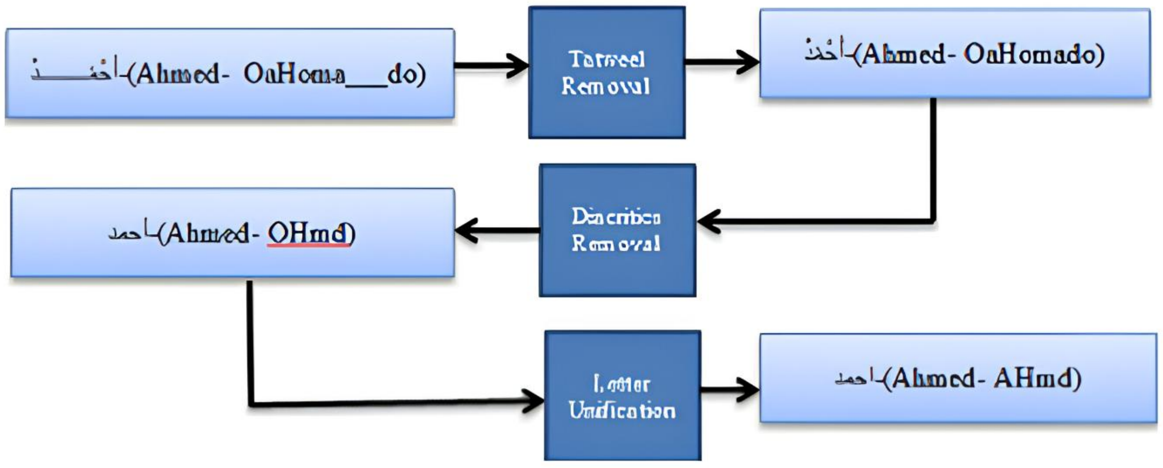


Figure 6: Normalization

2.2.4 Stemming:

Description:

Stemming in Arabic reduces words to their root form by removing common prefixes, suffixes, and sometimes infixes, focusing on the base structure of the word. For example, the words "كاتب" ,"مكتوب", and "كتابة" may all be reduced to the root "كتب". While this process simplifies text processing and reduces dimensionality, it may not always produce linguistically accurate or complete forms.

Importance:

- Helps group related words under a common root, improving text matching and retrieval.
- Reduces the vocabulary size, which is crucial for training efficient models.
- Particularly useful in search engines and topic classification.

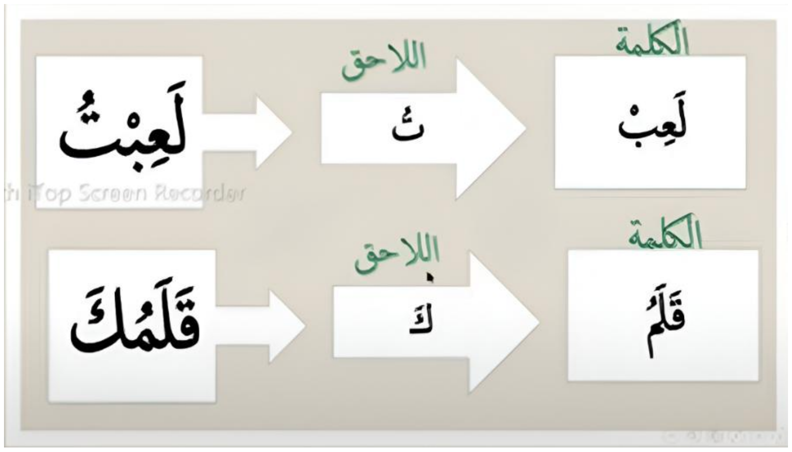


Figure 7: Stemming

2.2.5 Lemmatization:

Description:

is similar to stemming in function—it reduces words to their base or root form. However, unlike stemming, lemmatization produces linguistically correct base forms (called *lemmas*) that appear in the dictionary. Lemmas are not truncated or ambiguous, making lemmatization more accurate for tasks that require true word meaning and grammar.

For example, in Arabic:

- "كتبت" ,"يكتب", and "اكتب" are all lemmatized to **"كتب"** as a proper dictionary form.

Importance:

- Generates meaningful and grammatically correct base forms.
- Improves performance in tasks like machine translation, question answering, and language modeling.
- Essential for preserving semantic accuracy when interpreting Arabic text.

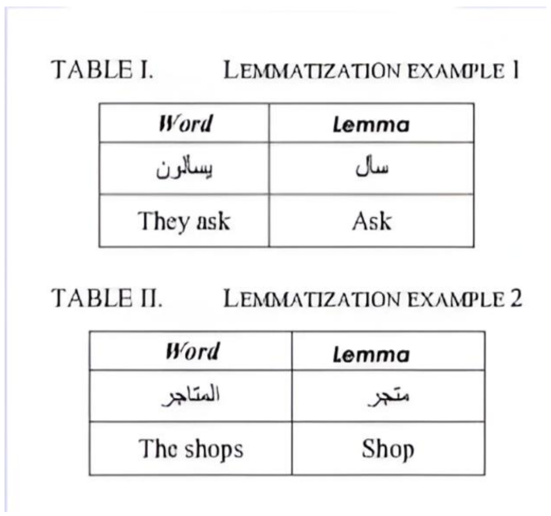


TABLE I. LEMMATIZATION EXAMPLE 1

Word	**Lemma**
يسالون	سال
They ask	Ask

TABLE II. LEMMATIZATION EXAMPLE 2

Word	**Lemma**
المتاجر	متجر
The shops	Shop

Figure 8: Lemmatization

2.2.6 Stopword:

Description:

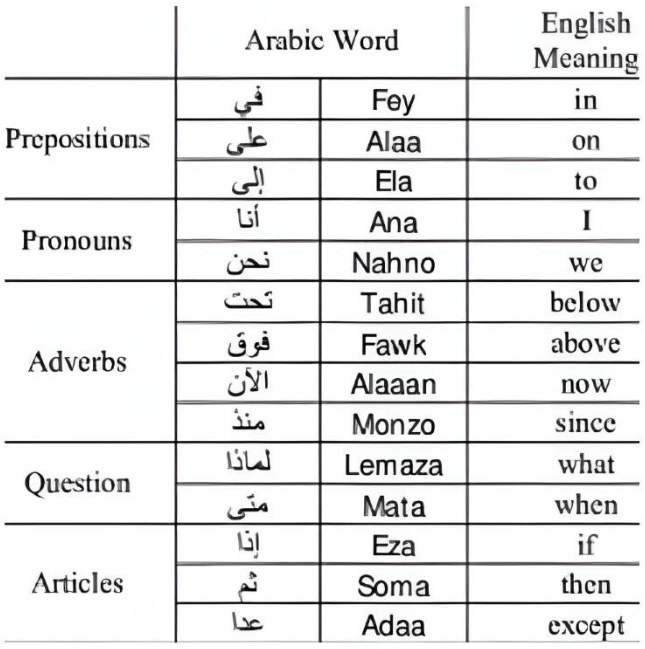
Stop words are commonly used words in a language that are often filtered out during NLP tasks because they carry little meaningful content on their own. These words typically do not contribute significantly to the core meaning of a sentence. In Arabic, stop words include pronouns, conjunctions, prepositions, articles, and certain particles.

Examples of Arabic stop words:

- "في", "من", "على", "هذا", "أن", "و", "هو"

Importance:

- Reduces noise in the text, allowing models to focus on more meaningful words.
- Improves the performance of tasks like text classification, information retrieval, and keyword extraction.
- Decreases the dimensionality of feature space in vector-based models.



	Arabic Word		English Meaning
Prepositions	في	Fey	in
	على	Alaa	on
	إلى	Ela	to
Pronouns	أنا	Ana	I
	نحن	Nahno	we
Adverbs	تحت	Tahit	below
	فوق	Fawk	above
	الآن	Alaaan	now
	منذ	Monzo	since
Question	لماذا	Lemaza	what
	متى	Mata	when
Articles	إذا	Eza	if
	ثم	Soma	then
	عدا	Adaa	except

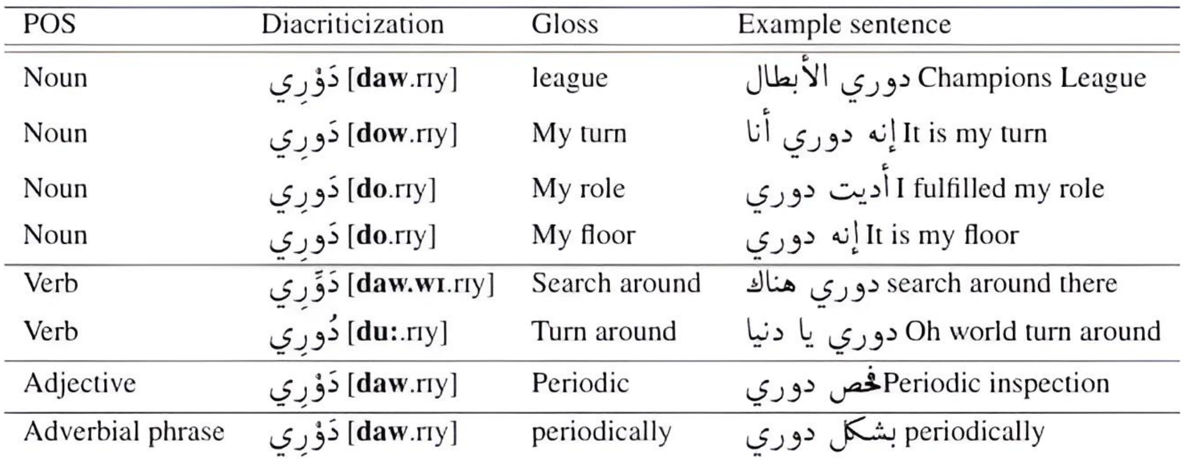
Figure 9: Stopword

2.2.7 Part Of Speech

Description:
POS tagging is the process of labeling each word in a sentence with its corresponding part of speech, such as noun, verb, adjective, etc. In Arabic, this task is particularly complex due to the language's rich morphology, extensive use of prefixes and suffixes, and flexible word order.

Importance:

- Enables deeper syntactic and semantic analysis.
- Supports key NLP tasks like information extraction, machine translation, and question answering.
- Improves accuracy in information retrieval and search engine relevance.



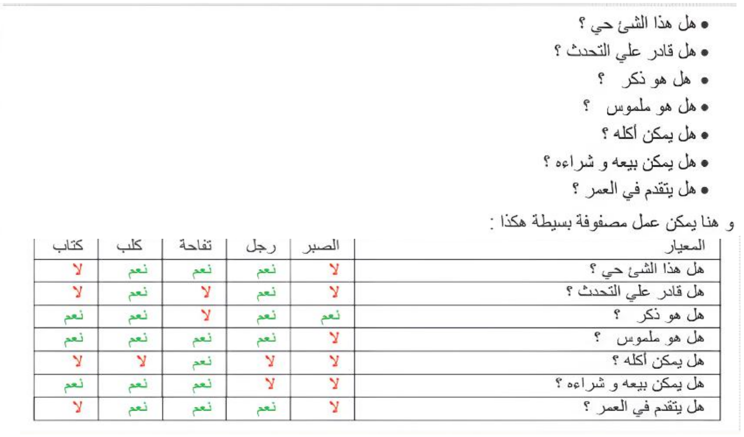
POS	Diacriticization	Gloss	Example sentence
Noun	دَوْرِي [**daw**.rɪy]	league	دوري الأبطال Champions League
Noun	دَورِي [**dow**.rɪy]	My turn	إنه دوري أنا It is my turn
Noun	دَورِي [**do**.rɪy]	My role	أديت دوري I fulfilled my role
Noun	دَورِي [**do**.rɪy]	My floor	إنه دوري It is my floor
Verb	دَوِّرِي [**daw.wɪ**.rɪy]	Search around	دوري هناك search around there
Verb	دُورِي [**du:**.rɪy]	Turn around	دوري يا دنيا Oh world turn around
Adjective	دَوْرِي [**daw**.rɪy]	Periodic	فحص دوري Periodic inspection
Adverbial phrase	دَوْرِي [**daw**.rɪy]	periodically	بشكل دوري periodically

Figure 10: Part Of Speech

2.3 Word Embeddings:

Word embedding is a crucial concept in Natural Language Processing (NLP) that involves representing words as numerical vectors in a continuous vector space. The main goal of word embedding is to convert text data, which machines cannot interpret directly, into a form that can be understood and processed by machine learning algorithms. Unlike traditional methods that treat words as independent symbols, word embeddings aim to capture the semantic relationships and contextual similarities between words. Words that appear in similar contexts or share similar meanings are mapped to vectors that are close to each other in the embedding space. This approach is especially valuable for complex languages like Arabic, which has rich morphology and a wide variety of word forms derived from the same root. Word embeddings help reduce the dimensionality of textual data and allow the system to better generalize and relate different word forms or expressions. In the context of our Arabic Book Search Engine, word embedding allows us to go beyond exact word matching and understand the semantic intent of user queries. This leads to more accurate, intelligent, and user-friendly search results. By embedding both the user's query and the book content into a common vector space, we can measure their similarity more effectively and return the most relevant books. In the following sections, we will discuss various word representation techniques used in our project,
including:

- One-Hot Encoding
- Bag of Words (BoW)
- TF-IDF
- Keras Embedding Layer



- هل هذا الشئ حي ؟
- هل قادر علي التحدث ؟
- هل هو ذكر ؟
- هل هو ملموس ؟
- هل يمكن أكله ؟
- هل يمكن بيعه و شراءه ؟
- هل يتقدم في العمر ؟

و هنا يمكن عمل مصفوفة بسيطة هكذا :

المعيار	الصبر	رجل	تفاحة	كلب	كتاب
هل هذا الشئ حي ؟	لا	نعم	نعم	نعم	لا
هل قادر علي التحدث ؟	لا	نعم	لا	نعم	لا
هل هو ذكر ؟	نعم	نعم	لا	نعم	نعم
هل هو ملموس ؟	لا	نعم	نعم	نعم	نعم
هل يمكن أكله ؟	لا	لا	نعم	لا	لا
هل يمكن بيعه و شراءه ؟	لا	لا	نعم	نعم	نعم
هل يتقدم في العمر ؟	لا	نعم	نعم	نعم	لا

Figure 11: Word Embeddings

2.3.1 One-Hot Encoding:

One-Hot Encoding is one of the simplest methods for representing words as numerical vectors. In this technique, each unique word in the vocabulary is assigned a binary vector of the same length as the vocabulary size. This vector contains a single ‘1’ at the index corresponding to the word’s position in the vocabulary, and **‘0’**s in all other positions. For example, if the vocabulary contains the words: ["كتاب", "مؤلف", "رواية", "مكتبة"] Then the word "رواية" would be represented as: [0, 0, 1, 0] .

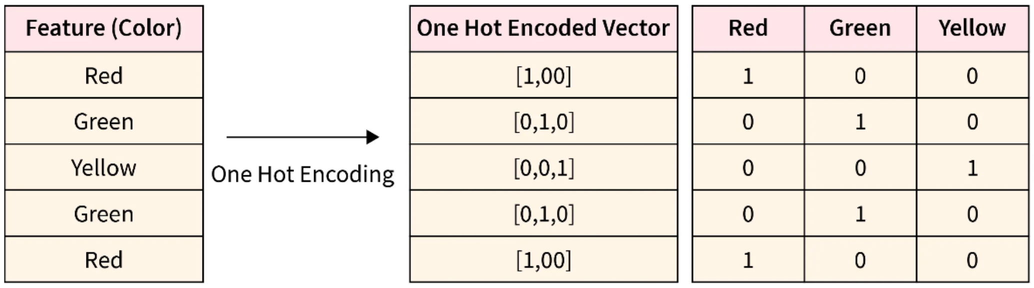
Advantages:

- Simplicity: Very easy to implement and understand.
- No assumptions: It does not assume any relationship between words.

Limitations:

- High Dimensionality: For large vocabularies, the resulting vectors become very sparse and memory-inefficient.
- No Semantic Meaning: One-hot vectors do not capture any relationship between words. For example, the words "كتاب" (book) and "رواية" (novel) are just as distant from each other as they are from unrelated words.
- Scalability Issues: As the vocabulary grows, the vector size increases, making it less practical for real-world NLP tasks.

Due to these limitations, One-Hot Encoding is rarely used in advanced NLP applications today. However, it is still a foundational concept and served as a starting point in our project before exploring more sophisticated methods like TF-IDF and word embeddings.


One Hot Encoding

Feature (Color)
Red
Green
Yellow
Green
Red

One Hot Encoded Vector	Red	Green	Yellow
[1,00]	1	0	0
[0,1,0]	0	1	0
[0,0,1]	0	0	1
[0,1,0]	0	1	0
[1,00]	1	0	0

Figure 12: One-Hot Encoding

2.3.2 Bag of Words:

The Bag of Words (BoW) model is a commonly used method for representing text data in Natural Language Processing. Unlike One-Hot Encoding, which focuses on individual words, BoW represents a document (such as a book description or a user query) by counting the frequency of each word in the vocabulary. In this model, the structure or order of the words is not considered—only the number of times each word appears. The result is a fixed-length vector for each document, where each element represents the count of a specific word from the vocabulary. Example: Assume the vocabulary is: ["كتاب", "مؤلف", "رواية", "مكتبة"] If a document contains the sentence: "كتاب رواية كتاب" The BoW vector would be: [2, 0, 1, 0] This indicates that "كتاب" appears twice, "رواية" once, and the others not at all.

Advantages:

- Simple and Effective: Easy to implement and works well for many basic text classification and retrieval tasks.

- Captures Frequency Information: Reflects how important a word might be within a document.

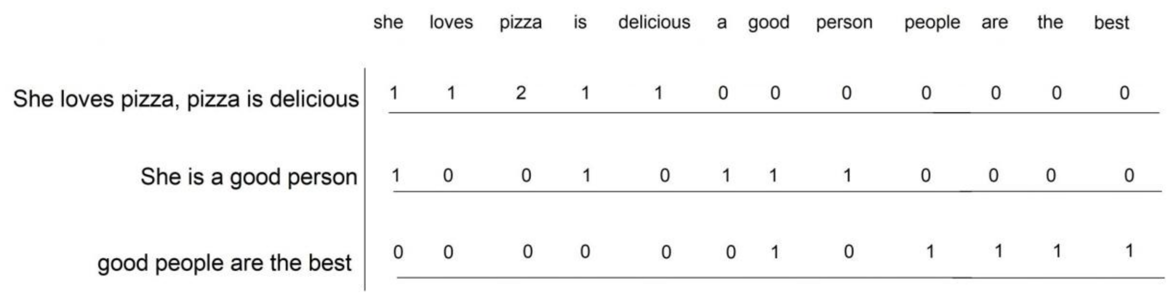
Limitations:

- Ignores Word Order: BoW does not consider the sequence or context in which words appear.

- No Semantic Understanding: Similar to One-Hot Encoding, it treats each word as independent and unrelated to others.

- High Dimensionality: For large vocabularies, BoW vectors can still be sparse and computationally expensive.

In our Arabic Book Search Engine, BoW was used in early experimentation to represent book descriptions and queries. While it provided a basic form of matching, it lacked the ability to understand context and semantic relationships—especially important in the Arabic language. Therefore, we extended our approach using more advanced techniques like TF-IDF and word embeddings to improve search relevance and performance.



	she	loves	pizza	is	delicious	a	good	person	people	are	the	best
She loves pizza, pizza is delicious	1	1	2	1	1	0	0	0	0	0	0	0
She is a good person	1	0	0	1	0	1	1	1	0	0	0	0
good people are the best	0	0	0	0	0	0	1	0	1	1	1	1

Figure 13: Bag of Words

2.3.3 Keras Embedding Layer:

Definition:
In the development of the Arabic book search engine, we experimented with multiple techniques to convert textual data into numerical representations (embeddings), one of which was the Keras Embedding layer. The Keras Embedding layer is typically used in supervised learning settings where text input is associated with target labels (e.g., for sentiment classification or text categorization). It works by learning a dense vector representation of words during the training process based on the loss function and target labels provided. However, in our case, the project involves unsupervised learning — specifically, building a search engine that indexes books and retrieves them based on query similarity. Since we do not have labeled data, the Embedding layer in Keras cannot be trained effectively; it has no loss signal to optimize against. As a result, the embeddings remain either randomly initialized or meaningless for our use case. Due to this limitation, we shifted our focus to alternative methods better suited for unsupervised contexts, such as TF-IDF, pre-trained transformer models, and sentence-level embedding techniques that do not rely on labeled data.
The Keras Embedding layer is a powerful tool often used in Natural Language Processing (NLP) tasks within supervised learning settings. Its main purpose is to map discrete word indices into continuous dense vectors (embeddings) that capture semantic and syntactic relationships between words.

→ How it Works

```
keras.layers.Embedding(input_dim, output_dim, input_length)
```

- input_dim: Size of the vocabulary (total number of unique tokens).
- output_dim: Dimension of the dense embedding vector.
- input_length: Length of the input sequences.

When used in a model, the Embedding layer creates a trainable weight matrix, where each row corresponds to the embedding of a specific word/token. During training, these embeddings are updated through backpropagation, guided by the model's loss function — which in turn depends on the availability of target labels.

→ **Attempting This with Keras Embedding**

Initially, we considered using the Keras Embedding layer to generate embeddings for both the query and documents. The plan was to:

- Tokenize the documents and the user query.
- Pass the sequences through a shared Keras Embedding layer.
- Average or pool the output embeddings to get fixed-size vectors.
- Compute **cosine similarity** between the query vector and each document vector.

However, this approach encountered **a fundamental problem**:

→ **Core Issue: No Training Signal**

The Keras Embedding layer does not come pre-trained (unless you manually load weights). By default, it initializes a matrix of word vectors randomly and expects the embeddings to be learned during training via backpropagation. But in our architecture:

- We don't have labels or a loss function to train against.
- We don't have pairs of query and matching documents to supervise training.
- We are not training a model end-to-end — we're just generating vectors and computing similarity.

As a result, the embeddings produced by Keras remained random and meaningless. When we tried to compute cosine similarity using these embeddings:

- The results were unpredictable and irrelevant.
- There was no learning or fine-tuning, so the vectors had no semantic structure.
- The search results did not reflect the actual meaning or relevance of the input query.

→ Why This Fails in Practice
The Keras Embedding layer works well only in supervised learning pipelines like sentiment analysis or text classification, where you can train a model like:

```
model = Sequential([
   Embedding(input_dim, output_dim, input_length),
   GlobalAveragePooling1D(),
   Dense(1, activation='sigmoid')  # Or softmax for multi-class
])
```

Figure 14: Simple neural network architecture for text classification

In such cases, the model learns meaningful embeddings by adjusting the embedding matrix to minimize a loss function (e.g., cross-entropy). Without this training phase, the embedding matrix remains random, and its output vectors are useless for similarity comparisons.

→ Why We Switched to Better Alternatives
Because our project required **ready-to-use semantic embeddings** without supervised training, we moved to alternatives that are more appropriate for information retrieval:

- TF-IDF: Uses token frequency to capture keyword relevance. Though shallow, it doesn't require labels or training.
- Pre-trained Transformer Models: Such as AraBERT or multilingual BERT, which produce contextualized embeddings that capture semantic meaning out-of-the-box.
- Sentence Transformers: Like SBERT or its Arabic equivalents, optimized specifically for similarity tasks and sentence-level embeddings.

These methods allowed us to generate meaningful query and document vectors, making cosine similarity a reliable metric for search relevance.

2.3.4 TF-IDF

TF-IDF (Term Frequency – Inverse Document Frequency)
TF-IDF is a commonly used technique in Natural Language Processing (NLP) to evaluate how important a word is to a specific document relative to a collection (or corpus) of documents. It is especially useful in information retrieval systems, as it highlights words that are significant to a particular document while downplaying commonly used terms.

TF-IDF is composed of two main components: Term Frequency (TF) and Inverse Document Frequency (IDF).

1. Term Frequency (TF)
This measures how often a term appears in a document:

$$TF(t,d) = f < sub > t,d < /sub > / N < sub > d < /sub >$$

where:

$$t = term$$

$$d = document$$

$$f < sub > t,d < /sub >= number\ of\ times\ term\ t\ appears\ in\ document\ d$$

$$N < sub > d < /sub > = total\ number\ of\ terms\ in\ document\ d$$

2. Inverse Document Frequency (IDF)
This reduces the weight of terms that occur frequently across all documents:

$$IDF(t) = log(N / (1 + n < sub > t < /sub >))$$

where:

$$N = total\ number\ of\ documents\ in\ the\ corpus$$
$$n < sub > t < /sub > = number\ of\ documents\ containing\ the\ term\ t$$

3. TF-IDF Score
The final TF-IDF score is computed by multiplying TF and IDF:

$$TF - IDF(t,d) = TF(t,d) \times IDF(t)$$

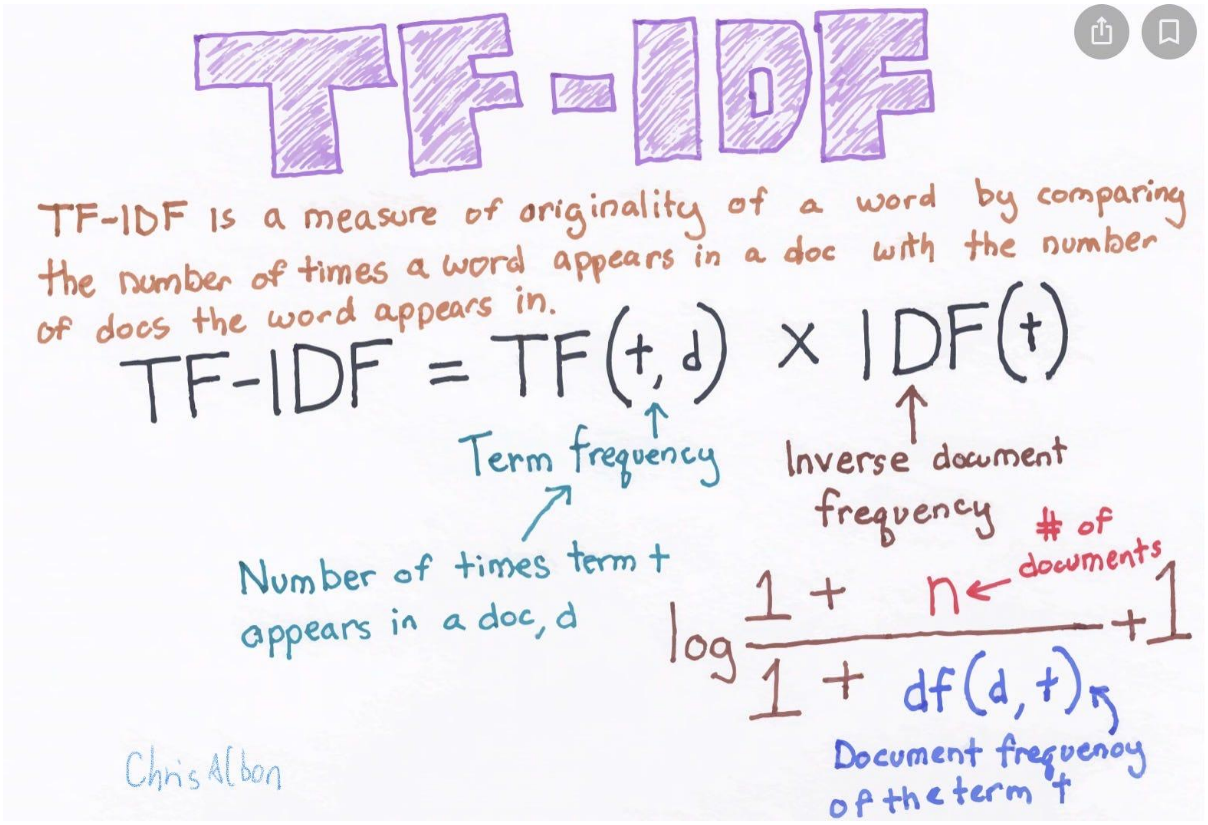

TF-IDF
TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.
TF-IDF = TF(t, d) × IDF(t)
Term frequency
Inverse document frequency
Number of times term t appears in a doc, d
# of documents
log (1 + n)/(1 + df(d, t)) + 1
Document frequency of the term t
Chris Albon

Figure 15: TF-IDF

Our Usage in the Project:
During the early stages of our project, we experimented with TF-IDF to measure the similarity between user queries and Arabic book descriptions. It showed better performance than basic Bag of Words, especially in reducing the weight of frequent, less informative Arabic words.

However, due to the limitations of TF-IDF in handling the morphological complexity of Arabic and its inability to capture semantic relationships between words, we decided not to use it in the final version of our search engine. Instead, we adopted more advanced word embedding techniques that provided better performance and contextual understanding.

Nonetheless, testing TF-IDF gave us valuable insights into the strengths and weaknesses of traditional keyword-weighting methods and helped guide our decision toward more suitable models for the Arabic language.

2.4 Transformer:

The transformer architecture was developed at Google in 2017 for use in a translation model.1 It's a sequence-to-sequence model capable of converting sequences from one domain into sequences in another domain. For example, translating French sentences to English sentences. The original transformer architecture consists of two parts: an encoder and a decoder. The encoder converts the input text (e.g., a French sentence) into a representation, which is then passed to the decoder. The decoder uses this representation to generate the output text (e.g., an English translation) autoregressively.1 Notably, the size of the output of the transformer encoder is linear in the size of its input. Figure 1 shows the design of the original transformer architecture. The transformer consists of multiple layers. A layer in a neural network comprises a set of parameters that perform a specific transformation on the data. In the diagram you can see an example of some layers which include Multi-Head Attention, Add & Norm, Feed-Forward, Linear, Softmax etc. The layers can be sub-divided into the input, hidden and output layers. The input layer (e.g., Input/Output Embedding) is the layer where the raw data enters the network. Input embeddings are used to represent the input tokens to the model. Output embeddings are used to represent the output tokens that the model predicts. For example, in a machine translation model, the input embeddings would represent the words in the source language, while the output embeddings would represent the words in the target language. The output layer (e.g., Softmax) is the final layer that produces the output of the network. The hidden layers (e.g., Multi-Head Attention) are between the input and output layers and are where the magic happens!

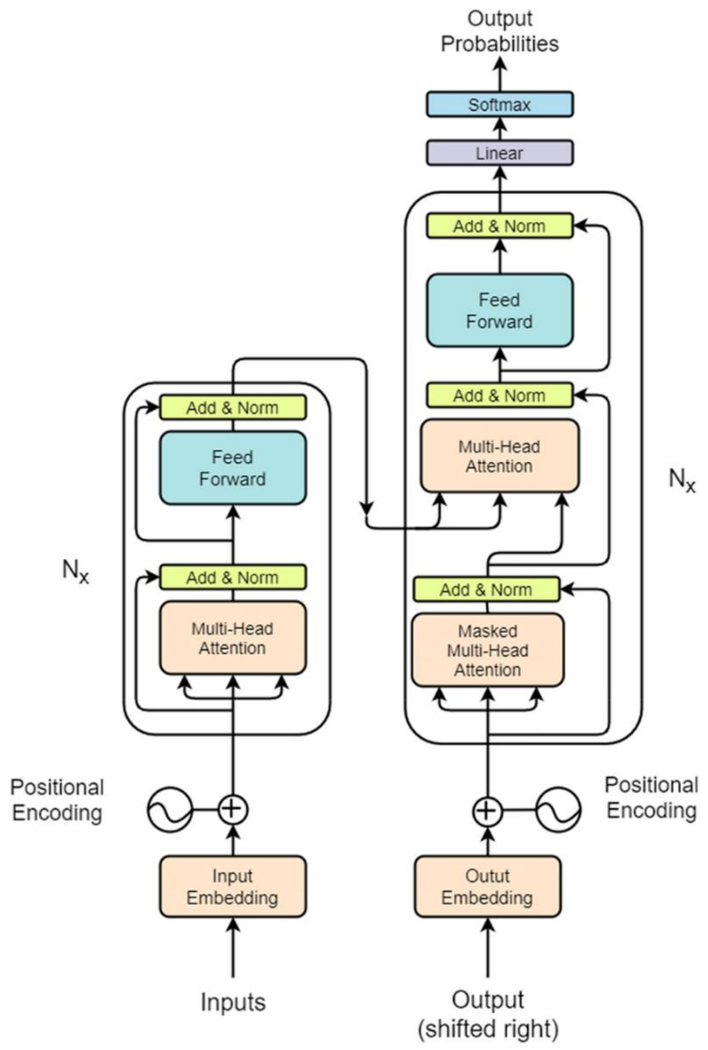


Figure 16: Transformer

Input preparation and embedding To prepare language inputs for transformers, we convert an input sequence into tokens and then into input embeddings. At a high level, an input embedding is a high-dimensional vector68 that represents the meaning of each token in the sentence. This embedding is then fed into the transformer for processing.

Generating an input embedding involves the following steps:

1. Normalization (Optional): Standardizes text by removing redundant whitespace, accents, etc.

2. Tokenization: Breaks the sentence into words or subwords and maps them to integer token IDs from a vocabulary.

3. Embedding: Converts each token ID to its corresponding high-dimensional vector, typically using a lookup table. These can be learned during the training process.

4. Positional Encoding: Adds information about the position of each token in the sequence to help the transformer understand word order. These steps help to prepare the input for the transformers so that they can better understand the meaning of the text.

The encoder is made up of stacked layers (usually 6–12 in modern models). Each layer includes:

- Multi-Head Self-Attention,
    - Learns how different tokens relate to each other.
    - Helps capture context around each word.

- Feed-Forward Networks,
    - Nonlinear transformation to project representations into richer spaces.

- Layer Normalization and Residual Connections,
    - Stabilizes training and improves learning.

Chapter 3: Dataset

3.1 Jamalon Arabic Books

Early Dataset Exploration:
Jamalon At the beginning of our project, one of the most critical steps was finding a suitable dataset of Arabic books that could support the development of our NLP-based search engine. Our first attempt was with the Jamalon dataset, a publicly available dataset sourced from the website of Jamalon—the largest online bookstore in the Middle East, known for offering more than 9.5 million Arabic and English book titles across a wide range of categories.

The dataset consisted of information for over 8,000 Arabic books, with each entry containing key metadata such as the book's title, author, category, and description. Initially, the dataset appeared ideal for our needs due to its size, diversity, and clean structure.

However, upon deeper inspection and preprocessing, we encountered a significant issue: nearly half of the book entries had null or missing values in the description field. Since our search engine heavily relies on textual descriptions to understand and match user queries using NLP techniques, this missing data drastically limited the dataset's usefulness.

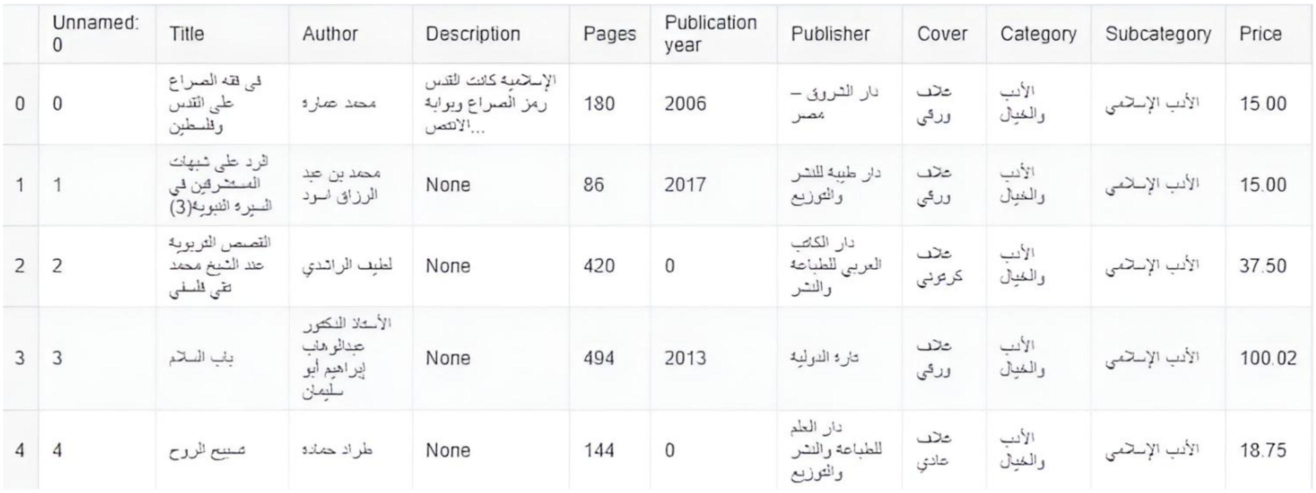
To maintain the quality and integrity of our project, we filtered out all entries with empty or incomplete descriptions. This cleaning process reduced the dataset to approximately 4,000 usable records. While this was enough for initial experimentation and testing, it was too small and limited in scope to support a scalable and semantically rich search system.

Despite this setback, working with the Jamalon dataset was valuable.
It helped us:

- Test our preprocessing pipeline
- Apply and evaluate initial NLP techniques (like text normalization and tokenization) And understand the practical challenges of working with real-world Arabic data.

This experience taught us a key lesson early in the project:

- in NLP
- data quality is just as important as data quantity
- especially when dealing with morphologically rich languages like Arabic.

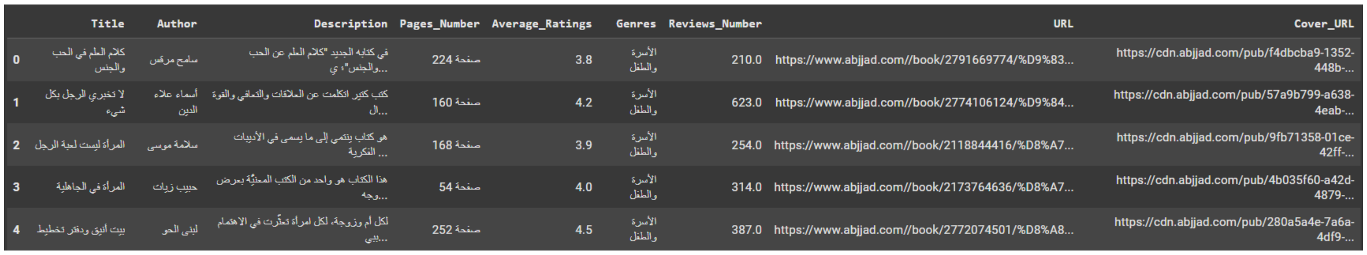


	Unnamed: 0	Title	Author	Description	Pages	Publication year	Publisher	Cover	Category	Subcategory	Price
0	0	فى فقه الصراع على القدس وفلسطين	محمد عماره	الإسلاميه كانت القدس رمز الصراع وبوابة الانتص...	180	2006	دار الشروق – مصر	غلاف ورقي	الأدب والخيال	الأدب الإسلامي	15.00
1	1	الرد على شبهات المستشرقين في السيره النبويه(3)	محمد بن عبد الرزاق اسود	None	86	2017	دار طيبة للنشر والتوزيع	غلاف ورقي	الأدب والخيال	الأدب الإسلامي	15.00
2	2	القصص التربويه عند الشيخ محمد تقي فلسفي	لطيف الراشدي	None	420	0	دار الكاتب العربي للطباعة والنشر	غلاف كرتوني	الأدب والخيال	الأدب الإسلامي	37.50
3	3	باب السلام	الأستاذ الدكتور عبدالوهاب إبراهيم أبو سليمان	None	494	2013	داره الدوليه	غلاف ورقي	الأدب والخيال	الأدب الإسلامي	100.02
4	4	تسبيح الروح	طراد حماده	None	144	0	دار العلم للطباعة والنشر والتوزيع	غلاف عادي	الأدب والخيال	الأدب الإسلامي	18.75

Figure 17: sample of Jamalon Dataset

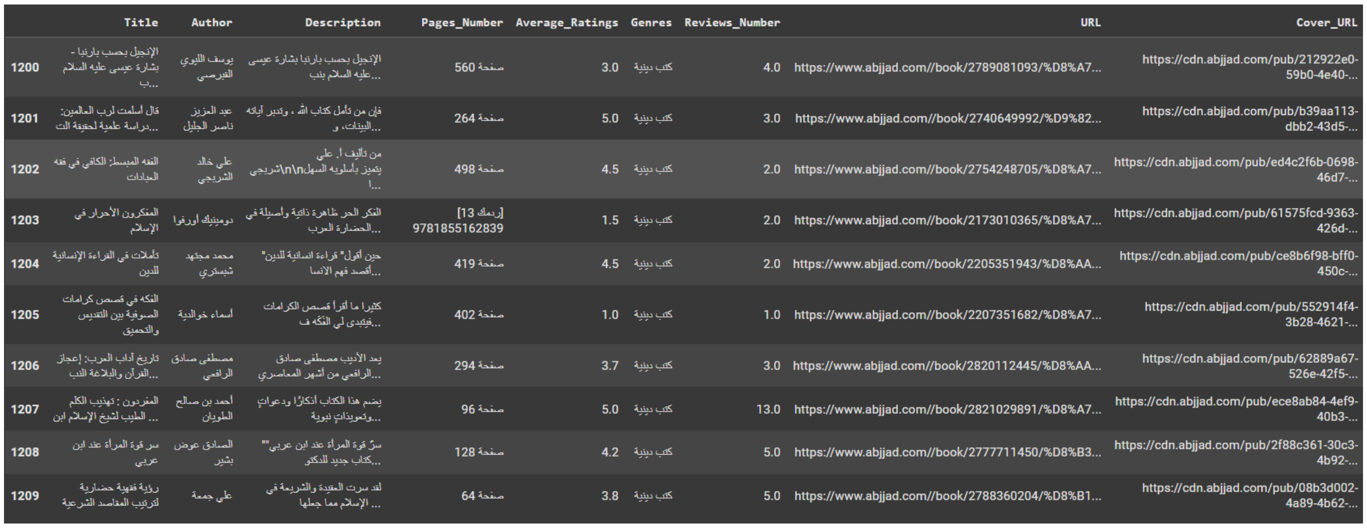
3.2 Abjjad Arabic books:

Final Dataset:
After multiple iterations and data source evaluations, we successfully built a large, custom Arabic book dataset containing approximately 31,000 books, each with detailed textual descriptions. This dataset became the foundation of our Arabic Book Search Engine and was designed specifically to support the Natural Language Processing (NLP) techniques used in our project.



	Title	Author	Description	Pages_Number	Average_Ratings	Genres	Reviews_Number	URL	Cover_URL
0	كلام العلم في الحب والجنس	سامح مرقس	في كتابه الجديد "كلام العلم عن الحب والجنس"، ي...	224 صفحة	3.8	الأسرة والطفل	210.0	https://www.abjjad.com//book/2791669774/%D9%83...	https://cdn.abjjad.com/pub/f4dbcba9-1352-448b-...
1	لا تخبري الرجل بكل شيء	أسماء علاء الدين	كتب كثير اتكلمت عن العلاقات والتعاقي والقوة ال...	160 صفحة	4.2	الأسرة والطفل	623.0	https://www.abjjad.com//book/2774106124/%D9%84...	https://cdn.abjjad.com/pub/57a9b799-a638-4eab-...
2	المرأة ليست لعبة الرجل	سلامة موسى	هو كتاب ينتمي إلى ما يسمى في الأدبيات الفكرية ...	168 صفحة	3.9	الأسرة والطفل	254.0	https://www.abjjad.com//book/2118844416/%D8%A7...	https://cdn.abjjad.com/pub/9fb71358-01ce-42ff-...
3	المرأة في الجاهلية	حبيب زيات	هذا الكتاب هو واحد من الكتب المعنيَّة بعرض وجه...	54 صفحة	4.0	الأسرة والطفل	314.0	https://www.abjjad.com//book/2173764636/%D8%A7...	https://cdn.abjjad.com/pub/4b035f60-a42d-4879-...
4	بيت أنيق ودفتر تخطيط	لبنى الحو	لكل أم وزوجة، لكل امرأة تعثّرت في الاهتمام ببي...	252 صفحة	4.5	الأسرة والطفل	387.0	https://www.abjjad.com//book/2772074501/%D8%A8...	https://cdn.abjjad.com/pub/280a5a4e-7a6a-4df9-...

Figure 18: sample_1 of Abjjad Dataset



	Title	Author	Description	Pages_Number	Average_Ratings	Genres	Reviews_Number	URL	Cover_URL
1200	الإنجيل بحسب بارنبا - بشارة عيسى عليه السلام ب...	يوسف الليوي القبرصي	الإنجيل بحسب بارنبا بشارة عيسى عليه السلام بنب...	560 صفحة	3.0	كتب دينية	4.0	https://www.abjjad.com//book/2789081093/%D8%A7...	https://cdn.abjjad.com/pub/212922e0-59b0-4e40-...
1201	قال أسلمت لرب العالمين: دراسة علمية لحقيقة الت...	عبد العزيز ناصر الجليل	فإن من تأمل كتاب الله ، وتدبر آياته البينات، و...	264 صفحة	5.0	كتب دينية	3.0	https://www.abjjad.com//book/2740649992/%D9%82...	https://cdn.abjjad.com/pub/b39aa113-dbb2-43d5-...
1202	الفقه المبسط: الكافي في فقه العبادات	علي خالد الشربجي	من تأليف أ. علي شربجي\n\nيتميز بأسلوبه السهل ا...	498 صفحة	4.5	كتب دينية	2.0	https://www.abjjad.com//book/2754248705/%D8%A7...	https://cdn.abjjad.com/pub/ed4c2f6b-0698-46d7-...
1203	المفكرون الأحرار في الإسلام	دومينيك أورفوا	الفكر الحر ظاهرة ذاتية وأصيلة في الحضارة العرب...	[ردمك 13] 9781855162839	1.5	كتب دينية	2.0	https://www.abjjad.com//book/2173010365/%D8%A7...	https://cdn.abjjad.com/pub/61575fcd-9363-426d-...
1204	تأملات في القراءة الإنسانية للدين	محمد مجتهد شبستري	حين أقول" قراءة انسانية للدين" أقصد فهم الانسا...	419 صفحة	4.5	كتب دينية	2.0	https://www.abjjad.com//book/2205351943/%D8%AA...	https://cdn.abjjad.com/pub/ce8b6f98-bff0-450c-...
1205	الفكه في قصص كرامات الصوفية بين التقديس والتحميق	أسماء خوالدية	كثيرا ما أقرأ قصص الكرامات فيتبدى لي الفَكَه ف...	402 صفحة	1.0	كتب دينية	1.0	https://www.abjjad.com//book/2207351682/%D8%A7...	https://cdn.abjjad.com/pub/552914f4-3b28-4621-...
1206	تاريخ آداب العرب: إعجاز القرآن والبلاغة النب...	مصطفى صادق الرافعي	يعد الأديب مصطفى صادق الرافعي من أشهر المعاصري...	294 صفحة	3.7	كتب دينية	3.0	https://www.abjjad.com//book/2820112445/%D8%AA...	https://cdn.abjjad.com/pub/62889a67-526e-42f5-...
1207	المفردون : تهذيب الكلم الطيب لشيخ الإسلام ابن ...	أحمد بن صالح الطويان	يضم هذا الكتاب أذكارًا ودعواتٍ وتعويذاتٍ نبوية...	96 صفحة	5.0	كتب دينية	13.0	https://www.abjjad.com//book/2821029891/%D8%A7...	https://cdn.abjjad.com/pub/ece8ab84-4ef9-40b3-...
1208	سر قوة المرأة عند ابن عربي	الصادق عوض بشير	سرّ قوة المرأة عند ابن عربي"" كتاب جديد للدكتو...	128 صفحة	4.2	كتب دينية	5.0	https://www.abjjad.com//book/2777711450/%D8%B3...	https://cdn.abjjad.com/pub/2f88c361-30c3-4b92-...
1209	رؤية فقهية حضارية لترتيب المقاصد الشرعية	علي جمعة	لقد سرت العقيدة والشريعة في الإسلام مما جعلها ...	64 صفحة	3.8	كتب دينية	5.0	https://www.abjjad.com//book/2788360204/%D8%B1...	https://cdn.abjjad.com/pub/08b3d002-4a89-4b62-...

Figure 19: sample_2 of Abjjad Dataset

3.2.1 Dataset Collection

The dataset was collected through web scraping from the Abjjad website using the BeautifulSoup library in Python. Abjjad is a leading Arabic platform for book lovers, featuring a vast collection of Arabic literary and non-literary works, complete with metadata and user-generated content. We developed a scraping script to systematically extract the most relevant information from each book's page while following ethical scraping practices and scraping only publicly available content for academic purposes.

We developed a scraping script to systematically extract the most relevant information from each book's page while following ethical scraping practices and scraping only publicly available content for academic purposes.

Fields Collected
Each book entry in the dataset includes:

Title – The name of the book
Author – The name of the writer
Category – The main genre or subject of the book
Description – A detailed summary of the book's content
Rating (if available) – The average user rating
Cover URL (optional) – A link to the book’s image

Format and Storage
The dataset was saved in both CSV and XLSX formats for easy access and integration with our machine learning pipeline. Each row represents a single book, and each column corresponds to one of the fields listed above.

Size and Quality
Total entries: ~31,000 books
Language: Arabic

Description completeness:
All books include non-empty, meaningful descriptions

Consistency: Data was cleaned and normalized after collection to ensure reliability

Categories and Coverage
We extracted the main category of each book (excluding subcategories), allowing us to maintain a simple and consistent classification across the dataset. The categories include a wide range of subjects, ensuring broad topical coverage for search and retrieval:

الأسرة والطفل	الطب والصحة	كتب علمية	فنون	فلسفة	السيرة الذاتية والمذكرات
كتب دينية	علم النفس وتطوير الذات	السفر والترحال	روايات وقصص	كتب سياسية	
مراجع وأبحاث	لغات	مال وأعمال	تكنولوجيا وإنترنت	كتب الأدب	
قانون	ميثولوجيا وأساطير	الرياضة والتسلية	الصحافة والإعلام	كتب أطفال	

Table 1: Abjjad Categories

This wide coverage enabled us to build a search system that performs well across different user interests and query types.

Why This Dataset Matters
Creating our own dataset gave us full control over the structure, relevance, and completeness of the data. The scale and richness of this dataset allowed us to:

- Apply advanced NLP techniques on real Arabic text.
- Conduct keyword-based search and similarity matching.
- Train and evaluate word embedding models.
- Deliver accurate and meaningful results to end users.

This dataset became the cornerstone of our project and played a critical role in every stage of development—from preprocessing and modeling to evaluation and deployment.

3.3 dataset preparation

Data Preparation To ensure that our Arabic Book Search Engine could process and analyze the dataset effectively, we performed a comprehensive data preparation phase. This phase involved text consolidation, cleaning, and normalization of the Arabic text to make it suitable for Natural Language Processing (NLP) tasks.

1. Text Preprocessing

Before combining the different fields into a single text body, we applied a series of preprocessing steps to clean and normalize the raw Arabic text. This phase was essential for improving the performance of Natural Language Processing (NLP) models by reducing noise and unifying variations in the Arabic language. The steps we followed were:

- **Punctuation Removal:** We removed Arabic and special punctuation characters using regular expressions to avoid unnecessary symbols during tokenization.

- **Repeated Whitespace Removal:** Extra and redundant white spaces were eliminated, leaving a consistent single space between words.

- **Arabic Number Removal:** Arabic numerals (٠ - ٩) were removed to prevent them from influencing search results.

- **Diacritics Removal**: We removed Arabic diacritics (e.g., َ، ُ، ِ) to ensure consistent spelling and reduce lexical variation.

- **Non-Arabic Character Removal:** All non-Arabic characters (e.g., English letters, digits, and symbols) were stripped to retain only Arabic text and whitespace.

- **Link Removal**: URLs and web links were removed using pattern matching to clean the content of unnecessary external data.

- **Stop Word Removal:** Common Arabic stop words were removed using the NLTK library's Arabic stopword list. These words often carry little semantic weight and can reduce search accuracy.

- **Text Normalization:** We standardized several Arabic characters to unify spelling variations:

  i. All forms of Alef ا" → (أ، إ، آ)".
  ii. Teh Marbuta ه" → (ة)".
  iii. Alef Maqsura ي" → (ى)".
  iv. Hamza-based characters (ؤ، ئ) normalized to "و" and "ي".
  v. Persian/Urdu letters (پ، ڤ، چ) converted to Arabic equivalents (ب، ف، ج).

2. Text Consolidation

After cleaning and normalizing each text field, we created a new column called combined_text to hold a structured and comprehensive version of each book's content.
This involved combining the most informative fields for each entry:

- Title.
- Author.
- Genres.
- Description.

This consolidation allowed our system to treat each book as a single, rich document, improving the effectiveness of vectorization methods and similarity-based search.

3. Outcome

The result of the data preparation phase was a clean, standardized, and semantically structured dataset ready for vectorization and modeling. This process significantly improved:

- The performance of keyword-based and embedding-based search.
- The ability of the system to understand query context.
- The consistency and accuracy of Arabic text handling.

By investing effort into preprocessing and structured consolidation, we laid a strong foundation for the success of the search engine’s core NLP functionalities.

Chapter 4: System Architecture

4.1 Keyword based search:

Keyword-based search matches user queries to documents by identifying overlapping terms. It relies on an inverted index that maps keywords to relevant documents, making it fast and effective—especially for short queries with clear, specific terms.

However, this approach has notable limitations:

- It struggles with long queries, where important concepts may be spread across multiple words or phrased differently.
- It fails to understand semantic meaning or user intent, treating words literally without recognizing synonyms, context, or phrasing variations

As a result, keyword-based search may return incomplete or irrelevant results when the user's language differs from the indexed terms, making it less suitable for complex or conversational queries.

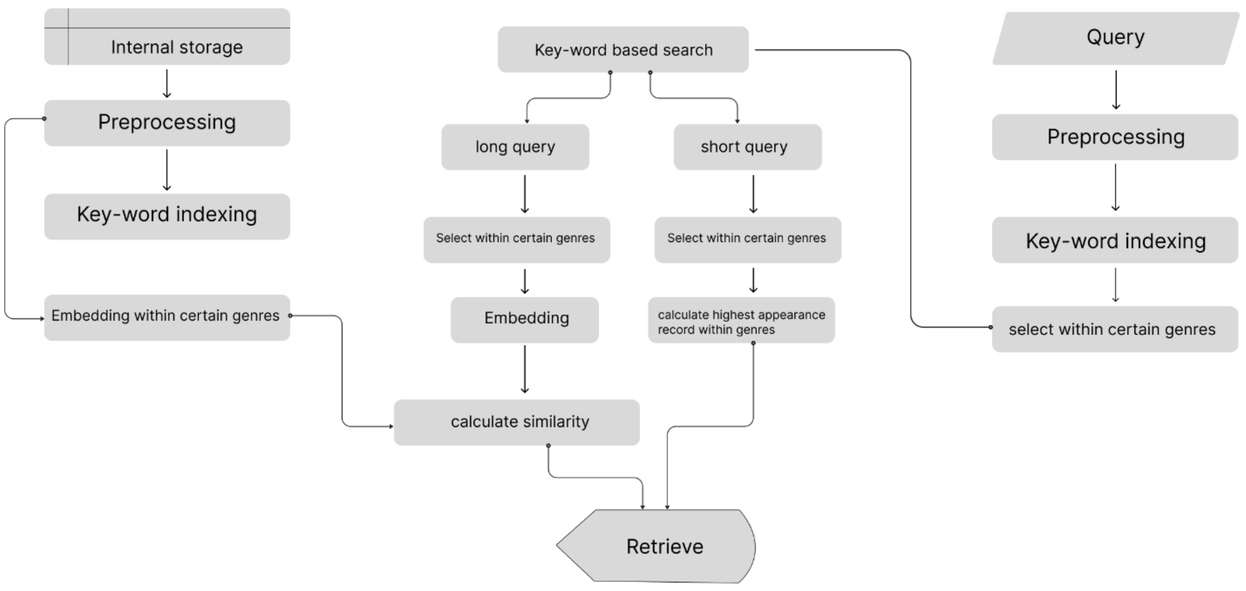

Internal storage
Preprocessing
Key-word indexing
Embedding within certain genres
Key-word based search
long query
short query
Select within certain genres
Select within certain genres
Embedding
calculate highest appearance record within genres
calculate similarity
Retrieve
Query
Preprocessing
Key-word indexing
select within certain genres

Figure 20: system Architecture

4.1.1 Keyword indexing:

Definition:
Keyword indexing maps important words to the documents in which they appear, using an inverted index. This structure supports fast retrieval based on keyword overlap.

How It Works:

- Documents are preprocessed (tokenized, cleaned, stop words removed) Each word is mapped to the document IDs it appears in Inverted Index. Example: { "علم": [1, 5, 12], "اللغة": [3, 7] }.

Strengths:

- High efficiency for short, specific queries Immediate filtering of documents by keyword.

Limitations:

- Fails on long or vague queries Cannot understand context, synonyms, or user intent.

4.2 Embedding-Based Search

Definition:
Embedding-based search uses language models (e.g., AraBERT, all-MiniLM-L6-v2) to convert text into dense semantic vectors.
These vectors are compared using cosine similarity to find relevant documents.

Process:
Generate embedding for the query Compare it with stored document embeddings using cosine similarity Rank documents by similarity.

Strengths:
Captures deep semantic meaning Effective for long or complex queries Robust to synonyms and rephrasing.

Limitations:
Less effective with short queries (limited context) Requires more computational resources Cosine Similarity Formula:
Example: Given a query and top 10 documents, average the cosine similarity scores to evaluate performance.

4.3 Hybrid Search Approach

Definition:
Combines keyword and embedding search to balance speed and semantic relevance.

Strategy:
For short queries with keyword matches → use keyword-based search If no match or query is long → use embedding-based search Benefits: Efficient and fast for short queries Accurate and robust for long or contextual queries.

4.4 retrieving

Semantic search engines compare user queries with each text from a set of documents, calculate a similarity score for each comparison, and get the n-top results based on the similarity scores. To provide an effective search engine over a large corpus Cosine similarity measures how similar two vectors are by calculating the cosine of the angle between them. It's commonly used to compare text embeddings.

$$Cosine\ Similarity = || A ||\cdot|| B || A \cdot B$$

Values range from -1 (opposite) to 1 (identical), with 0 meaning no similarity. To compute average similarity, calculate the cosine similarity between the query vector and each document vector, then take the mean of all scores. This helps identify how semantically close the query is to a group of documents.

Chapter 5: system implementation & experimental results

5.1 Best Practice for all approaches used

There are three primary approaches for text representation and retrieval, each suited for different tasks and requirements:

5.1.1 TF-IDF

Overview: TF-IDF is a statistical method that quantifies the importance of a word in a document relative to a collection of documents. It transforms text into sparse numerical vectors based on word frequency and rarity.

How It Works:

- TF (Term Frequency) measures how often a word appears in a document.
- IDF (Inverse Document Frequency) reduces the weight of common words across all documents.
- The final vector highlights significant words that are frequent in a document but rare across others.

Strengths:

- Simple and computationally efficient.
- Effective for keyword-based search and small-scale applications.

Limitations:

- Ignores word order and semantics.
- Cannot understand context, synonyms, or user intent.
- Performs poorly on long or complex queries.

5.1.2 Keras-Based Deep Learning Models

Overview: Keras provides high-level APIs to build neural networks that learn patterns in text data for tasks like classification or ranking.

How It Works:

- Input text is converted into sequences or embedded vectors.

Models such as:

- DNNs (Dense Neural Networks) capture general patterns.
- CNNs (Convolutional Neural Networks) extract local features.
- LSTMs/GRUs (Recurrent Networks) model sequential dependencies.
- Trained on labeled data to learn query-document relevance.

Strengths:

- Learns complex relationships from training data.
- Customizable to specific tasks (e.g., genre classification).

Limitations:

- Requires large amounts of labeled data.
- Higher training and inference time.
- Less generalizable without extensive tuning.

➔ Why Keras Embedding Layer Was Incompatible with Our Architecture

Our Arabic book search engine follows a semantic search architecture that relies on comparing vector representations (embeddings) of the user's query with those of the book documents in the database. The general flow is as follows:

1. User inputs a query (e.g., a phrase or sentence in Arabic).
2. The query is converted into an embedding vector using the same embedding method used for the documents.
3. Cosine similarity is calculated between the query vector and each document vector.
4. The top N most similar documents are returned to the user as search results.

This architecture is inherently unsupervised: there are no labeled pairs of queries and relevant documents. Instead, the effectiveness of the system depends entirely on how well the embeddings capture semantic meaning.

5.1.3 Transformer-Based Embeddings

Overview: Transformer models use self-attention mechanisms to understand word meaning in context. Models like BERT, AraBERT, and MiniLM produce dense vector embeddings for input text.

How It Works:

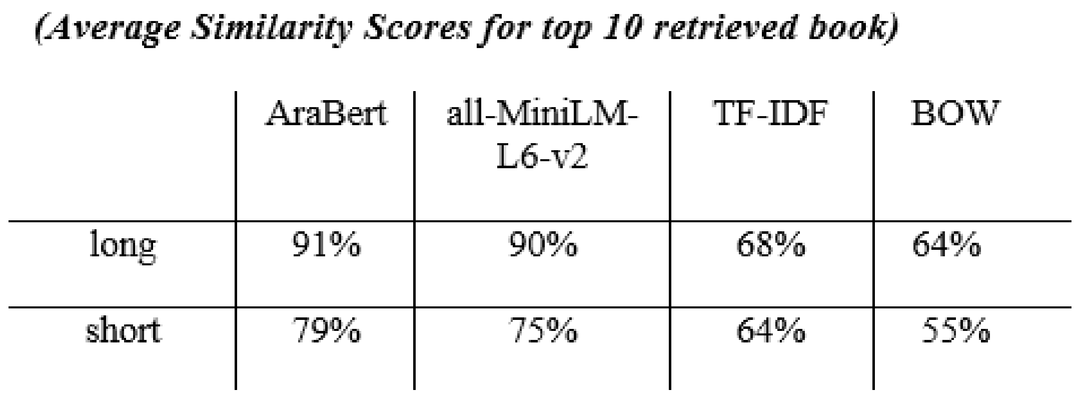
- Entire input (sentence/query/document) is processed to generate a single dense vector.
- Embeddings encode context, semantics, and syntax.
- Search uses cosine similarity between query and document embeddings.

Strengths:

- Captures deep semantic meaning and intent.
- Robust to word variations and paraphrasing.
- Ideal for semantic search, question answering, and long queries.

Limitations:

- Computationally intensive.
- Less effective with short queries that lack meaningful context.
- Requires vector indexing for large-scale retrieval.



(Average Similarity Scores for top 10 retrieved book)

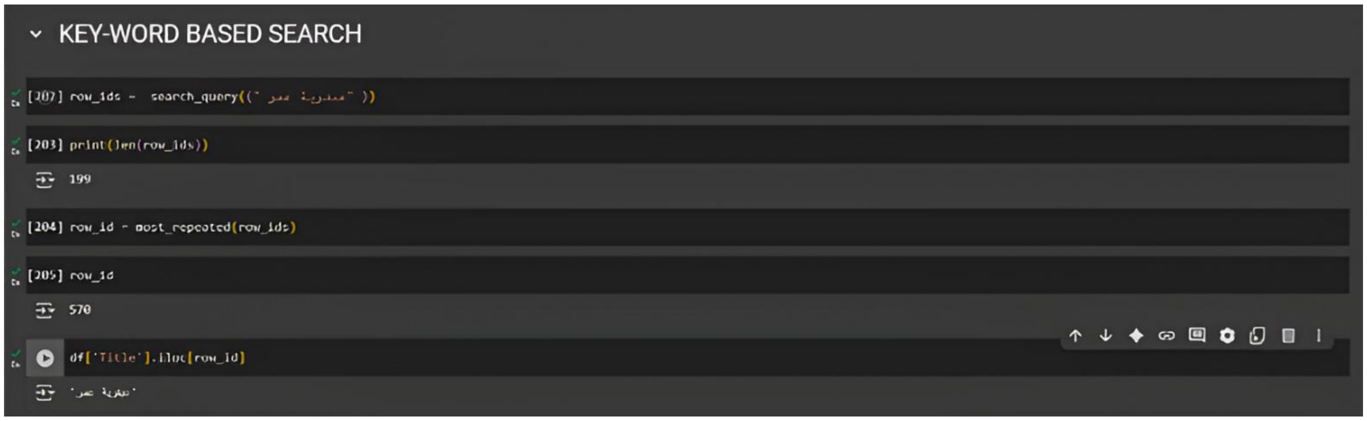
	AraBert	all-MiniLM-L6-v2	TF-IDF	BOW
long	91%	90%	68%	64%
short	79%	75%	64%	55%

Table 2: Average similarity scores

5.2 Best Practices for the final approach

5.2.1 Search keyword

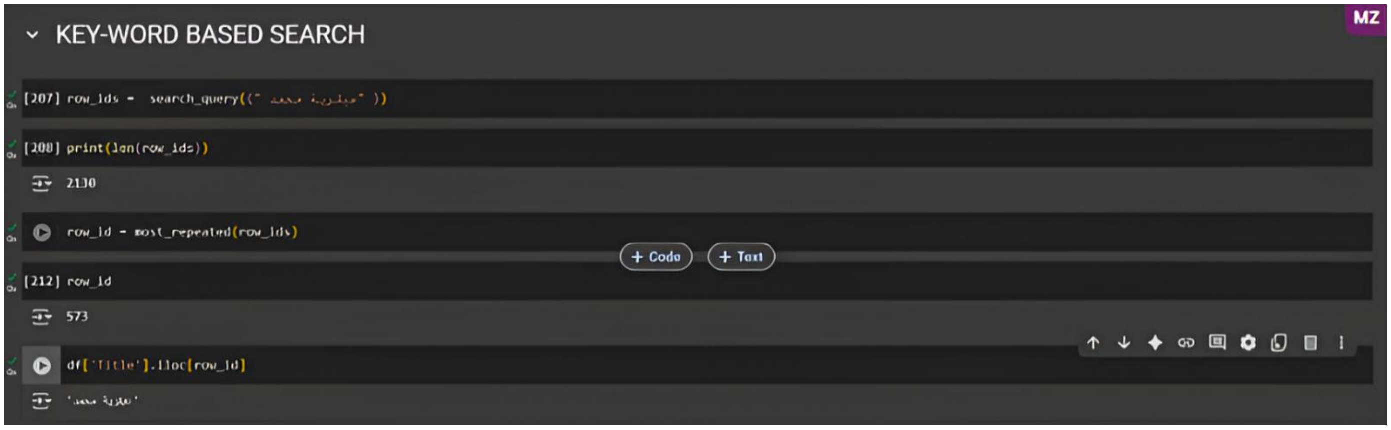
Keyword-based search matches user queries to documents by identifying overlapping terms. It relies on an inverted index that maps keywords to relevant documents, making it fast and effective, especially for short queries with clear, specific terms.



KEY-WORD BASED SEARCH

```
[207] row_ids = search_query((" عبقرية عمر" ))

[203] print(len(row_ids))
199

[204] row_id = most_repeated(row_ids)

[205] row_id
570

df['Title'].iloc[row_id]
'عبقرية عمر'
```

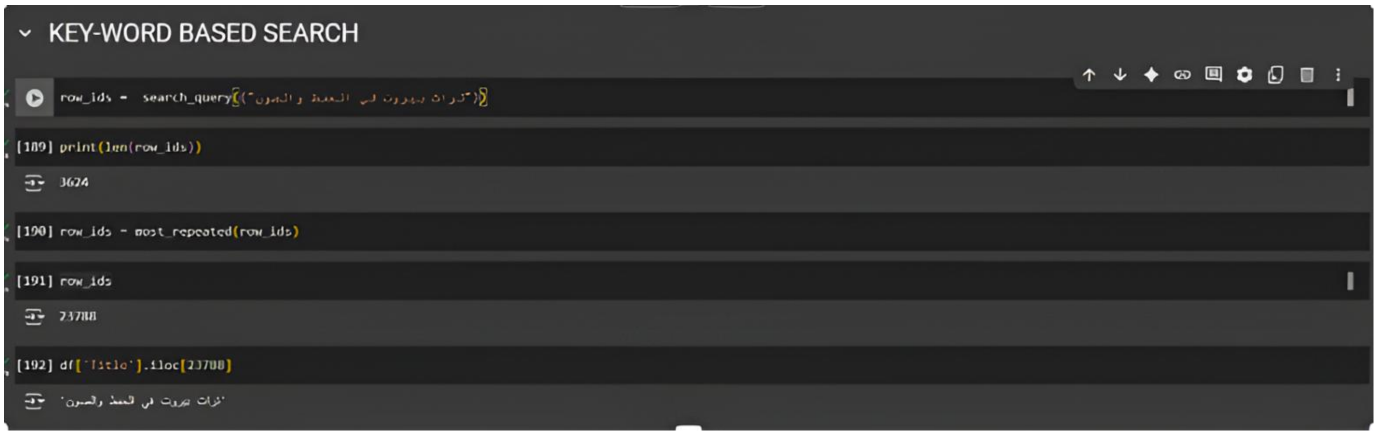
Figure 20: Search keyword Sample_1



KEY-WORD BASED SEARCH

```
[207] row_ids = search_query((" عبقرية محمد" ))

[208] print(len(row_ids))
2130

row_id = most_repeated(row_ids)

[212] row_id
573

df['Title'].iloc[row_id]
'عبقرية محمد'
```

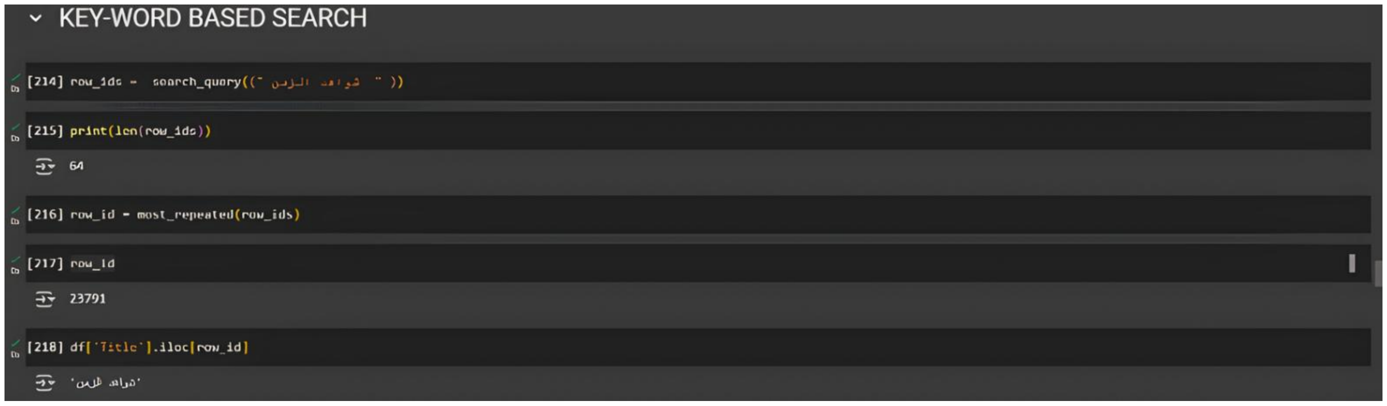
Figure 21: Search keyword Sample_2



KEY-WORD BASED SEARCH

```
row_ids =  search_query(("تراث بيروت في العمة والصور"))
[189] print(len(row_ids))
 3624
[190] row_ids = most_repeated(row_ids)
[191] row_ids
 23788
[192] df['Title'].iloc[23788]
 'تراث بيروت في العمة والصور'
```

Figure 22: Search keyword Sample_3



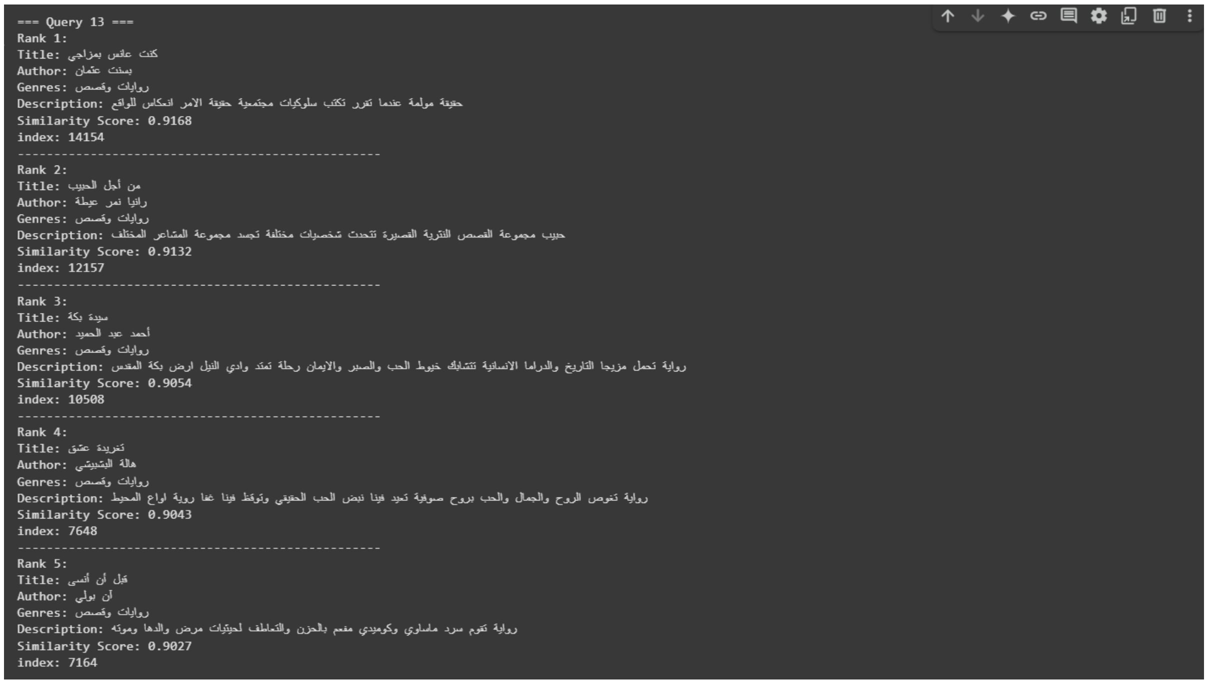
KEY-WORD BASED SEARCH

```
[214] row_ids =  search_query((" شواهد الزمن  " ))
[215] print(len(row_ids))
 64
[216] row_id = most_repeated(row_ids)
[217] row_id
 23791
[218] df['Title'].iloc[row_id]
 'شواهد الزمن'
```

Figure 23: Search keyword Sample_4

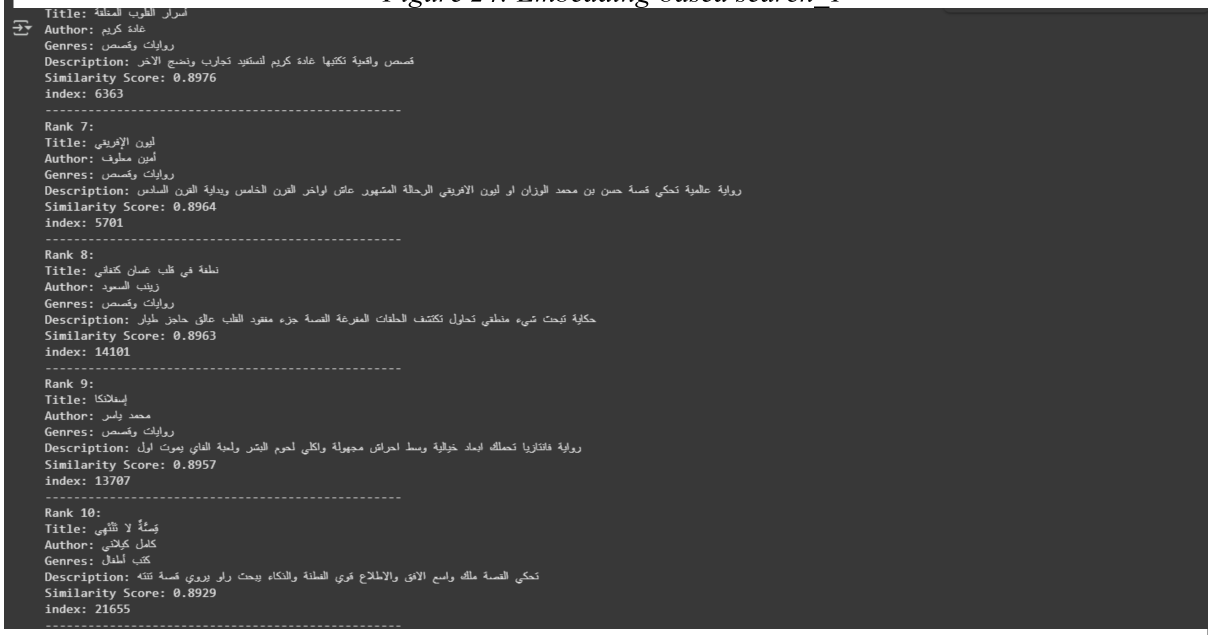
5.2.2 Embedding-based search

represents queries and documents as dense semantic vectors using pretrained language models. It captures the meaning and context of text, enabling accurate matching even when the query and documents use different words or phrasing. This makes it highly effective for long or complex queries, where user intent is better expressed.



```
=== Query 13 ===
Rank 1:
Title: كنت عانس بمزاجي
Author: بسنت عثمان
Genres: روايات وقصص
Description: حقيقة مولمة عندما تقرر تكتب سلوكيات مجتمعية حقيقة الامر انعكاس للواقع
Similarity Score: 0.9168
index: 14154
--------------------------------------------------
Rank 2:
Title: من أجل الحبيب
Author: رانيا نمر عيطة
Genres: روايات وقصص
Description: حبيب مجموعة القصص النثرية القصيرة تتحدث شخصيات مختلفة تجسد مجموعة المشاعر المختلف
Similarity Score: 0.9132
index: 12157
--------------------------------------------------
Rank 3:
Title: سيدة بكة
Author: أحمد عبد الحميد
Genres: روايات وقصص
Description: رواية تحمل مزيجا التاريخ والدراما الانسانية تتشابك خيوط الحب والصبر والايمان رحلة تمتد وادي النيل ارض بكة المقدس
Similarity Score: 0.9054
index: 10508
--------------------------------------------------
Rank 4:
Title: تغريدة عشق
Author: هالة البشبيشي
Genres: روايات وقصص
Description: رواية تغوص الروح والجمال والحب بروح صوفية تعيد فينا نبض الحب الحقيقي وتوقظ فينا غفا روية اواع المحيط
Similarity Score: 0.9043
index: 7648
--------------------------------------------------
Rank 5:
Title: قبل أن أنسى
Author: آن بولي
Genres: روايات وقصص
Description: رواية تقوم سرد ماساوي وكوميدي مفعم بالحزن والتعاطف لحيثيات مرض والدها وموته
Similarity Score: 0.9027
index: 7164
```

Figure 24: Embedding-based search_1

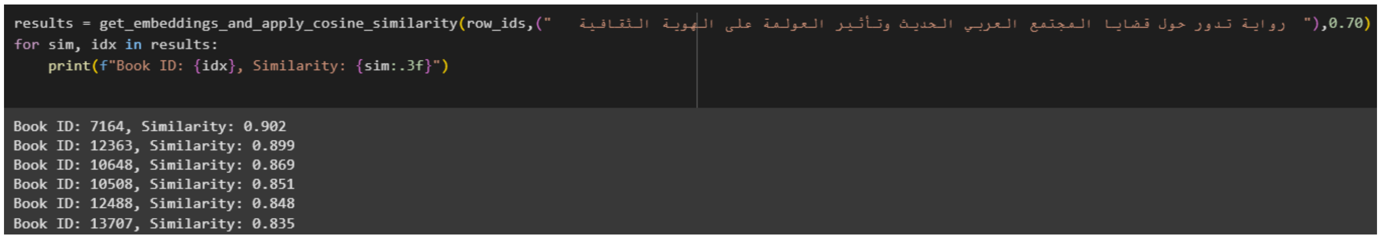


```
Title: أسرار القلوب المغلقة
Author: غادة كريم
Genres: روايات وقصص
Description: قصص واقعية تكتبها غادة كريم لتستفيد تجارب ونضج الاخر
Similarity Score: 0.8976
index: 6363
--------------------------------------------------
Rank 7:
Title: ليون الإفريقي
Author: أمين معلوف
Genres: روايات وقصص
Description: رواية عالمية تحكي قصة حسن بن محمد الوزان او ليون الافريقي الرحالة المشهور عاش اواخر القرن الخامس وبداية القرن السادس
Similarity Score: 0.8964
index: 5701
--------------------------------------------------
Rank 8:
Title: نطفة في قلب غسان كنفاني
Author: زينب السعود
Genres: روايات وقصص
Description: حكاية تبحث شيء منطقي تحاول تكتشف الحلقات المفرغة القصة جزء مفقود القلب عالق حاجز طيار
Similarity Score: 0.8963
index: 14101
--------------------------------------------------
Rank 9:
Title: إسفلانكا
Author: محمد ياسر
Genres: روايات وقصص
Description: رواية فانتازيا تحملك ابعاد خيالية وسط احراش مجهولة واكلي لحوم البشر ولعبة الفاي يموت اول
Similarity Score: 0.8957
index: 13707
--------------------------------------------------
Rank 10:
Title: قِصَّةٌ لا تَنْتَهِي
Author: كامل كيلاني
Genres: كتب أطفال
Description: تحكي القصة ملك واسع الافق والاطلاع قوي الفطنة والذكاء يبحث راو يروي قصة تنته
Similarity Score: 0.8929
index: 21655
--------------------------------------------------
```

Figure 25: Embedding-based search_2

5.2.3 search Multistage

To balance speed and semantic accuracy, we adopted a hybrid search strategy that combines keyword-based and embedding-based methods.
For short queries, the system first checks for keyword matches using an inverted index. If matching terms are found, the system performs fast keyword-based retrieval, which is more efficient and accurate for concise queries where keywords are clear and direct. If no keywords are found, or if the query is long and context-rich, the system falls back to embedding-based search, which leverages dense vector representations to capture deeper semantic meaning and user intent.
This allows the system to retrieve relevant results even when the query and documents use different vocabulary or phrasing. By combining both techniques, the hybrid approach maximizes retrieval quality: it maintains high efficiency for short queries while ensuring robust semantic understanding for longer, more expressive inputs.



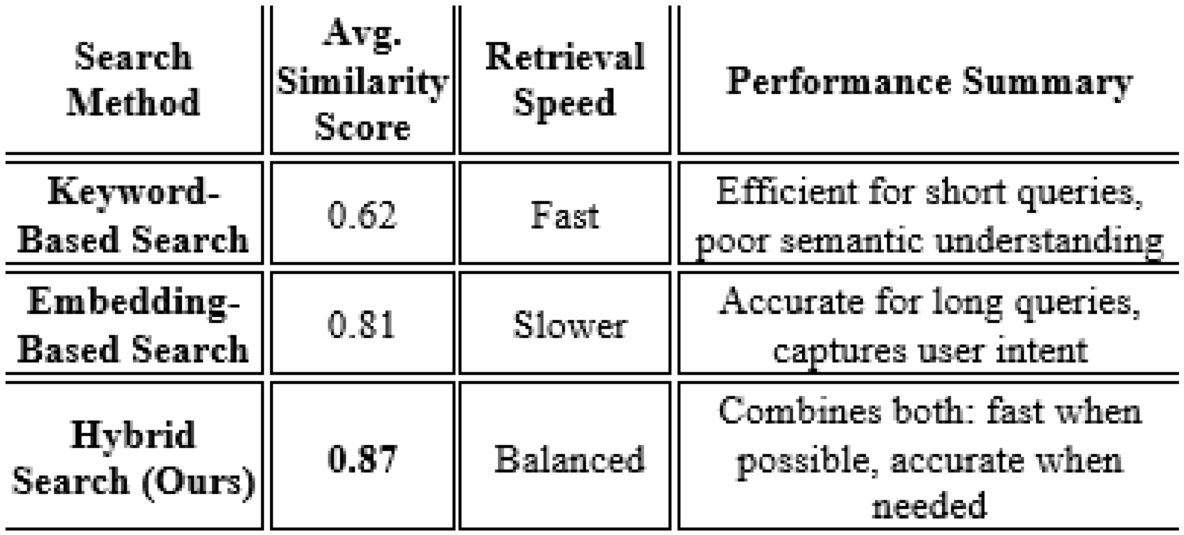
```
results = get_embeddings_and_apply_cosine_similarity(row_ids,("  رواية تدور حول قضايا المجتمع العربي الحديث وتأثير العولمة على الهوية الثقافية  "),0.70)
for sim, idx in results:
    print(f"Book ID: {idx}, Similarity: {sim:.3f}")
```

```
Book ID: 7164, Similarity: 0.902
Book ID: 12363, Similarity: 0.899
Book ID: 10648, Similarity: 0.869
Book ID: 10508, Similarity: 0.851
Book ID: 12488, Similarity: 0.848
Book ID: 13707, Similarity: 0.835
```

Figure 21: search Multistage

5.3 Results

Result for searching approaches:



Search Method	Avg. Similarity Score	Retrieval Speed	Performance Summary
Keyword-Based Search	0.62	Fast	Efficient for short queries, poor semantic understanding
Embedding-Based Search	0.81	Slower	Accurate for long queries, captures user intent
Hybrid Search (Ours)	**0.87**	Balanced	Combines both: fast when possible, accurate when needed

Table 3: Result

Chapter 6: GUI

6.1 Mobile Application Walkthrough

We developed an android mobile application called “مكتبة افق”, The details of the application will be discussed briefly in this section.

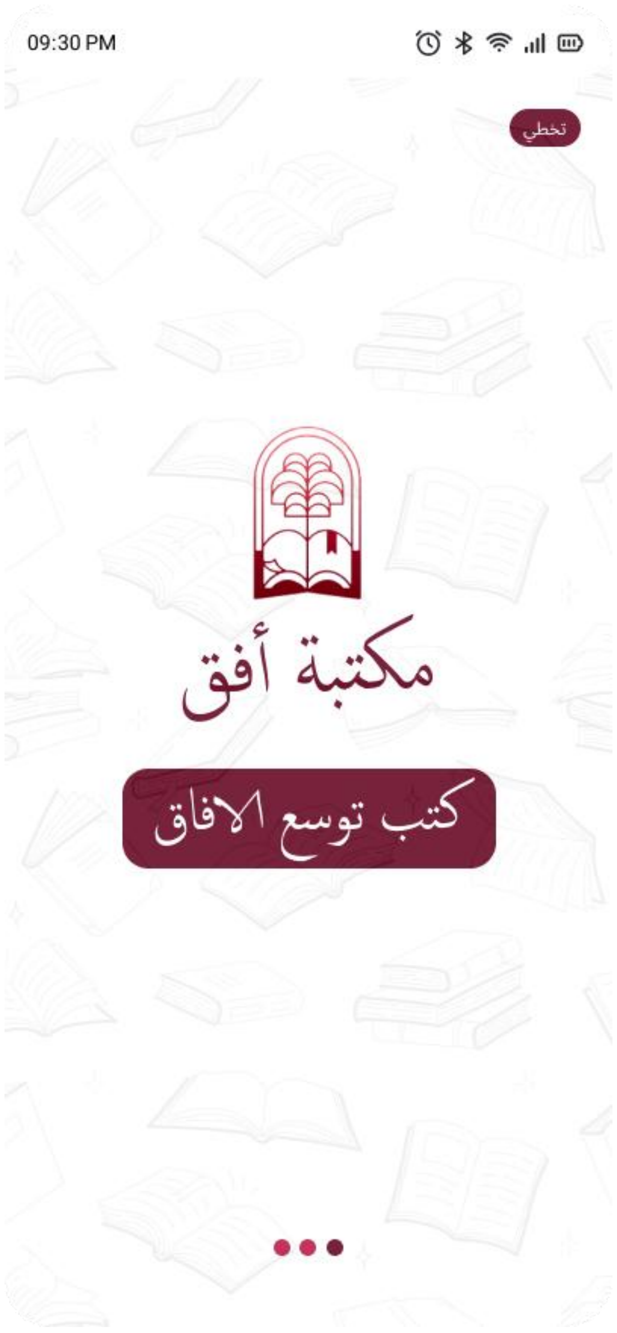
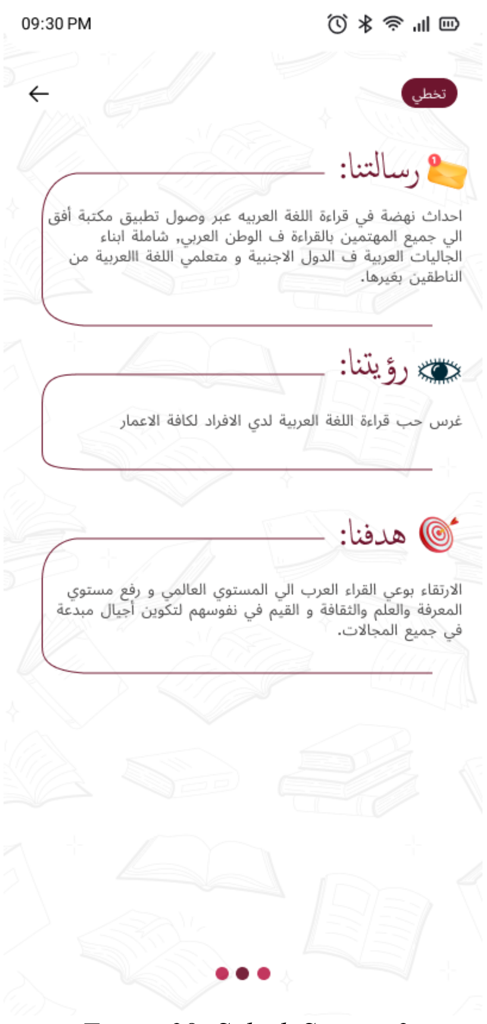


Figure 22: Splash Screen_1

Despite the growth of digital content, **searching for Arabic books** is still a challenge for many. Most search engines are not optimized for Arabic language structure, and that makes finding the right book harder than it should be.
We created this application to provide a **powerful yet simple** solution — a platform that understands Arabic, indexes content effectively, and helps users find the books they need in seconds.
This project reflects our team's mission:
To make Arabic knowledge more accessible, searchable, and user-friendly.
Through this app, we aim to support education, research, and cultural growth in the Arab world.
Join us on this journey to bring Arabic literature closer to everyone — one search at a time.



رسالتنا:

احداث نهضة في قراءة اللغة العربيه عبر وصول تطبيق مكتبة أفق الي جميع المهتمين بالقراءة ف الوطن العربي, شاملة ابناء الجاليات العربية ف الدول الاجنبية و متعلمي اللغة االعربية من الناطقين بغيرها.

رؤيتنا:

غرس حب قراءة اللغة العربية لدي الافراد لكافة الاعمار

هدفنا:

الارتقاء بوعي القراء العرب الي المستوي العالمي و رفع مستوي المعرفة والعلم والثقافة و القيم في نفوسهم لتكوين أجيال مبدعة في جميع المجالات.

Figure 28: Splash Screen_2

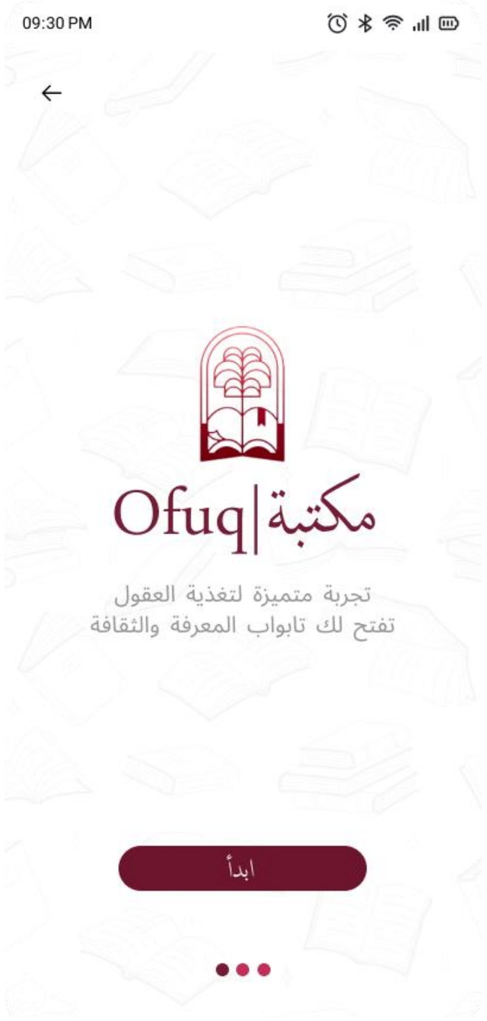
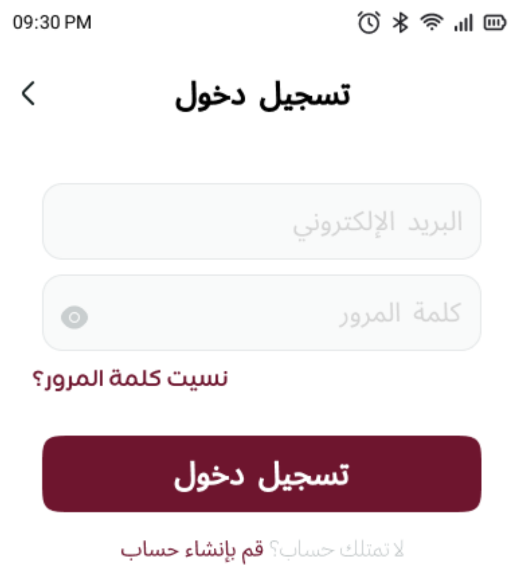


Figure 29: Splash Screen_3

"Here’s the login page — simple, clean, and user-friendly. We made sure returning users can access the app quickly and easily."
"And for new users, this is our sign-up page. We designed it to be as welcoming and smooth as possible, encouraging more readers to join."



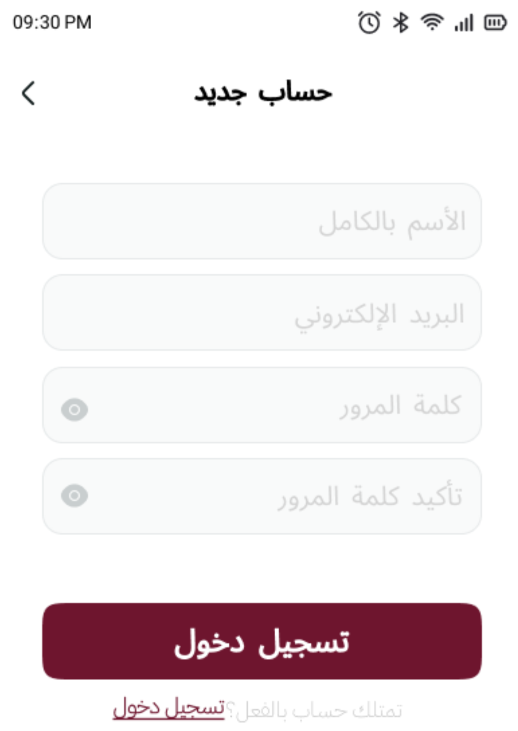


Figure 23: Login Screen

made it easy for users to browse by interest — whether they're looking for novels, history, religion, science, or any other genre.
Our goal here was to make the browsing experience smooth, intuitive, and visually clear, so that users can find what they’re looking for — or even discover something new — in just a few clicks."

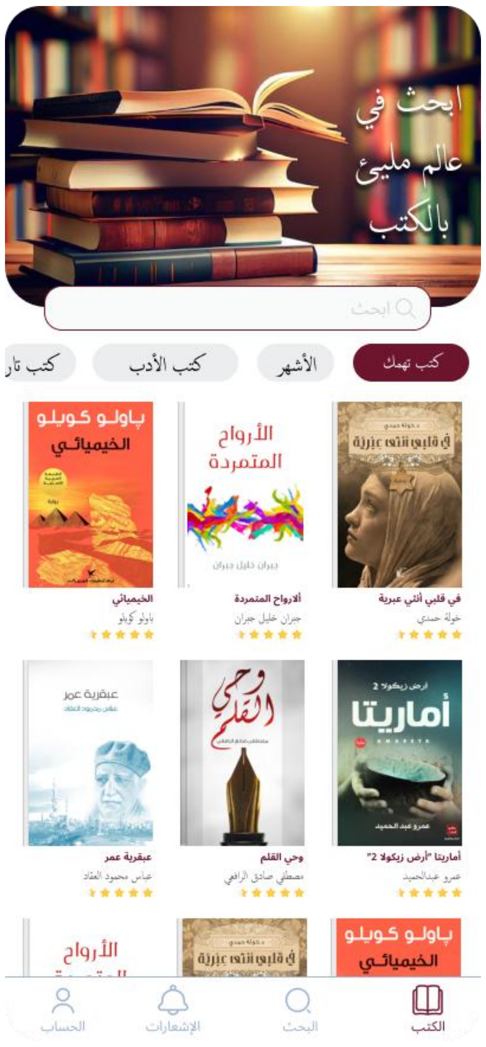


Figure 24: Home Screen

"This is our smart search page — the core of the application. Users can type in keywords, book titles, or even author names in Arabic, and instantly get accurate, relevant results.
We’ve designed the search to be fast, intelligent, and fully optimized for Arabic content."

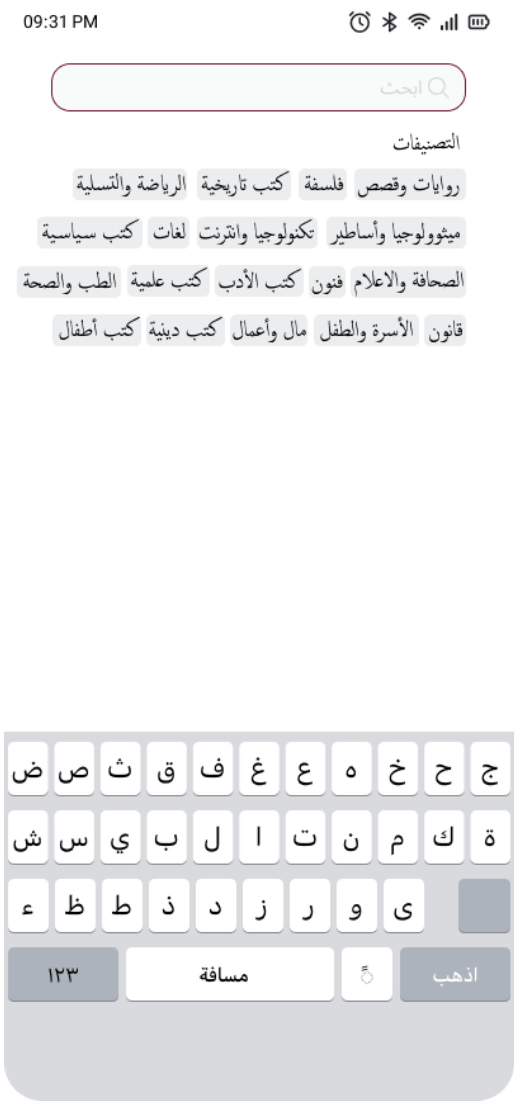
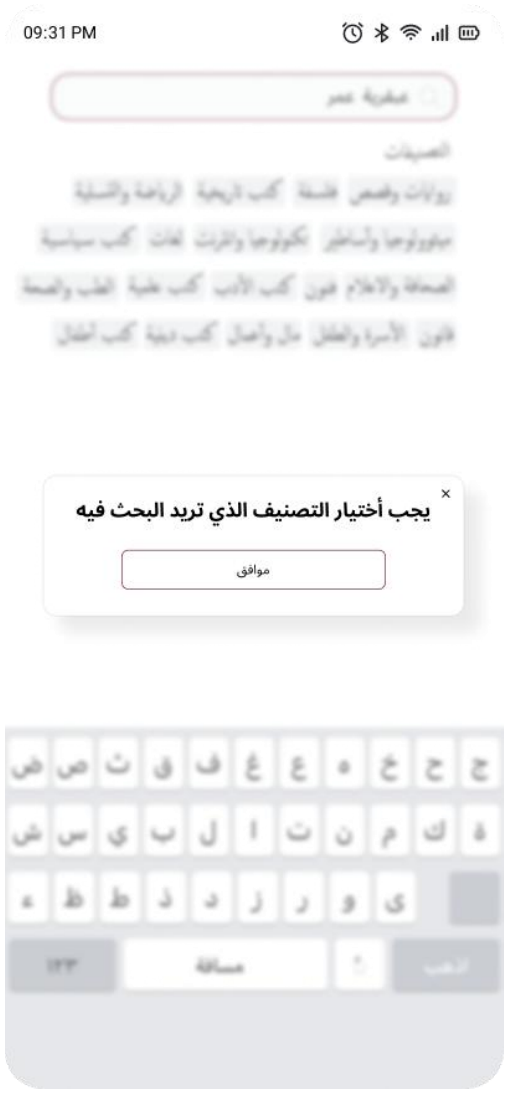


Figure 25: Search Screen

"Please select a category first — it's essential for getting accurate search results. Our smart search is designed to work best when a category is chosen, helping us deliver the most relevant Arabic content based on your input."



يجب أختيار التصنيف الذي تريد البحث فيه

Figure 27: Warning Category Selection

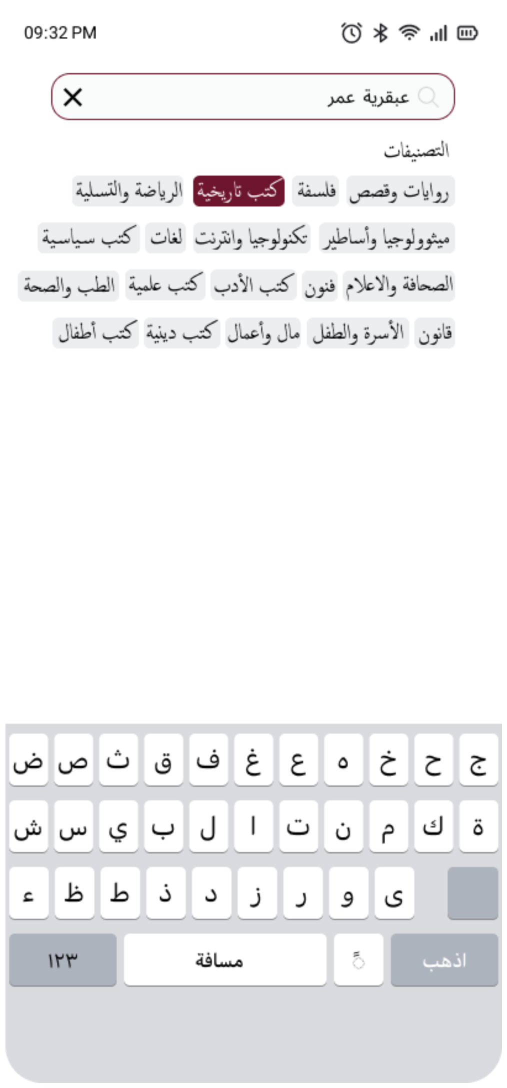
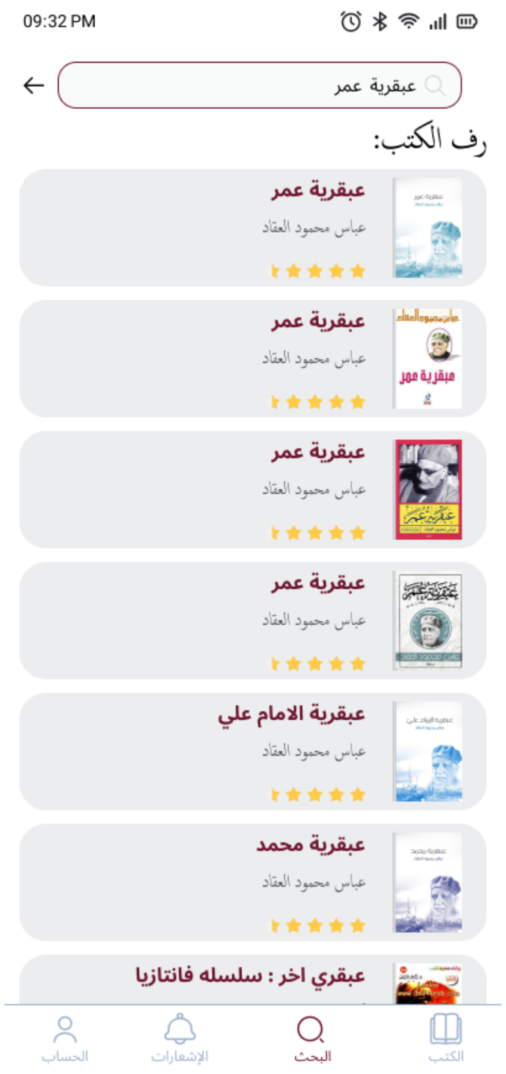


Figure 26: Select Category

"Here are your results carefully filtered and ranked to match your keywords. Whether you searched by title, author, or topic in Arabic, our smart engine ensures you see the most relevant content first."



09:32 PM

عبقرية عمر

رف الكتب:

عبقرية عمر
عباس محمود العقاد

عبقرية عمر
عباس محمود العقاد

عبقرية عمر
عباس محمود العقاد

عبقرية عمر
عباس محمود العقاد

عبقرية الامام علي
عباس محمود العقاد

عبقرية محمد
عباس محمود العقاد

عبقري اخر : سلسله فانتازيا

Figure 28: Result Page

Chapter 7: Conclusion & Future Work

7.1 Conclusion

This work presents a hybrid search engine for Arabic books that intelligently combines keyword-based search with semantic embedding techniques, achieving significant improvements in retrieval accuracy and relevance over traditional approaches. By integrating inverted indexes for fast lexical matching and incorporating transformer-based embeddings for deep semantic understanding, our system effectively addresses the challenges posed by the morphological richness, synonym, and syntactic complexity characteristic of Arabic text.
Key contributions and findings include:

- Enhanced Retrieval Performance.
- Scalable and Efficient Architecture.
- User-Centered Relevance.
- Broader Implications for Arabic NLP.

7.2 Future Work

- Feedback-Driven Refinement: Integrate user relevance feedback to continuously fine-tune ranking models.

- Full-Text Support: Extend indexing and embedding beyond descriptions to include complete book content.

- Multilingual Capability: Support cross-lingual searches by embedding Arabic and foreign-language texts into a shared vector space.

- Production-scale Deployment: Optimize system components for deployment in real-world environments, focusing on efficiency and scalability.

7.3 Tools

7.3.1 Frameworks

Flutter Firebase

PyTorch Flask

TensorFlow

7.3.2 Language

7.3.3 Environment

Google colab

7.3.4 Version control

git GitHub

7.4 References

[1] M. A. Kassimi and A. Essayad, “BERT Representation for Arabic Information Retrieval: A Comparative Study,” Multimedia Research, vol. 6, no. 3, pp. 1–12, 2023.

[2] M. A. Kassimi, H. Abdellatif, and A. Essayad, “Mono-Lingual Search Engine: Combining Keywords with Context for Semantic Search Engine,” in Advances in Intelligent System and Smart Technologies, Lecture Notes in Networks and Systems, vol. 826, Springer, 2024, pp. 353–361.

[3] A. El Hadi, Y. Madani, R. El Ayachi, and M. Erritali, “A New Semantic Similarity Approach for Improving the Results of an Arabic Search Engine,” Procedia Computer Science, vol. 151, pp. 1170–1175, 2019.

[4] Darwish et al. (2021) provide a broad overview, including Arabic IR with discussions on stemming, diacritization, embeddings, and semantic search.

[5] A full pipeline including Arabic text normalization, TF–IDF vectorization, inverted-index construction, and a Flask search API.

[6] Compares TF-IDF against word embeddings for sentence-level similarity in Arabic, analyzing various stemmers’ impact on retrieval performance.

[7] Proposes TF-IDF feature selection for Arabic text classification—a key component in IR pipelines.

[8] Abderrahim et al. (2018) apply word embedding similarity techniques to boost Arabic web search accuracy.