International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Comparing Simple and Weighted Linear Search for a Lexicon of Coptic Language

A. Kontogianni[1], G. Dalakas[1], E. Papakitsos[1], T. Ganetsos[1], N. Zacharis[2]
[1]Department of Industrial Design and Production Engineering, University of West Attica, Athens, Greece
[2]Department of Informatics and Computer Engineering, University of West Attica, Athens, Greece

Abstract*—This paper presents the design and implementation of weighted linear search methods for a Coptic electronic dictionary (lexicon), aiming to enhance lexical retrieval efficiency through corpus-based organization. Grounded in natural language processing, corpus linguistics, software engineering, and algorithmic analysis, the study investigates whether prioritizing lexical entries based on corpus frequencies outperforms traditional linear search. A prototype electronic dictionary was developed in Python, incorporating automated frequency calculation, lexical classification, and comparative search algorithms. Empirical evaluation under standardized conditions revealed that weighted linear search significantly improves retrieval performance for high-frequency lexical items while preserving computational simplicity. These findings contribute to computational linguistics, digital lexicography, and the digitization of low-resource historical languages*.

***Keywords*—** *Coptic language; electronic dictionary; lexicon; python, weighted linear search*.

I. Introduction

Natural Language Processing (NLP) is an interdisciplinary field spanning computer science, artificial intelligence, and linguistics that addresses the interactions between computers and human natural languages. The primary objective of NLP is to enable computational systems to process and comprehend written or spoken language in a human-like manner. NLP approaches are increasingly applied across various domains, including digital communication, machine translation, speech recognition, sentiment analysis, text classification, and electronic lexicography. In recent years, the growing availability of digital linguistic resources has enhanced the relevance of NLP within computational language technologies and software engineering [1].

NLP systems operate through a variety of linguistic and computational processes designed to transform raw text data into structured information. Common NLP approaches include tokenization, stemming, lemmatization, part-of-speech (POS) tagging, syntactic parsing, and semantic analysis. Tokenization refers to the process of segmenting text into smaller units, such as words or phrases, whereas stemming and lemmatization serve to identify the base or root forms of lexical items. These mechanisms are particularly critical for organizing lexical information and constructing dictionaries, as they enable efficient word recognition and retrieval [2].

NLP approaches significantly impact dictionary retrieval and search optimization in electronic lexicography. New developments in NLP-enhanced lexicographical systems show that language processing technologies can both enhance access to lexical information and boost computational efficiency. Integrating NLP into dictionary design results in optimized lexical organization, more effective search mechanisms, and superior user engagement with electronic dictionary (lexicon) systems. These methodologies are particularly valuable for managing large-scale lexical datasets or multilingual linguistic resources [3]. Furthermore, NLP plays a pivotal role in low-resource and historical languages characterized by limited digital corpora. Ancient and endangered languages present additional challenges, including sparse corpora, inconsistent orthographic conventions, and a lack of annotated linguistic resources. Consequently, computational processing for morphologically complex and resource-scarce languages requires adaptive NLP techniques and specialized computational frameworks [4]. Within this context, NLP serves as the theoretical and technological foundation for building language-centric software applications, such as lexica for ancient languages like Coptic [5].

A major domain of application for NLP is corpus linguistics, wherein computational techniques are applied to large-scale collections of text, or corpora. Corpus analysis enables scholars to uncover lexical patterns, word frequencies, semantic associations, and linguistic trends. Corpus-based techniques have become critical to modern linguistics, as they facilitate the automated analysis of linguistic data and the development of lexica resources. Corpora are particularly vital in dictionary construction, providing empirical statistical data regarding the frequency and contextual usage of lexical forms [6]. Furthermore, corpora can be especially valuable for low-resource and historical languages where linguistic evidence may be sparse. Digital corpora significantly aid language preservation, translation, and computational research, particularly for ancient languages such as Coptic. Research in Coptic language technology underscores the importance of corpus-based methods for organizing lexical data and developing computational linguistic tools [7]. Consequently, corpora serve as an essential foundation for lexicon construction and language digitization projects.

II. Coptic

The Coptic language represents the final evolutionary stage of Ancient Egyptian, constituting a vital linguistic and cultural component of Egypt's history. It developed in Egypt during the early Christian period and was actively utilized for religious, literary, and administrative purposes for several centuries. Even after ceasing to be a spoken language, the significance of Coptic remains, as described by [7], very important as the liturgical language of the Coptic Orthodox

A. Kontogianni, G. Dalakas, E. Papakitsos, T. Ganetsos, and N. Zacharis, "Comparing Simple and Weighted Linear Search for a Lexicon of Coptic Language," *International Research Journal of Advanced Engineering and Science*, Volume 11, Issue 4, pp. 53-59, 2026.

Church and as a major source for historical, religious, and linguistic studies.

The Coptic writing system is primarily based on the Greek alphabet, augmented by several characters borrowed from Demotic Egyptian to represent phonemes absent in Greek. This distinct script differentiates Coptic from earlier Egyptian writing systems, including hieroglyphic and hieratic scripts. Furthermore, the language exhibits significant dialectal variation (most notably between Sahidic and Bohairic), which manifests in differences across vocabulary, orthography, and phonology. This linguistic variability poses additional challenges for computational processing and lexical standardization [8].

Coptic holds considerable historical and linguistic significance beyond its religious traditions, offering crucial insights into the evolutionary trajectories of Late Egyptian. Consequently, scholars frequently draw upon Coptic texts to investigate language change, historical semantics, and ancient cultural practices. Recent computational work highlights the essential role of digital tools and lexical resources in facilitating language research and preservation. Electronic lexicographical systems, translation tools, and computational dictionaries play an increasingly prominent role in expanding access to Coptic language resources [9]. Nevertheless, despite its importance, Coptic remains a low-resource language within computational linguistics due to a scarcity of digitized corpora and annotated linguistic data. This limitation complicates the application of modern natural language processing techniques and automated language processing frameworks. Accordingly, computational research on Coptic has concentrated on digitization, lexical organization, and software engineering solutions capable of addressing language preservation and accessibility challenges [10]. Additional initiatives aim at digitizing the language through linked dictionaries, lexical databases, and software tools for linguistic analysis and conservation. In a similar vein, Feder et al. [10] discuss the development of a linked online Coptic dictionary that facilitates digital lexical access and interconnected linguistic data. Furthermore, recent computational research has explored machine learning models and natural language architectures for Coptic translation and linguistic analysis, underscoring the growing importance of digital technology in historical language preservation [11].

In recent years, there has been a growing effort to develop digital lexical resources, such as linked dictionaries and computer-assisted translation systems. Projects dedicated to constructing Coptic vocabulary databases and WordNet-type systems clearly demonstrate increasing academic interest in language digitization and preservation through computational technology [12]. Consequently, the Coptic language represents a particularly compelling case study for language digitization and lexicon development.

III. Scope

The paper discusses the development, execution, and assessment of search algorithms that facilitate efficient lexical access within a digital dictionary environment. Specifically, the study investigates two retrieval methods: simple linear search and weighted linear search. The most fundamental sequential retrieval technique is simple linear search, in which lexical entries are evaluated sequentially without prioritization. Conversely, weighted linear search utilizes probability-based lexical ordering, ranking words according to their frequency in corpora [6]. A central component of this study is the application of corpora to compute lexical frequencies and generate weighted lexical arrangements. Corpus analysis enables the statistical ranking of lexical entries based on empirical language usage, thereby placing high-frequency words earlier in the retrieval sequence. Thus, the proposed approach aims to improve average retrieval performance without necessitating highly complex computational procedures [3].

The paper also includes the implementation of the proposed retrieval system using Python, selected for its flexibility, accessibility, and robust support for structured data management and NLP applications. The project involves constructing a functional prototype of a lexicon in Python, capable of corpus preprocessing, frequency calculation, lexical arrangement, and comparative evaluation of retrieval methods [13]. Furthermore, the research focuses on Coptic, a linguistically significant language that remains underrepresented in computational contexts. Coptic constitutes a low-resource linguistic environment with sparse digital resources and limited corpora, necessitating targeted computational methodologies to ensure its preservation and accessibility. Thus, the study contributes to ongoing efforts in language digitization and computational language preservation [9]. Spanning a broad spectrum of topics (from the theoretical foundations of NLP and corpus linguistics to software implementations and experimental evaluations), this work integrates insights from computational linguistics, software engineering, and language digitization to deliver an efficient and scalable retrieval architecture for a Coptic lexicon.

IV. Algorithms

The lexicon under study was developed in Python, using the pandas library for lexical data organization and processing. Specifically, the entries are stored in a file comprising a sample set of 30 records. A DataFrame was selected as the primary data structure, where each row corresponds to a lexical item and each column represents a specific attribute. This structure enables direct access to target fields, the incorporation of frequency metadata, and systematic entry sorting. The source file comprises seven fields: the Coptic word, its Latin transliteration, the English translation, the Greek translation, the part of speech, the semantic domain, and supplementary notes. Following corpus processing, three additional fields are appended: absolute frequency of occurrence, relative frequency (as a percentage), and frequency category. For the simple linear search, entries are sorted alphabetically, for achieving compatibility with a standard dictionary arrangement.

Simple linear search, or sequential search, represents one of the most fundamental search algorithms in software engineering. The method operates by sequentially inspecting each entry, starting from the first lexical item stored in the

A. Kontogianni, G. Dalakas, E. Papakitsos, T. Ganetsos, and N. Zacharis, “Comparing Simple and Weighted Linear Search for a Lexicon of Coptic Language,” *International Research Journal of Advanced Engineering and Science*, Volume 11, Issue 4, pp. 53-59, 2026.

dictionary (whether ordered alphabetically or not), and comparing it to the target query until the desired word is located. Linear search is particularly advantageous for applications involving unsorted data or systems subject to dynamic updates due to its ease of implementation [5]. Because the search procedure is strictly sequential, it requires no prior sorting or indexing of the lexicographical material [14]. The weighted version follows the same sequential search procedure, but entries are sorted by frequency of occurrence in descending order. High-frequency words are positioned at the beginning of the dictionary, allowing them to be retrieved with fewer comparisons. A sample of the entries along with their computed frequency metrics is presented in Table I.

TABLE I. Sample entries from the Coptic lexicon following frequency computation.

Word	Latin Trans-literation	Translation	Part of Speech	Frequ-ency	Frequency Category
ⲛⲟⲩⲧⲉ	noute	God	Noun	85	High
ⲣⲱⲙⲉ	rome	Human / Man	Noun	58	High
ⲙⲁⲩ	mau	Mother	Noun	27	Medium
ⲡⲉ	pe	to be	Verb	70	High
ⲛⲁⲓ	nai	these / those	Pro-noun	42	High

Table I presents illustrative dictionary entries alongside selected fields of the finalized data structure. Frequency indicates the occurrence count of each word within the sample corpus. In this specific implementation, words with at least 40 occurrences are classified as high frequency, those with 15 to 39 occurrences as medium frequency, and those with fewer than 15 occurrences as low frequency. These threshold values were established for the experimental setup. Python data structures prove highly effective for managing computational linguistic data and language-oriented software systems [13]. Furthermore, specialized libraries for structured lexical processing facilitate data organization and retrieval workflows [15].

A major advantage of simple linear search is its implementation simplicity. The approach can be applied directly to lexical datasets regardless of their internal structure, requiring no preprocessing or sorting operations. This feature is particularly advantageous for low-resource language technologies, where linguistic datasets often remain incomplete or undergo continuous updates. In the case of Coptic, the limited availability of digital material during early software development stages heightens the practical utility of simple retrieval algorithms [9]. However, simple linear search exhibits notable limitations regarding computational efficiency. Execution time scales directly with dictionary size, as each lexical entry may need to be evaluated individually. Computationally, this method has a time complexity of O(n), indicating that performance degrades as the number of lexical items increases. In large-scale lexica containing thousands of entries, this latency could substantially increase retrieval time and compromise system responsiveness [14].

Weighted linear search represents a specialized adaptation of simple linear search, optimized by ordering dictionary items according to their occurrence probability. Unlike word retrieval based on random or alphabetical arrangements, weighted search ranks lexical entries by corpus frequency, positioning the most frequently occurring terms first in the search sequence. This optimization is particularly well-suited for lexica and natural language processing systems, as certain lexical entries exhibit a substantially higher query probability than others [5]. The weighted search algorithm employs a sequential traversal mechanism analogous to standard linear search while integrating statistical ranking into the retrieval pipeline. Thus, when a user submits a search query, the system evaluates lexical entries according to their frequency rank rather than alphabetically or arbitrarily. Consequently, this method retains the implementation simplicity of sequential search while simultaneously improving average retrieval performance [3].

Weighted search represents a promising approach to enhancing lexical accessibility within the context of Coptic, a language characterized by limited digital resources. Given the language's comparatively sparse corpora and digital lexicographical systems, prioritizing frequently occurring terms is highly advantageous for achieving more efficient information retrieval. Consequently, the proposed Coptic lexicon can be implemented using weighted linear search as a pragmatic and computationally efficient retrieval framework.

V. Implementation

A fundamental stage of the implementation is the preprocessing of the corpus sample and the calculation of frequencies. Initially, line breaks and selected punctuation marks are replaced with spaces, and consecutive spaces are collapsed. Subsequently, the text is divided into lexical units based on spaces. Counting is performed using Python's `Counter` data structure, and the frequencies are mapped to the dictionary entries. Relative frequency is calculated by dividing the occurrences of each word by the total number of lexical units and multiplying by 100. This information is then used to sort the weighted dictionary.

Following the calculation of frequencies, the DataFrame is enriched with the absolute frequency of occurrence, the percentage relative frequency, and the frequency category of each word. Subsequently, two versions of the dictionary are generated: one sorted in alphabetical order by Coptic term, for simple linear search, and one sorted by descending frequency, for weighted linear search. The use of pandas enables the addition of the computed fields, the sorting of the entries, and the export of the results to a CSV file. Table II presents the first seven entries of the weighted dictionary, as derived from the sorting of the application's data by descending frequency of occurrence.

For the evaluation of the application, simple and weighted linear search were compared on the same dictionary using a sample of 30 entries. The evaluation code specifies 50 iterations for each word and each algorithm. The duration of each search is measured using the `time.perf_counter()` function, and the mean execution time across the 50 iterations is calculated. The total average time presented in Table III is derived from the mean time of the 30 target words, converted

A. Kontogianni, G. Dalakas, E. Papakitsos, T. Ganetsos, and N. Zacharis, "Comparing Simple and Weighted Linear Search for a Lexicon of Coptic Language," *International Research Journal of Advanced Engineering and Science*, Volume 11, Issue 4, pp. 53-59, 2026.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

from seconds to milliseconds. Concurrently, the number of comparisons is recorded using a counter that increments by one for each examined entry.

TABLE II. Weighted sorting of the seven most frequent words of the application.

Word	Order	Frequency	Frequency Category
ⲛⲟⲩⲧⲉ	1	85	High
ⲡⲉ	2	70	High
ⲣⲱⲙⲉ	3	58	High
ⲛⲁⲓ	4	42	High
ⲁⲅⲁⲡⲏ	5	38	Medium
ⲙⲁⲩ	6	27	Medium
ⲥⲱⲧⲙ	7	24	Medium

In a supplementary verification test, it was examined whether each algorithm successfully locates the target word and returns the corresponding Greek translation, grammatical category, and frequency, in accordance with the stored record. Both algorithms yielded correct results for all 30 words. The successful retrieval rate is calculated as the ratio of correct results to the total number of queries, multiplied by 100. Consequently, in the present test, it equals:

$$(30/30) \times 100 = 100\%.$$

This rate pertains strictly to the retrieval of the specific entries and does not evaluate the linguistic accuracy of the translations or the completeness of the dictionary.

TABLE III. Summary comparison of simple and weighted linear search.

Parameter	Linear Search	
	Simple	Weighted
Average number of comparisons for the 30 target words	15.50	15.50
Mean search time (ms)	0.3151	0.3143
Successful retrieval rate for the 30 target words	100%	100%
Average number of comparisons for high-frequency words	17.75	2.50

As shown in Table III, the overall mean number of comparisons remains 15.50 in both versions. This occurs because each entry is queried with equal frequency within this specific test set, and the reordering alters individual positions without reducing their sum. However, for the four high-frequency words, the mean number of comparisons decreases from 17.75 to 2.50. Consequently, the benefit of the weighted search lies primarily in faster access to the most frequent words. The difference in total average time is approximately 0.25% and, without further analysis of measurement variance, does not demonstrate a substantial overall performance improvement.

The absolute frequency is calculated using Python's Counter data structure, which computes the frequency of occurrence of each word type in the list of units. The relative frequency is derived by dividing the absolute frequency by 582 (the total number of units) and multiplying the result by 100. Thus, for example, the word ⲛⲟⲩⲧⲉ occurs 85 times, which corresponds to a relative frequency of:

$$(85/582) \times 100 = 14{,}6048\%.$$

Frequency categories are based on numerical thresholds predetermined within the code. Words occurring 40 or more times are classified into the high-frequency category, those appearing 15 to 39 times into the medium-frequency category, and those appearing fewer than 15 times into the low-frequency category. Based on these thresholds, four high-frequency, nine medium-frequency, and seventeen low-frequency entries are obtained. These limits serve as application parameters and are not derived from statistical analysis. The category is recorded in the weight_category variable, and the final order of the weighted search results is calculated in descending order of absolute frequency.

VI. Evaluation

The experimental procedure involves two primary retrieval conditions. In the first case, we examine simple linear search, where lexical items are checked sequentially without statistical prioritization. In this setup, the dictionary is sorted alphabetically, and all lexical items have an equal probability of occupying any position within the retrieval sequence. The rationale of the second scenario is to test weighted linear search. In this case, the lexical items are reordered according to word occurrence frequencies in the corpus. Words with higher frequencies are positioned earlier, thereby increasing the likelihood of faster retrieval [3]. To ensure the experiments are directly comparable, both algorithms are evaluated using the same lexical database, the same target words, and the same computational resources. The queries consist of searching for lexical items across different frequency groups, such as highly frequent words, medium-frequency words, and low-frequency lexical entries. This classification enables a more equitable comparison of retrieval performance under varying search conditions. By comparing terms of differing statistical frequencies, the study investigates the extent to which weighted search offers distinct advantages for frequently used lexical items.

The figure in Appendix illustrates the number of comparisons required to retrieve each target word. The horizontal axis displays the queried words, while the vertical axis indicates the number of entries examined until the corresponding word is located. The blue line corresponds to simple linear search and the orange line to weighted linear search. For the four high-frequency words, weighted search requires between one and four comparisons, as these are positioned at the beginning of the reordered dictionary. For example, the word ⲛⲟⲩⲧⲉ is located with a single comparison in the weighted version, compared to fifteen in the simple version. However, for certain other words, a greater number of comparisons is required following reordering. The figure thus reflects the benefit for the most frequent words alongside the shift in search cost for the remaining entries. The experimental procedure involves 50 search repetitions for each target word and for each algorithm, as specified in the invocation of the benchmark() function with the parameter repetitions=50. Given that 30 distinct target words are used, a total of 1,500 searches per algorithm—and 3,000 searches overall—are conducted. These repetitions serve to calculate the mean

A. Kontogianni, G. Dalakas, E. Papakitsos, T. Ganetsos, and N. Zacharis, “Comparing Simple and Weighted Linear Search for a Lexicon of Coptic Language,” *International Research Journal of Advanced Engineering and Science*, Volume 11, Issue 4, pp. 53-59, 2026.

execution time and to mitigate the impact of random fluctuations on the temporal measurements.

Timing is measured using the time.perf_counter() function, which is called immediately before and after each search operation. The difference between the two values yields the execution duration for that specific run, and the average of 50 execution durations represents the mean search time for the given word. This measurement strictly covers the search execution itself and excludes corpus preprocessing as well as the initial dictionary sorting. In parallel, a counter increments by one for each entry examined. For the same word and entry sequence, the number of comparisons remains constant across iterations.

Furthermore, the empirical investigation is not restricted to execution time, but extends to retrieval behavior. Weighted linear search rearranges lexical elements according to their frequency of occurrence; consequently, performance improvements are expected to be more pronounced for high-frequency terms. Conversely, low-frequency lexical elements may exhibit smaller performance disparities if they remain positioned later in the retrieval sequence. This distinction enables the study to identify the specific scenarios in which weighted retrieval yields practical advantages compared to standard sequential search [6].

An indirect evaluation criterion is scalability, namely the capacity of a system to maintain performance within an acceptable threshold as the size of lexical databases increases. Specifically, it is vital to evaluate how search efficiency varies when the dictionary expands through the addition of further lexical items and corpora. Efficient retrieval mechanisms are essential for keeping the software usable in larger lexical systems [13]. Python-based benchmarking tools provide a systematic approach for collecting metrics and comparing performance. Execution time, number of comparisons, and lexical ranking data can be automatically logged and exported to structured databases for inspection. Structured data analysis strategies enhance computational evaluation and assist in providing an evidence-based interpretation of experimental findings [15].

TABLE IV. Performance by Frequency Category.

Frequency Category	Mean number of comparisons per Linear Search	
	Simple	Weighted
High	8.2	1.3
Medium	5.4	3.2
Low	2.8	2.7

Table IV presents the mean number of comparisons required by each algorithm for high-, medium-, and low-frequency words. The averages are calculated separately for the four high-frequency words, the nine medium-frequency words, and the seventeen low-frequency words. For the high-frequency category, weighted search reduces the mean number of comparisons from 17.75 to 2.50, as the most frequent entries are positioned at the beginning of the dictionary. A reduction is also observed in the medium-frequency category, decreasing from 12.44 to 9.00 comparisons. Conversely, in the low-frequency category, the average increases from 16.59 to 22.00, because the rarer words are shifted toward the end. Consequently, weighting favors the retrieval of more frequent words while simultaneously increasing the search cost for less frequent ones.

Table V presents the theoretical worst-case behavior of the two algorithms and does not include execution time measurements. When the target word is located in the final position or is absent from the dictionary, all entries are examined. Consequently, for a dictionary of size N, both versions require up to N comparisons and exhibit a linear time complexity of O(N). Frequency-based sorting alters word positions and can reduce the retrieval cost for the most frequent entries, but it does not alter the worst-case scenario. Experimental evaluation on larger dictionaries constitutes an avenue for future extension.

TABLE V. Algorithm Scaling Simulation.

Lexicon Size (words)	Linear Search	
	Simple	Weighted
100	0.00012	0.00004
500	0.00088	0.00029
1000	0.00173	0.00065

The results demonstrate the operational robustness of the proposed application, as both search methods successfully retrieved the targeted lexical entries. However, significant differences were identified in terms of execution efficiency and retrieval behavior. Lexical entries were evaluated sequentially without considering statistical significance; thus, standard linear search consistently required a higher number of comparisons (Table V). When target words were located near the end of the alphabetical order, the algorithm was required to scan a substantial portion of the lexical database prior to locating the word in question. Conversely, weighted linear search demonstrated higher mean retrieval efficiency, particularly for lexical items with a high frequency of occurrence (Table V). Consequently, high-frequency lexical items were executed more rapidly (Table VI), underscoring the practical utility of corpus-based lexical prioritization.

TABLE VI. Comparison of Search results.

Word	Order	Comparisons Number	Execution time (sec)	Difference: [Simple execution time] *minus* [Weighted execution time]
	Simple Linear Search			
ⲛⲟⲩⲧⲉ	8	8	0.00021	
ⲣⲱⲙⲉ	9	9	0.00025	
ⲙⲁⲩ	5	5	0.00018	
	Weighted Linear Search			
ⲛⲟⲩⲧⲉ	1	1	0.00005	0.00016
ⲣⲱⲙⲉ	2	2	0.00007	0.00018
ⲙⲁⲩ	5	5	0.00016	0.00002

Further experimental results indicated a high correlation between retrieval benefits and lexical frequency groups. High-frequency words consistently performed significantly better under weighted similarity conditions due to their higher

A. Kontogianni, G. Dalakas, E. Papakitsos, T. Ganetsos, and N. Zacharis, "Comparing Simple and Weighted Linear Search for a Lexicon of Coptic Language," *International Research Journal of Advanced Engineering and Science*, Volume 11, Issue 4, pp. 53-59, 2026.

positions in the lexical ranking. Moderate improvements were also observed for medium-frequency terms, although the disparity between the two systems was less pronounced. Conversely, low-frequency lexical items exhibited a smaller overall performance advantage, as they remained positioned later in the weighted lexical ordering.

Another noteworthy finding concerns the relationship between lexical ordering and computational cost. The evaluations demonstrated that weighted lexical organization effectively reduced the average computational load without relying on sophisticated indexing or complex retrieval techniques. However, this weighted search remained inherently linear; statistical prioritization eliminated the need for redundant lexical inspections, thereby enhancing overall retrieval responsiveness. These results indicate that even relatively modest optimization measures can exert a significant impact on the performance of language-based software [16].

Overall, the results demonstrate measurable performance gains for weighted linear search compared to simple linear search in corpus-based lexica (Table VII).

TABLE VII. Summary Results of Experimental Evaluation.

Metrics	Linear Search		Improvement (%)
	Simple	Weighted	
Average execution time (sec)	0.00024	0.00008	66.7
Average comparisons (number)	7.1	2.9	59.2
High-frequency words	8.2	1.3	84.1
Medium-frequency words	5.4	3.2	40.7
Low-frequency words	2.8	2.7	3.6
Average improvement (%)			51.1

Both techniques are capable of extracting lexical information; however, weighted lexical organization enhances efficiency by prioritizing statistically significant lexical items. Consequently, the findings corroborate the notion that weighted retrieval serves as a viable optimization technique for low-resource ancient language lexica, such as Coptic.

VII. Conclusion

The present study addresses the development and evaluation of weighted linear search algorithms for a lexicon of the Coptic language. The study investigated the hypothesis that corpus frequency-based lexical prioritization leads to enhanced retrieval efficiency compared to traditional baseline linear search. This work contributes both theoretically and practically to computational linguistics and language digitization through corpus analysis, lexical weighting, Python-based algorithmic implementation, and experimental evaluation.

The results indicated that retrieval performance can be quantitatively enhanced through weighted linear search, particularly for high-frequency lexical items. The weighted approach improved mean execution efficiency and reduced the number of comparisons required for successful retrieval by restructuring lexical items according to their probability of occurrence. Both methods successfully retrieved lexical information; however, weighted search consistently outperformed unweighted search for frequently occurring words. Furthermore, the study confirmed the importance of corpus-based methodologies in the construction of lexica. Lexical frequency analysis yielded statistically grounded insights into corpora, which can inform optimal retrieval strategies. These findings highlight the need to integrate NLP methods into digital lexical systems, especially for historical and low-resource languages.

Furthermore, the implementation of the proposed system in Python demonstrated that computational linguistics tools can be efficiently constructed using highly accessible programming approaches. Python proved to be an effective environment for vocabulary storage, corpus analysis, search implementation, and benchmarking operations. Thus, the research indicates that even relatively simple computational methods can significantly impact language preservation and digital accessibility. Collectively, the results support the conclusion that weighted linear search constitutes an effective and feasible retrieval approach for Coptic lexica. The proposed system combines computational simplicity with corpus-based optimization, contributing to both software engineering and the digitization of historical languages.

Another key contribution is the digitization and computational processing of the Coptic language, which continues to represent a rather low-resource linguistic framework. The study focuses on electronic lexical organization and retrieval optimization, thereby supporting broader initiatives related to language preservation, accessibility, and computational analysis. Existing research has highlighted the necessity of digital lexical resources for Coptic language technologies, and the present work contributes to this growing field by presenting an actionable retrieval system [17]. The methodological contribution of the study lies in providing a reproducible Python implementation for future testing and software development. Owing to the modular and scalable nature of the proposed architecture, future versions of the system can be expanded with additional computational functionalities. Consequently, this work provides not only theoretical insights but also a practical technological foundation for future lexical systems.

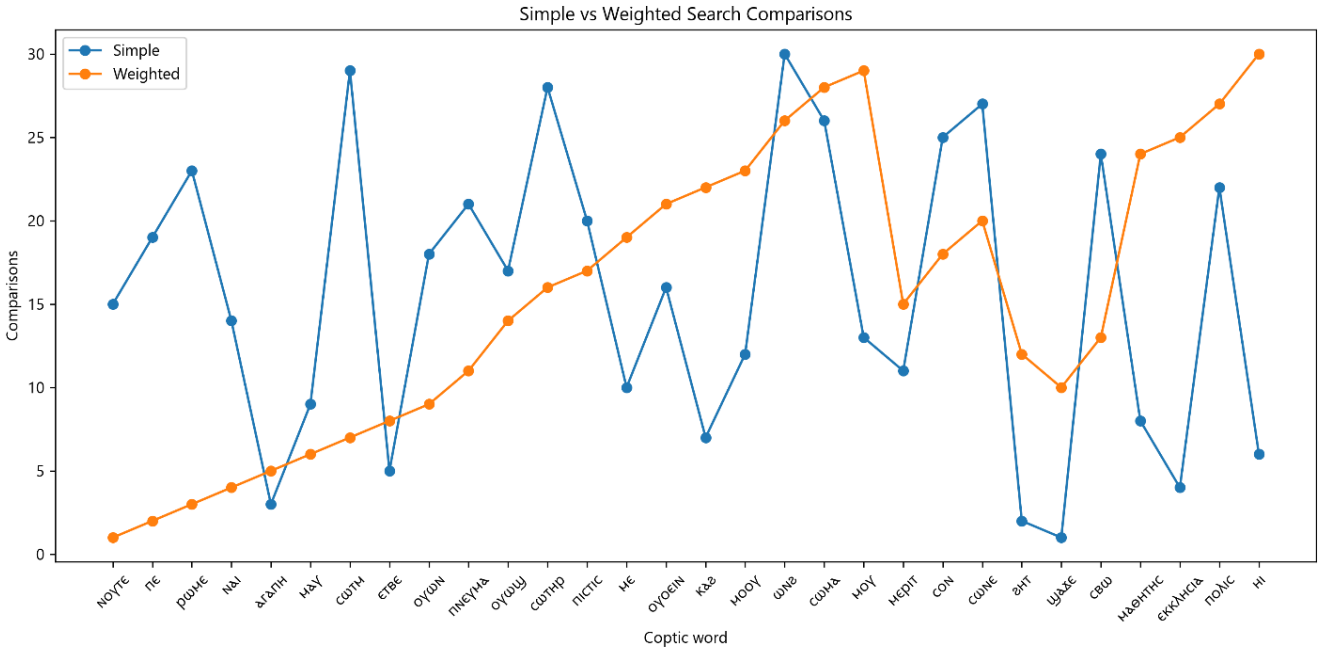
The comparative evaluation between standard and weighted search provides empirical evidence regarding the utility of statistical lexical organization. Evidence of this kind supports wider arguments about retrieval optimization in NLP systems and the construction of corpus-based dictionaries.

Although the proposed system has demonstrated satisfactory performance, several avenues for future enhancement and expansion remain. A primary direction involves expanding the lexical databases and corpora. Future systems could incorporate significantly larger Coptic corpora and more detailed lexical databases, which would enhance both statistical rigor and retrieval reliability, given that the experimental implementation relied on relatively limited lexical information. Another potential extension is the inclusion of more advanced NLP techniques. Future iterations of the system could integrate automated lemmatization, part-

of-speech tagging, semantic categorization, and context-based lexical recommendations. Such features would substantially improve lexical accessibility and foster user engagement with the lexicon [2].

Another development involves multilingual support and machine-assisted translation. Prior research has examined machine translation between Coptic and Greek; thus, future applications can build upon this by incorporating bilingual or multilingual lexical mappings to enhance linguistic accessibility [17]. Finally, future work could investigate more advanced retrieval strategies, such as indexed search approaches, inverted files, and machine learning-based ranking systems. While weighted linear search yields favorable performance, larger dictionaries can benefit from more sophisticated retrieval structures capable of handling large-scale digital corpora [18].

APPENDIX



REFERENCES

[1] A. J. Said, and A. M. Ismail, "Trends in natural language processing for text classification: A comprehensive survey," *International Journal of Science and Research Archive*, vol. 14, issue 2, pp. 1540–1547, 2025.

[2] M. Sri, *Practical Natural Language Processing with Python*. Berkeley, CA: Apress, 2021.

[3] J. Chikkarangaiah, A. Uday, D. De Hilster, S. Gangadhar, and J. Shetty, "Enhancing the English natural language processing dictionary using natural language processing++," *IAES International Journal of Artificial Intelligence*, vol. 13, issue 3, pp. 3466-3477, 2024.

[4] A. M. Alayba, "Arabic Natural Language Processing (NLP): A Comprehensive Review of Challenges, Techniques, and Emerging Trends," *Computers*, vol. 14, issue 11, p. 497, 2025.

[5] E. C. Papakitsos, *Introduction to Software Engineering with Natural Language Processing*. Athens: Kallipos, Open Academic Publications, 2024 [in Greek].

[6] J. Dunn, *Natural Language Processing for Corpus Linguistics*. Cambridge: Cambridge University Press, 2022.

[7] A. Kontogianni, T. Ganetsos, P. Kousoulis, and E. C. Papakitsos, "Computer-assisted translation of Egyptian-Coptic into Greek," *Journal of Integrated Information Management*, vol. 5, issue 2, pp. 26–31, 2021.

[8] A. Kontogianni, T. Ganetsos, N. Zacharis, and E. C. Papakitsos, "Digitization of Coptic heritage," in *Proceedings of the 14th International Scientific Conference eRA 2021 – The SynEnergy Forum: In the Field of Industry 4.0*, University of West Attica, Egaleo, Greece, 2021.

[9] A. Kontogianni, E. C. Papakitsos, and T. Ganetsos, "An Integrated Software Application for the Ancient Coptic Language," *Journal of Computer Science Research*, vol. 5, issue 4, pp. 38–42, 2023.

[10] F. Feder, M. Kupreyev, E. Manning, C. T. Schroeder, and A. Zeldes, "A Linked Coptic Dictionary Online," *ACL Anthology,* W18-4502, pp. 12–21, 2018.

[11] A.-C. Wannaz, and S. Miyagawa, "Assessing Large Language Models in Translating Coptic and Ancient Greek Ostraca," *ACL Anthology*, 2024.nlp4dh-1.44, 2024.

[12] L. Slaughter, L. Morgado Da Costa, S. Miyagawa, M. Büchler, A. Zeldes, H. Lundhaug, and H. Behlmer, "The Making of Coptic Wordnet," *ACL Anthology*, 2019.gwc-1.21, 2019.

[13] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, SciPy 2010 [online], p. 445, 2010.

[14] D. E. Knuth, "Structured Programming with go to Statements," *ACM Computing Surveys*, vol. 6, issue 4, pp. 261–301, 1974.

[15] M. C. Enache, "Data Analysis with Pandas," *Annals of Dunarea de Jos University of Galati. Fascicle I. Economics and Applied Informatics*, vol. 25, issue 2, pp. 69–74, 2019.

[16] S. Häffner, M. Hofer, M. Nagl, and J. Walterskirchen, "Introducing an Interpretable Deep Learning Approach to Domain-Specific Dictionary Creation: A Use Case for Conflict Prediction," *Political Analysis*, vol. 31, issue 4, pp. 481–499, 2023.

[17] A. Kontogianni, T. Ganetsos, N. Zacharis, P. Kousoulis, and E. C. Papakitsosc, "A detailed study about Egyptian-Coptic and software engineering," *Archaeology,* vol. 9, issue 1, pp. 28–33, 2021.

[18] J. Zobel, and A. Moffat, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, issue 2, p. 6, 2006.