Back-of-the-Book Index Automation for Arabic Documents

Nawal Haidar and Ahmad Kashmar and Fadi Zaraket
Arab Center for Research and Policy Studies
Doha, Qatar
{nhaidar, akashmar, fzaraket}@dohainstitute.edu.qa

Abstract

Back-of-the-book indexes (BoBIs) are crucial for book readability. However, their manual creation is laborious and error prone. In this paper, we introduce ArBoBIM to automate BoBI extraction and review processes for Arabic books. Given a book with a corresponding BoBI, ArBoBIM extracts BoBI terms and identifies their occurrences and aligns those across several versions of the book. ArBoBIM first defines a pool of candidates for each term by leveraging noun phrases and named entities. ArBoBIM leverages several metrics, including exact matches, morpho-lexical similarity, and semantic similarity, to determine the best candidates. We empirically fine-tuned thresholds for ArBoBIM and achieve an F1-score of 0.94 (precision= 0.97, recall=0.91). These results are significantly better than baseline results, and top LLM based results with lower computational cost and no publishing house IP risks. Additionally, with ArBoBIM, over 500 books have been processed, resulting in the ArBoBIMap dataset, containing books, their terms, occurrences, and various metadata related to them, to be made available for the public. This dataset is used to train a model to identify if a term, given its features, should be added to the back-of-the-book index of a specific book. The model achieves an F1-score of 0.91 (precision = 0.97, recall = 0.85).

1 Introduction

In nonfiction literature, the inclusion of a back-of-the-book index (BoBI) is standard practice. BoBIs provide readers with valuable and efficient capabilities to identify important terms and concepts within large books.

Manual construction and review of a BoBI for a given book is a time-consuming task prone to errors. Domain experts, researchers, and research assistants in research and scholastic publishing houses spend valuable time manually performing these tasks. Their institutions prefer to spare them that work as much as possible so that they concentrate their time and effort on more valuable tasks such as original research and authoring work.

Automating the extraction and review of existing BoBIs is therefore an important task that saves time for domain experts, costs for publishing houses, and improves the quality of BoBIs.

While indexing and term extraction tools exist for English texts, these are not mature for Arabic, especially in specific research domains. Therefore, a first step in automating the task for Arabic books, is to build datasets from existing BoBIs.

We introduce ArBoBIM, an Arabic BoBI mapper that takes an Arabic book, extracts the BoBI therein and maps its index terms to their occurrences in the book. ArBoBIM is essential for the following reasons.

- Reviewing and correcting the terms in the BoBI, and specifying their occurrences,
- Augmenting the BoBI with potential missing terms from the book,
- Building and augmenting datasets to train models for BoBI extraction and revision.

To develop ArBoBIM, we transformed existing BoBIs, typically found at the end of a book, into accurate structured index maps with detailed descriptions of the occurrences of each term, including its exact forms and positions within the text of the book(s).

BoBIs already contain page numbers for each term; however, several challenges exist.

First, final BoBI page numbers refer to production versions of the books. These include final layout and design elements introduced by graphic design and layout management teams towards the last stages of production. They use professional tools, e.g. Adobe Illustrator, to produce print ready portable document format (.pdf) books. These may differ significantly from editing stage documents, e.g. Microsoft Word (.docx format), used by edi-

Proceedings of the 2nd Workshop on NLP for Languages Using Arabic Script (AbjadNLP 2026), pages 208–217

tors and reviewers. This entails a full review of the page numbers of the BoBI.

Automating the process of mapping production and editing level BoBI across several adjustment iterations reduces a key pain point for editors, reviewers, and the whole publishing cycle.

Second, BoBI indexes indicate page numbers for their terms. For a (term, page) pair, the term might exist in several locations within the page. When the format changes, it is necessary to track these occurrences and their corresponding page numbers.

Third, due to the richness of Arabic morphology, terms might occur in a variety of forms that do not necessarily match at the lexical level. Morphological and semantic analysis helps ensure the accuracy of term matching and define relevant search scopes.

To address these challenges, we developed ArBoBIM to leverage existing natural language processing (NLP) tools and techniques. These include information extraction, page layout analysis, pattern matching, named entity recognition (NER), morphological analysis, and lexical and semantic similarity matching.

In summary, our contributions are:

1. ArBoBIM, automation of the extraction of BoBIs from Arabic documents.
2. Robust alignment of parallel BoBIs to address production and editing versions discrepancies.
3. A dataset of BoBI terms enriched with contextual and topic metadata.

The results presented in Section 6 feature a .94 F-1 score across several book types and open the door for deployment of ArBoBIM to automate BoBI tasks. Limited and initial user studies report better performance than LLMs with Arabic capacities with significantly lower computation cost and no risk of publishing house IP leak.

The rest of this paper is structured as follows. Sections 2 and 3 motivate the work and review relevant literature. Section 4 presents the methodology, and Section 5 highlights the resulting dataset created with ArBoBIM, and the model trained on it. Section 6 discusses the results, followed by Section 7 analyzing the errors.

2 Motivation

As a publishing house, BoBI generation, maintenance and review is laborious and consumes valuable expert time and effort. Typically, BoBI creation requires 4-6 work days from one expert and 2-3 days of review from a second expert. BoBI page numbers require tracking term occurrences across different versions with major changes between editing and production versions. The process is time consuming and prone to errors. This motivates publishing houses to invest in building ArBoBIM and automate BoBI extraction and review tasks.

3 Related Work

To the best of our knowledge, this is the first work to tackle the extraction and mapping of existing BoBI from Arabic documents. Rather, most of the literature focuses on the problem of direct automatic BoBI generation, with few attempts applied to Arabic books. For example, TMG-BoBI utilizes text mining and the Text-to-Matrix Generator to automate BoBI creation (Koutropoulou and Gallopoulos, 2019). It employs Automated Keyword Extraction (AKE) to identify relevant terms, including unigrams, bigrams, and trigrams, and integrates Part-of-Speech (PoS) tagging to categorize words by their linguistic roles, optimizing index readability with minimal user intervention.

Keyword extraction plays a crucial role in enhancing the efficiency and accuracy of back-of-the-book index creation by automating the identification of relevant terms and concepts within the text. An unsupervised framework for keyword extraction (Mao et al., 2020) focuses on candidate keyword selection and word scoring. They enhance keyword extraction by incorporating word co-occurrence and semantic relationships, resulting in improvements over baseline methods.

Keyword extraction requires additional relevance and presentation criteria to become efficient for term discovery. Hierarchical relations between terms (Li et al., 2020) improves results. BoBI review research work used syntactic and semantic similarity features to retrieve relevant page number (Christina and Oktaviyani, 2017), and used Naive-Bayes classification to identify BoBI index terms within the book (Christina and Ronaldo, 2020).

Arabic language has a grammar structure that *allows noun sentences*. It has a *rich morphology* which makes lexical matching less efficient for term identification. It *lacks capital case letters* and thus requires sophisticated named entity recognizers. Finally, it has a relatively *denser semantic similarity distribution* which requires term match dis-

ambiguation. Thus, direct application of keyword extraction techniques to Arabic documents is not enough for BoBI tasks.

The KpST system (Sahmoudi and Lachkar, 2016) leverages a suffix tree data structure for the extraction of Arabic key-phrases. It employs linguistic patterns and an adapted C-value method to extract relevant key-phrases. KpST also considers morphological features to improve key-phrase extraction for Arabic documents.

4 Methodology

This section describes the data processing, page mapping, and index entry mapping methodology we used to develop ArBoBIM. It is briefly explained in figure 1.

4.1 Data Preprocessing

To transform the BoBI to a structured index map, we first need to extract it as raw text from books.

Our source materials are books provided by *The Arab Center for Research and Policy Studies*, covering a range of topics including politics, sociology, and history, in editing format (.docx). The documents are initially parsed to extract the paragraphs, their page numbers, split positions on the pages, and indentation.

ArBoBIM parses the BoBI, which is typically located at the back of the book, as raw text to generate lists of index terms mapped to their claimed page numbers in the BoBI.

ArBoBIM considers the extraction of several features including multiple points of entry (MPE) terms, terms with hierarchical structures, and terms with style variations in phrases with specific patterns.

MPE signify the utilization of terms with similar meanings but differing terminology, such as "المملكة المتحدة ينظر بريطانيا" (United kingdom aka Britain). ArBoBIM detects and stores MPEs separately for future reference. We differentiate between two MPE types:

- MPEs that are treated equally and without differentiation in the sense that they share the same page numbers and are referenced under a single entry;
- MPEs that are separated and each appear in its own designated entry with its own page numbers.

Hierarchical structure involves a main entry with several sub-entries organized and listed beneath it. The structure signifies relations such as *isTypeOf*, *isSubtopicOf*, and *isPartOf* relations. The following example illustrates the United Nations with the Security council as a sub-entry.

الأمم المتحدة

-- مجلس الأمن

Finally, styled terms exist in some books to respect specific named entity styles, such as inverting person names as in the following example. The name "ناصر أبو رحمة" (Nasser Abu Rahma) is indexed as "أبو رحمة، ناصر" (Abu Rahma, Nasser) such that the last name precedes the first. When ArBoBIM parses the BoBI, it normalizes such entries.

4.2 Page Mapping Across BoBI Versions

Typically, the BoBI page numbers in the published version of a given book are manually specified. This happens after the application of final formatting steps and the addition of final style and brading elements. Parsing that BoBI provides a list of index terms, each mapped to a list of corresponding pages.

To locate these terms in an editing version of the same book, usually provided in Word format, we must establish a mapping between the pages across both versions. Content matching provides an initial alignment in case the final version was available in textual digital format. However, these versions are often not available for legacy books as they undergo several non-trackable iterations in the printing house.

Comparison between production (P) and editing (E) versions shows that differences in page numbers stem from aspects such as font size, line spacing, title styling, margins, and inclusion of graphics. Consequently, the mapping from production to editing is nearly linear.

To compute this mapping, ArBoBIM extracts the table of contents from E, whose page numbers also correspond to the production version for the same reasons mentioned above. Next, ArBoBIM traverses the content of the book, identifying titles and extracting their respective page numbers in version E. Now that we have the table of contents with page references for titles in P and E versions, ArBoBIM employs linear interpolation to map a given production page to its corresponding editing page(s). To account for potential misalignment and ensure a more accurate mapping, it applies a slid-

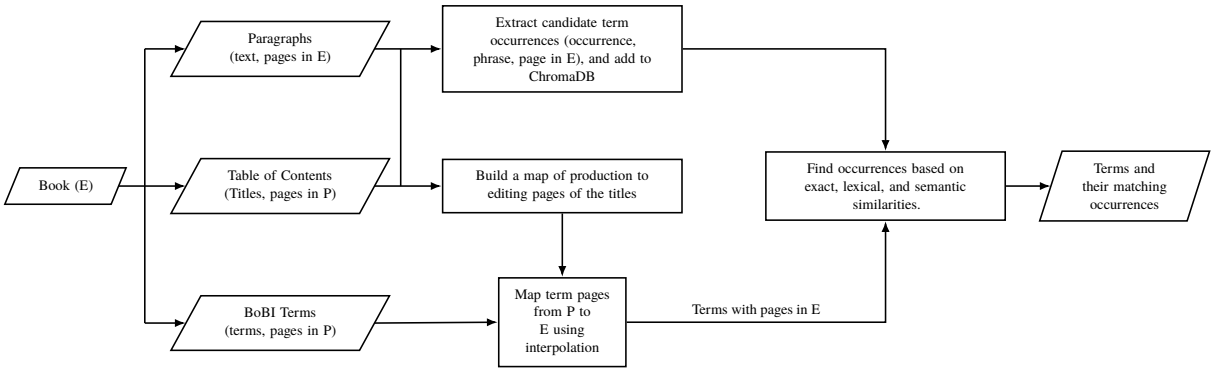

Figure 1: Pipeline for book with a table of contents with page references.

ing window of u pages in both directions, meaning that if we expect the mapping of page p to be page q, we consider the window starting from page $q-u$ and ending at $q+u$, while clamping to stay inside the book page bounds. In our case, u is chosen as 1. Note that for each page in P, ArBoBIM uses the closest pair of titles – one preceding and one following the page to be mapped – as reference points in the linear interpolation for better accuracy. The complete process is illustrated in Algorithm 1.

In cases where the book does not contain a table of contents, or contains a table of contents without page numbers, the page numbers extracted by parsing the BoBI are used instead. We apply a search on the entire document, similar to the one performed on a single window, as explained later, to extract the top candidate occurrences of each term in the BoBI. From there, each term t has a set of n pages where it occurs in P, denoted as $P_p = \{p_i \mid i \leq n\}$, as well as a set of m pages in E, denoted as $P_e = \{q_j \mid j \leq m\}$. We create a new set $S = \{(p_i, q_j, w_{ij}) \mid i \leq n, j \leq m, p_i \in P_p, q_j \in P_e\}$, with w_{ij} being the weight of this pair of pages. This weight decreases when n increases, and increases with the confidence of the occurrence in page q_j being correct. A Random Sample Consensus (RANSAC) is applied on S to deduce the most important page mappings, specifically those with acceptable variation between p_i and q_j are kept, and used to estimate the parameters (a, b) of a linear regression model, predicting for each page in P its corresponding page in E. A weighted error is minimized, where each pair (p_i, q_j) contributes proportionally to w_{ij}.

The dispersion W of this model is then estimated, to be used as a search window for each editing version page predicted in the next steps. More formally, we define the following:

- $R(p_i) = \{e \in \mathbb{N}, |ap_i + b - e| \leq W\}$,
- $S' = \{(p_i, e_k, w'_{ik}) \mid p_i \in P_p,\ e_k \in R(p_i)\}$,
- $w'_{ik} = \sum_j w_{ij}\ \exp\Big(-\frac{(e_k - q_j)^2}{2\sigma^2}\Big)$.

Our objective is then to find a set $M_p \subset S'$ of pairs (p_i, e_k, w'_{ik}) satisfying the following conditions:

- Each p_i appears at most once in M_p.
- If the elements of M_p are arranged in increasing order of p_i, the values of e_k are non-decreasing.
- The sum of w'_{ik} is maximized.

Such a result can be achieved by using Viterbi's algorithm, with forced monotonicity on the available paths. These initial page mappings are considered to have the same weight of 1 and are used to re-estimate the regression parameters (so, they are used for non-weighted linear regression). With the new regression parameters, we rebuild S' and re-apply Viterbi's algorithm. The resulting set M_p is then used for page mappings instead of the values deduced from the table of contents and its titles, with a sliding window of size $u = 2$.

4.3 Mapping Terms to Their Occurrences

ArBoBIM traverses the whole book in its editing version and extracts its paragraphs with their corresponding details, including their page numbers and table of contents titles (section, subsections and other headings), as well as the positions they are split at on the editing pages.

For each index term $\langle t, (\ell_1, \ell_2, \ldots)\rangle$ extracted from P, ArBoBIM maps the page P locations $(\ell_1, \ell_2, \ldots)$ to corresponding locations in E.

It consequently identifies the list of paragraphs $\langle pr_1^1, pr_2^1, \ldots, pr_1^2, pr_2^2, \ldots\rangle$ belonging to these

Algorithm 1: Index Page Mapping from Production to Editing Version Using Table of Contents

Input : Table of Contents (TOC) with production pages,
Book text with editing pages,
Index terms with production pages.

Output : Index terms mapped to editing pages.

Step 1: Extract Table of Contents (TOC);
Store TOC as list $(title, p_{\text{prod}})$;
Step 2: Map Titles to Editing Pages;
for *each title in TOC* **do**
    Locate first occurrence in book text;
    Store mapping $(title, p_{\text{edit}})$;
end
Step 3: Map Index Terms;
for *each term in index* **do**
    Retrieve production page p_{prod};
    Identify surrounding titles $(t_{\text{low}}, t_{\text{high}})$;
    Compute editing page using linear interpolation:

$$p_{\text{edit}} = p_{\text{low}}^{\text{edit}} + \frac{(p_{\text{prod}} - p_{\text{low}}^{\text{prod}})(p_{\text{high}}^{\text{edit}} - p_{\text{low}}^{\text{edit}})}{(p_{\text{high}}^{\text{prod}} - p_{\text{low}}^{\text{prod}})}$$

end
Return Index terms mapped to editing pages;

pages. These now constitute the search scope for term t in E.

ArBoBIM initially searches for exact matches of the term t within the search scope. If it finds an exact match, it records its position and returns the occurrence. Otherwise, ArBoBIM looks for phrases having lexical similarity above a specified threshold θ with term t. It uses morphological analysis (Obeid et al., 2020) to obtain the lemmas of the words in t. Then, it uses the Levenshtein distance (Levenshtein, 1966) to compute a morpho-lexical similarity ratio between the index term t and phrases from each paragraph.

Finally, if no exact and no morpho-lexical matches are found, ArBoBIM resorts to semantic similarity (Slimani, 2013). The different similarities are considered in sequential order, instead of employing them simultaneously to prioritize precision and computational efficiency. All thresholds used were empirically tuned.

Since index terms are naturally structured as noun phrases, ArBoBIM defines the pool of candidates within the paragraphs as all possible noun phrases in the search scope. ArBoBIM uses Part-of-Speech (PoS) tags (Obeid et al., 2020) and Named Entity Recognition (NER) (Jarrar et al., 2022) to predict noun phrases. PoS tags specify the role of a word in a sentence such as a noun, verb, adverb, or particle. They are key to identify noun phrases in the search scope. An example follows.

استخدمت	لجنة	تمثل	الأمم	المتحدة
verb	noun	verb	noun	adj

NER identifies typed entities within a text, such as names of people, organizations, locations, and dates. In the previous example, NER returns organization (ORG) for the phrase 'الأمم المتحدة' (United Nations).

Using sequences of PoS tags, ArBoBIM defines patterns that capture noun phrases. For example, a noun phrase starts with a noun, may include an adjective, cannot include a verb, and may not end with a preposition. These rules work well for limiting the number of candidate phrases to improve computational efficiency for similarity calculations. Several alternatives for PoS tagging exist (Qatar Computing Research Institute, 2025; Obeid et al., 2020) and may differ in abilities to detect the scope of the noun phrases.

ArBoBIM stores the extracted noun phrases, their metadata and the paragraph identifier, in a vector database (Chroma Team, 2026) using embedding from the "distiluse-base-multilingual-cased-v1" model (Reimers and Gurevych, 2019). Subsequently, if needed, it may query the vector database for top semantically similar entities.

ArBoBIM calculates the morpho-lexical similarity ratio of the index term with each candidate, and returns an aggregate score. For an index term t and a candidate occurrence c, the similarity score $S(t, c)$ is defined as:

$$S(t, c) = \alpha \cdot S_{\text{m-lex}}(t, c) + (1 - \alpha) \cdot S_{\text{sem}}(t, c)$$

where:

- $S_{\text{m-lex}}(t, c)$ is the morpho-lexical similarity,
- $S_{\text{sem}}(t, c)$ is the semantic similarity,
- $\alpha \in [0, 1]$ is a parameter balancing the morpho-lexical and semantic contributions. We set α to 0.3 by empirical tuning.

ArBoBIM selects the candidate with the maximum overall similarity score if and only if it ex-

ceeds a specific threshold which we empirically tuned at 0.45. For example, the top candidates for the index term "العلاقات الدولية" (international relations) follow.

- إجماع دولي (international consensus)
- القانون الدولي (international law)
- الصحف الدولية (international newspapers)
- الدول الأجنبية (foreign countries)
- العلاقات على المستوى الدولي (relations on international level)
- الضغوط الدولية (international pressures)
- الإرهاب (terrorism)

"العلاقات على المستوى الدولي" scores the highest with 0.67 morpho-lexical and 0.92 semantic similarities.

In case the index term has multiple points of entry (MPE) which share the same list of pages, ArBoBIM uses the measures above to decide on the top candidates relative to both the index term and its other points of entries.

5 Dataset and Training

We introduce ArBobIMap, a dataset created by running ArBoBIM on 510 non-fiction books from *The Arab Center for Research and Policy Studies*, mainly covering Economics, History, Civics, Religion, Politics, Social Sciences and Anthropology, Philosophy, Arts, Law, Linguistics, and Biographies. For each book, its title, topic, sections, and paragraphs are saved.

The dataset contains both terms that are indexed in books, and terms that are not indexed. Specifically, for each term the following information is present:

- The term as it appears inthe BoBI.
- The lemma, PoS sequence, and NER tags of the term.
- The books where this term is found in, and if it is indexed or not.
- The occurrences of each term in each book. This includes the page in the editing version, and production version (if possible), as well as its context, what paragraph it is found in, and the confidence in this occurrence being of that term.
- For indexed terms, the hierarchical structure is preserved by tracking its parent term, as well as alternative terms.

After inspection, we notice that about 1 in 10 terms are indexed. Using the term features present in the dataset, a model is trained to predict for a given (term, book) pair whether the term should be added to the back-of-the-book index of this book or not. We only highlight our most successful experiment in this section, which is a 4-layer neural network, trained with focal loss, as the dataset is unbalanced. After training, a threshold is chosen, where terms with scores above it are considered to be indexed terms (present in the back-of-the-book index) and terms below it as non-indexed terms (not present in the back-of-the-book index).

The following (term, book) features are the inputs of the model.

- The TF-IDF of the term in the book. Specifically, we break down the book into documents, each consisting of paragraphs with word count as close to 2k words as possible. We define TF as the number of times the term occurs in the book, and DF the number of documents it appears in. Then, we consider the TF-IDF score as $\frac{TF}{T} \times \log(\frac{N}{DF})$, where T is the number of terms in the book, and N is the number of documents.
- We define the NER score of a term in a similar fashion as the TF-IDF score, but considering the NER tag of the term as the unit. For example, a term with NER tag PERS (representing a person) would have a ner tag frequency equal to the number of times PERS occurs in all the documents, and its document frequency is the number of documents PERS occurs in. Terms that are not named entities have this score set to 0. We note that for this score, we consider all the documents in our knowledge base, unlike the previous score that relied on the documents of the current book only.
- In addition to the NER score, we also factor in a NER vector representation of a term. Our current NER model can give us 21 possible tags. Each tag is represented by a dimension, where the value of that dimension represents the probability of the term having this tag.
- Finally, we factor in the semantic similarity between the term and the book title, the term and the book topic, and the term and three titles. The three titles chosen are the

Book	Exact			Exact + Morpho-Lexical			ArBoBIM		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
1	1.0000	0.6189	0.7646	1.0000	0.6255	0.7696	0.9854	0.9512	0.9680
2	1.0000	0.7313	0.8448	0.9995	0.7606	0.8638	0.9834	0.9229	0.9522
3	0.9588	0.6107	0.7461	0.9599	0.6275	0.7589	0.9283	0.7880	0.8524
4	1.0000	0.8375	0.9116	1.0000	0.8843	0.9386	0.9867	0.9523	0.9692
5	1.0000	0.8000	0.8889	1.0000	0.8809	0.9367	0.9603	0.9256	0.9426
Average	**0.9918**	**0.7201**	**0.8344**	**0.9919**	**0.7556**	**0.8578**	**0.9689**	**0.9080**	**0.9375**

Table 1: Performance results for five books using Exact, Exact+Morpho-Lexical, and ArBoBIM Approaches

ones that show the top 3 semantic similarity values out of all the titles present in the table of contents. We note that the model used in this case for semantic similarity is "Omartificial-Intelligence-Space/Arabic-Triplet-Matryoshka-V2" (Nacar et al., 2025).

6 Results

We implemented ArBoBIM and experimented with several books. Table 1 reports the results across five books chosen across different topics and compare three approaches to evaluate the contribution of each technique. The exact matching within the scope serves as our baseline. Then we measure the effect of the morpho-lexical similarity metric. Finally, we incorporate the effect of the semantic similarity metric.

When searching for an occurrence of a term t in page p, we consider finding an occurrence in the range of p to give us a true positive. A false negative occurs if no candidate occurrences in the page range is considered to be an occurrence of t. A false positive happens when we misidentify an occurrence o as an occurrence of t, when it should not be.

In total, we analyzed 2471 index terms, with an average of 3.5 page references per index term.

The exact matching approach was able to precisely map the index terms to their contextual occurrences with an average precision of 0.9918 however it featured lower recall, with an average of 0.7201.

Incorporating the morpho-lexical similarity measure on top of the exact matching approach improved the recall to 0.7556 without significantly affecting the precision.

The reported results correspond to a threshold of 0.9 for lexical similarity, as we have observed that using a threshold lower than that significantly increases the number of false positives leading to a lower precision.

ArBoBIM incorporating semantic similarity significantly improves the recall across all books, with an average recall of 0.9080. This is because the semantic similarity measure is able to capture meaning-preserving variations of index terms that would otherwise be missed. However, this came at the cost of a decrease in precision to 0.9689 on average. The threshold used for $S(t, c)$ in this case is 0.45, with $\alpha = 0.3$, as mentioned in Section 4.

As for the prediction model, it is trained on 3.7M (term, book) pairs. The reported results are based on around 600k pairs in comparison, from full books that are not included in the training set. Specifically, for a book B from our dataset, it only appears in one of the two sets described above. We note that two books might share a common term, but be of different sets, but the (term, book) features would be unique for each of the two rows. Our confusion matrix with a threshold of 0.66 are highlighted in table 2. The variation in precision/recall is recorded in figure 2, as well as the precision-recall curve in figure 3. The recorded precision, recall, and F1-scores for the chosen threshold are 0.97, 0.85, and 0.9, respectively.

Finally, we note that strong baselines against LLMs are not reported in this paper. Current reported results (Bartmess and Combs, 2025) on indexing English documents still show much weaker performance than our approach. We note that under limited testing of a few handpicked terms and page references (that contain non-exact occurrences of the terms) GPT-OSS-20B (OpenAI, 2025), given an input of the term, and the list

Table 2: Confusion Matrix

	Predicted	
Actual	Indexed	Non-indexed
Indexed	57,065	9,994
Non-indexed	1,962	553,463

of paragraphs in the mapped pages of this term, would often output the term or its phrase in places where ArBoBIM managed to find the occurrence. While this is mostly as accurate as ArBoBIM, it is much more expensive to run and still relies on ArBoBIM mapping the pages for the model, as in most cases, LLM context windows may not be able to deal with the entire document in one go.

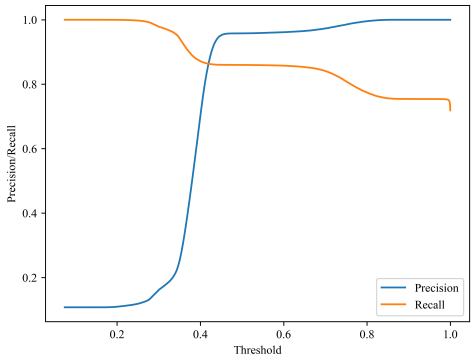

Figure 2: Precision/Recall as a Function of Threshold.

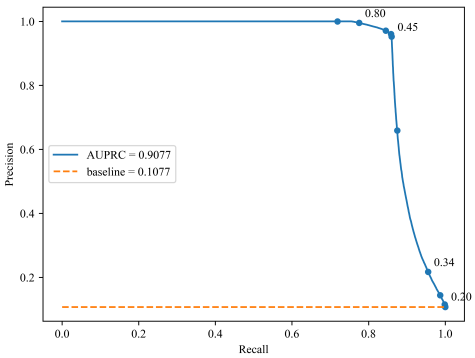

Figure 3: Precision-Recall Curve.

7 Error Analysis

7.1 Page Mappings

While our page mappings worked with most books, it is worth noting that in about 5% of the books processed by ArBoBIM, multiple term pages were wrongly mapped, resulting in re-runs with larger sliding window sizes. This can result in mixing up which production page occurrence matches which editing page occurrence found, especially when the pages are close.

This error is mainly the result of sparse tables of contents being present (as in these cases mentioned, the books had tables of contents with page numbers, not requiring the second method to be used), making the almost linear mapping fail on larger windows.

As for the mapping based on the pages in the BoBI, in about 92% of the cases, the editing page estimated from the production page is at most two pages off the actual one, and in about 75% of the cases, it is off by at most one page. This is tested fetching books with numbered tables of contents. After finding the mappings of the production pages to editing pages using the BoBI alone, and creating a set of (title, production page, editing page) elements, interpolation and extrapolation are used to approximate the editing page of the title, given its production page. We also note that some sections are not indexed, such as the BoBI itself, references, and depending on the book, the introduction. These titles are still present in the table of contents and are factored in the values reported, but do not affect the BoBI mappings in most cases. Excluding these titles, our values rise to respectively 96% and 79%.

7.2 Noun Phrase Extraction

Since we are mainly using CaMeL tools for PoS tagging, using the MLE disambiguator, which does not consider context, two different occurrences of the same word (e.g. عمل) can mean two different things depending on context (the verb to work, or the noun work). This can result in considering words that are not nouns as noun phrases, and vice versa. While this did not occur regularly, at best it can increase computations by adding unneeded candidate occurrences to our pools, and at worse can reduce our recall when filtering phrases before querying.

7.3 Mapping Terms to Occurrences

Precision can heavily be impacted by terms that have the same stem, but different lemmas. Such an example are the terms arts (الفنون), and artist (الفنان), where lexical and semantic similarities are both high. In other cases, the terms might be two different events under the same name, but different dates, or two different people sharing the same first name only.

7.4 Model Results

The current results of the model currently assume that our we are able to identify for each term its exact occurrences in the text. In practice, the precision and recall would decrease due to the accumulation of the extraction errors.

8 Conclusion

In this paper, we have presented a method to extract the back-of-the-book index of Arabic documents by transforming it from raw text to a structured index map to locate the occurrences of index terms in their context. This is important to automate navigation and review processes for BoBIs and to build resources for BoBI creation. ArBoBIM demonstrated excellent results and we identified opportunities for further improvements that we will seek shortly.

ArBoBIM is currently piloted with a small team of our editors and reviewers. Initial tests reveal that it performs better than LLMs with Arabic capacities without jeopardizing IP and copyrights.

The processed documents have allowed us to build the ARBoBIMap dataset of books and terms to be made available, as well as train a model capable of identifying which terms are worth adding to the back-of-the-book index. The dataset can be used to train new models to automate various back-of-the-book index related tasks for a given book.

9 Limitations

Although our proposed method achieves excellent results in automating back-of-the-book index mapping, several limitations need further improvement. First, the semantic similarity computations between numerous noun phrase candidates and index terms are computationally expensive, even when the vectors are indexed, limiting the method's efficiency for books with a larger number of index terms with no exact or lexical matches. Typically, a book with $100-200$ pages, containing $400-800$ terms takes around 5-10 minutes from start to finish.

Second, production-to-editing page mapping assumes linearity and relatively consistent pagination changes between the two versions, which may not always hold in cases of complex structural edits. Increasing the size of the sliding window may account for these differences, but would also increase the search scope. Additionally, all the books tested on are of the same publisher, using the same formats and consistent production and editing differences.

Moreover, our current approach faces challenges in cases where similar index terms occur on the same page, such as distinguishing multiple persons sharing the same surname. Thus, context-aware methods should be explored. Lastly, ArBoBIM may require context from external sources when the index term refers to a concept that is not explicitly mentioned in the text. This is where the data collected with initial runs of ArBoBIM would shine.

We would like to note that the scope of this paper covers Arabic only, as the available data and our knowledge of morphologically rich languages is limited to it. Our evaluations can be extended by replacing CaMel Tools, Wojood NER, and embedding models with others that are specialized with the target language.

Finally, our model still shows an overlap between indexed and non-indexed terms, which drops its recall drastically. It also does not consider the history of the term regarding its presence in the back-of-the-book indices, contextual embeddings, or other important features, which can potentially improve its results.

Ethics Statement

The data (see section 5) was collected and used with the appropriate approvals of the intellectual property owners. All results are reported following best academic standards and practices.

References

Elizabeth Bartmess and Michele Combs. 2025. Llm-generated book indexes: can they replace professionally created indexes? *The Indexer*, 43:327–348.

S Christina and D Ronaldo. 2020. Identify the relevant pages of book to be indexed using naive bayes classification method. *IOP Conference Series: Materials Science and Engineering*, 722(1):012043.

Sherly Christina and Enny Dwi Oktaviyani. 2017. Identifying the relevant page numbers that referred by the back-of-book index using syntactic similarity and semantic similarity. In *2017 Second International Conference on Informatics and Computing (ICIC)*, pages 1–6.

Chroma Team. 2026. Chroma: Open-source search and retrieval database for ai applications. https://github.com/chroma-core/chroma. Version 1.4.1 (released 2026-01-14), accessed 2026-02-06.

Mustafa Jarrar, Mohammed Khalilia, and Sana Ghanem. 2022. Wojood: Nested arabic named entity corpus and recognition using bert. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2022)*, Marseille, France.

Theoni Koutropoulou and Efstratios Gallopoulos. 2019. Tmg-bobi: Generating back-of-the-book indexes with the text-to-matrix-generator. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–8.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

Ning Li, Meng Tian, and Shuqi Lv. 2020. Extracting hierarchical relations between the back-of-the-book index terms. In *Chinese Lexical Semantics*, pages 433–443, Cham. Springer International Publishing.

Xiangke Mao, Shaobin Huang, Rongsheng Li, and Linshan Shen. 2020. Automatic keywords extraction based on co-occurrence and semantic relationships between words. *IEEE Access*, 8:117528–117538.

Omer Nacar, Anis Koubaa, Serry Sibaee, Yasser Al-Habashi, Adel Ammar, and Wadii Boulila. 2025. Gate: General arabic text embedding for enhanced semantic textual similarity with matryoshka representation learning and hybrid loss training. *arXiv preprint arXiv:2505.24581*.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. CAMeL tools: An open source python toolkit for Arabic natural language processing. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France. European Language Resources Association.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Qatar Computing Research Institute. 2025. Farasa Constituency Parser.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Issam Sahmoudi and Abdelmonaime Lachkar. 2016. Towards a linguistic patterns for arabic keyphrases extraction. In *2016 International Conference on Information Technology for Organizations Development (IT4OD)*, pages 1–6.

Thabet Slimani. 2013. Description and evaluation of semantic similarity measures approaches. *International Journal of Computer Applications*, 80(10):25–33.