Proposal of a lightweight, offline, full-text search engine for an mHealth app

Carla Teixeira Lopes
University of Porto / INESC TEC
Porto, Portugal
ctl@fe.up.pt

David Azevedo
University of Porto
Porto, Portugal
davidazevedo1996@gmail.com

João M. Monteiro
INESC TEC
Porto, Portugal
joaomiguelmonteiro@gmail.com

Abstract — **A patient's ability to recall and retrieve health information contributes to a better health management. HealthTalks was developed to address these issues by recording a summary of a medical appointment, uttered by the physician, and transcribing it. For each appointment, the user can also take free-text notes. Nowadays, search engines have become a ubiquitous part of everyone's life and are expected on most applications. Here, we describe the development of a search engine for HealthTalks. The app's characteristics demand a lightweight and offline engine, which requires a specific solution rather than an existing library or service. Our approach combines SQLite's Full-Text Search 4 module, which includes n-gram indexing, with traditional information retrieval techniques to rank the documents. We created a test collection with summaries of clinical appointments (our documents), information needs, search queries, and relevance assessments for an initial search engine evaluation. Using this test collection, we assessed performance using NDCG@10, the first rank position of a totally relevant result, and query latency. Results are promising, with an average NDCG of 0.97. The median rank position of the first relevant result varies between 1.9 and 1.95, depending on the use of 4-gram character tokenization, an aspect that did not significantly affect the results. We expect this work to be useful for future developments of full-text search engines in mobile environments.**

Keywords – mobile computing; lightweight search engine; health information.

I. INTRODUCTION

Search is a regular asset nowadays in several environments, and mobile apps are no exception, given their ubiquity [1]. HealthTalks [2] is a mHealth app that aims at increasing the patients' health condition by instigating in them a more active self-care. Using this app, patients can ask their physicians to dictate a summary of an appointment and permission to record it. The app will record this audio and automatically transcribe it to text in a favorable scenario. This transcription is later associated with more information about the medical concepts therein. These functionalities will allow patients to recall or retrieve information that can be helpful to a better understanding of medical information and better management of personal health information. Users may use the app for themselves or for someone they take care of.

Although not mandatory, in HealthTalks, medical appointments can therefore be associated with a recording and respective transcription. Moreover, the app relates medical appointments with the physician's name and specialty, the health establishment where the appointment happened, the date, and other parameters; those variables are meant to make an appointment easier to identify and locate. The user may also associate free-text notes, that is, notes without any constraint of format, to medical appointments. As the information available in the app grows over time, a search engine may be helpful to find information related to past appointments, especially regarding the free-text fields that the interface cannot display systematically.

HealthTalks' first search implementation was a simple query to the database with the CONTAINS operator. While this data retrieval implementation may be sufficient in situations where unstructured data is scarce, this type of data will grow with more frequent use of the app. Given this perspective, we decided to develop a full-text search engine in HealthTalks. Some of the benefits of a full-text search engine include retrieving results that do not explicitly contain the search terms but are related to the search query and ranking these results according to their relevance to the information need. Compared with a data retrieval approach, we also expect an increase in performance in searches done over big blocks of free text.

Given the sensitive nature of the data, HealthTalks stores all its data only on the device for privacy reasons. For the same reasons, the search engine works completely offline with a local specific index that exists in, and only in, the app data directory. The search engine's offline nature also minimizes data usage on mobile devices. Another important aspect is to keep the application as lightweight as possible to accommodate a larger spectrum of mobile devices. In addition, although the search engine was implemented for Android, to be compatible with future implementations of HealthTalks in other operating systems, our approach is independent of the operating system. Given the specific domain of the app, both documents and queries will probably contain medical terminology. We have, therefore, introduced medical-specific features to our search engine, namely, query expansion using specialized medical synonyms and medical acronyms interpreters. We have also considered abbreviated and full formats of dates and 4-gram character tokenization on the users' queries.

Section II presents a brief state of the art about using search tools in mobile environments. Afterward, we describe our requirements for the search engine in Section III and its development in Section IV. In Section V, we report the results

of a preliminary evaluation of the search engine. Finally, in Section VI, we conclude and present lines of future work.

II. Search tools for mobile environments

When it comes to open-source search engines, there are several options to choose from [3]. Given the offline requirement of our search engine, we only describe implementations that do not require a server-side module.

Lunr 1 is an open-source search engine. It is built in JavaScript, is small, and can be used either on the client or server-side of web applications. ElasticLunr2 is based on Lunr and extends it to provide Query-Time boosting and field search. Lucene3 is a Java written library. It features ranked search, proximity queries, wild card queries, and indexing. Due to its requirements, any information stored in a database by the app would have to be duplicated.

If the app uses a database, an alternative would be, if available, to use the full-text search capabilities of the database. These capabilities may change between different database systems, but we focus on this system given the popularity of SQLite in the mobile environment. SQLite is also the database used in HealthTalks.

SQLite has 3 non-obsolete full-text search modules: FTS3[4], FTS4[4], and FTS5[5]. These modules build an inverted index of terms from the inserted documents into a user-defined virtual table. FTS3/4 provide auxiliary functions such as *snippet*, *offsets*, and *matchinfo*. These functions are SQL scalar functions useful for full-text search. The first two allow the identification of queried terms in the retrieved documents. The *matchinfo* function provides metrics useful for relevance ranking. FTS5 has no *matchinfo* or offsets functions, and its snippet function is not as fully-featured as in FTS3/4. However, FTS5 has a built-in *bm25* ranking function and has an API that allows the creation of custom auxiliary functions.

The three FTS modules allow the use of both pre-built and custom tokenizers, including n-gram tokenizers that support substring matching instead of the usual token matching. N-gram indexing helps counteract misspellings [12] and searches for similar words to those given in the query [11]. For example, the words *indicator*, *indicative*, and *indication* share the common 7-gram *indicat*.

Regarding scientific literature, we found no articles describing the implementation of full-text search engines in the context of mobile applications.

III. Search Engine Requirements

Given the context of the HealthTalks, already described in Section I, our full-text search engine has to have the following characteristics:

- Offline – Work with a local specific index that exists in, and only in, the app data directory.
- Lightweight – Be as light as possible to be compatible with a more extensive range of mobile hardware devices.
- Interoperable – The implemented approach has to be compatible with operating systems other than Android.
- Medical-specific – Given the specificities of the terminology of the documents, the search engine should expand queries with medical synonyms and acronyms.
- Usable – The interface should be intuitive and straightforward for people with reduced technological skills.

IV. Search Engine development

Given that the search engine should be as light as possible, that HealthTalks already uses SQLite, and that SQLite works locally, we decided to explore the full-text search capabilities of this database system. This way, no other system is required. We chose the FTS4 module for the availability of the *matchinfo* function, useful for the ranking stage. A summary of our implementation can be seen in Fig. 1. The code is available on an online repository[6].

In our context, a document is the concatenation of all the columns we define in our virtual table. The different columns are the doctor's name, the location of the appointment, date, transcript, and associated notes. By default, the column containing the concatenation of all the columns has the same name as the virtual table. We query this column when we want to use the FTS4 index. The documents are user-specific and stored locally on their mobile device.

We chose the 'unicode61' tokenizer of FTS4, which integrates the removal of diacritics in the process. This tokenizer allowed us to handle regular diacritics in some non-English languages. Although occasionally, words may be distinguished by their accents, as users often enter queries without diacritics, we decided to equate all words to a form without diacritics. This process is applied to the documents when they enter the database.

A. Query processing

After receiving a query, we tokenize, process it, and perform a logical disjunction ('OR') search with every query term at the end of this stage. This way, we retrieve any document similar to the query's terms, regardless of term order. Query tokenization is done using the same tokenizer we used for documents. Query processing is divided into three stages: date processing, query expansion, and 4-gram pseudo stemming. We detail these stages in the following subsections.

1) Date Processing

An essential feature within the context of HealthTalks is the ability to handle queries with date and time [4–7] properly.

1 https://lunrjs.com/

2 http://elasticlunr.com/

3 https://lucene.apache.org/core/

4 https://www.sqlite.org/fts3.html

5 https://www.sqlite.org/fts5.html

6 https://github.com/david-azevedo/Android-FTS-Offline

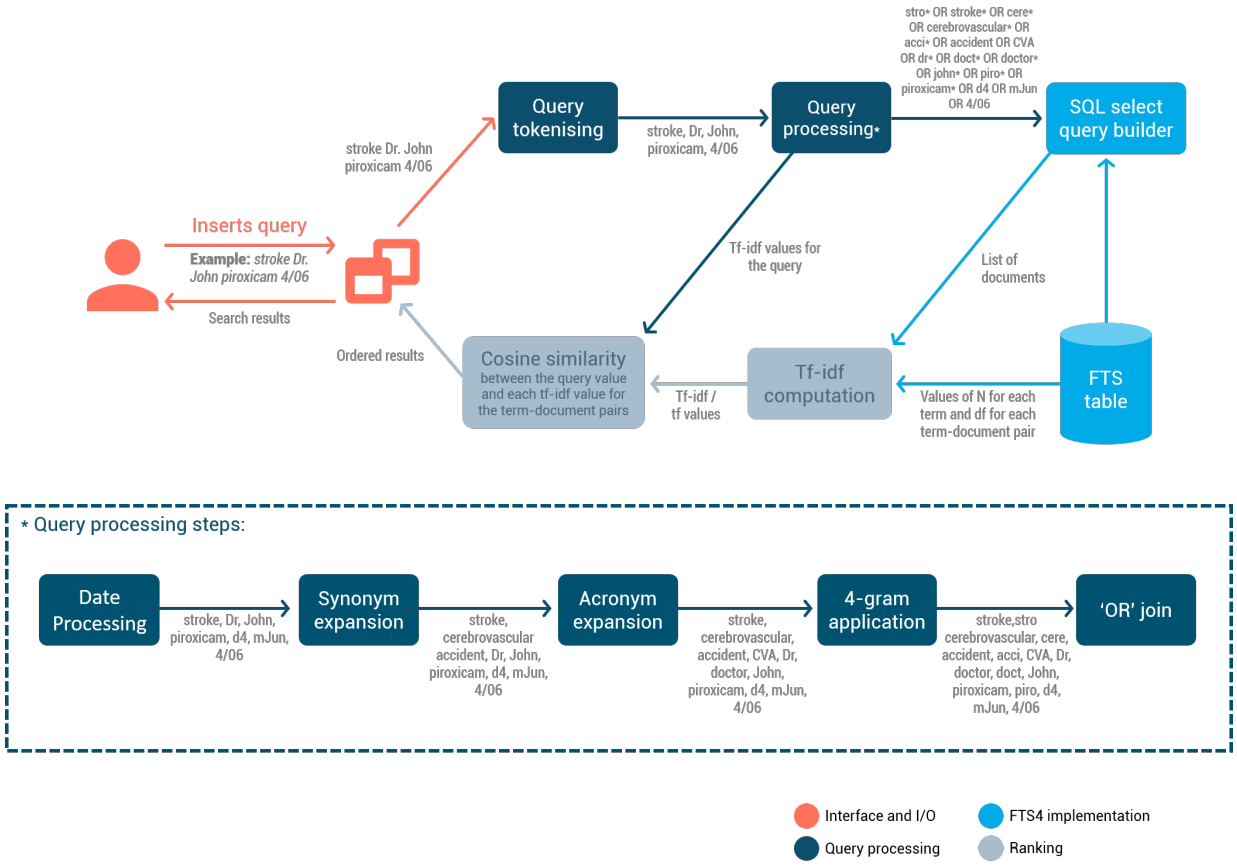

Figure 1. Search engine architecture

Given that appointments occur in specific periods, notes are created in different periods and, in this context, we are likely to have the date mentioned in the summary and notes.

Date-related data might be associated with different representations, such as "2022-01-01", "January 6, 2022" or "Jan. 6, 2022". Moreover, besides specific dates (e.g., a particular day of a year, a specific month of a year, or a particular year), temporal data may also include mentions of a single month or a weekday as in "… the appointment of June …" or "…last Monday…".

We devised a specific representation format for dates in documents and queries to accommodate different representation formats and commonplace descriptors of dates, such as "June". We have considered using an international standard such as ISO 8601 for this. Still, the need to contemplate parts of dates possibly associated with several periods (e.g., "June") pushed us away from this option.

We used regular expressions to detect dates, namely, the day, month, year, and day of the week, considering both abbreviated and full formats. We concatenate the first letter of the variable's name with the respective shortened value to represent the detected dates. For example, the date "1st of January 2022" or "01/01/2022" would result in the string "d1 mJan y2022 wSat". When new information is inserted into our database, we detect dates and store them in the format above. We do the same during query processing. With this, it is possible to match documents and queries having dates represented in different formats and to consider partial matches of a particular date, ranking the results by similarity.

2) Query expansion

Medical terms may be hard to retain for laypeople. Given the specific domain of the app, both documents and queries will probably contain medical terminology. We have, therefore, introduced medical-specific features to our search engine, namely, query expansion using specialized medical synonyms and medical acronyms interpreters.

We used the Consumer Health Vocabulary (CHV) of the Unified Medical Language System (UMLS) Metathesaurus [8] to build a set of pairs containing a given medical term and the corresponding preferred layperson term, including acronyms and abbreviations. When a user performs a query, the system first checks the synonym database to find a correspondence (either for a medical or layman's term). If a match occurs, we expand the query with the appropriate synonym. If the user introduces "stroke", we expand the query to include the medical term "cerebrovascular accident" and the "CVA" abbreviation.

3) N-gram pseudo stemming

Stemming reduces "inflectional forms and sometimes derivationally related forms of a word to a common base form" [9]. There are many stemming algorithms [10], with Porter's stemming algorithm being the most known and used.

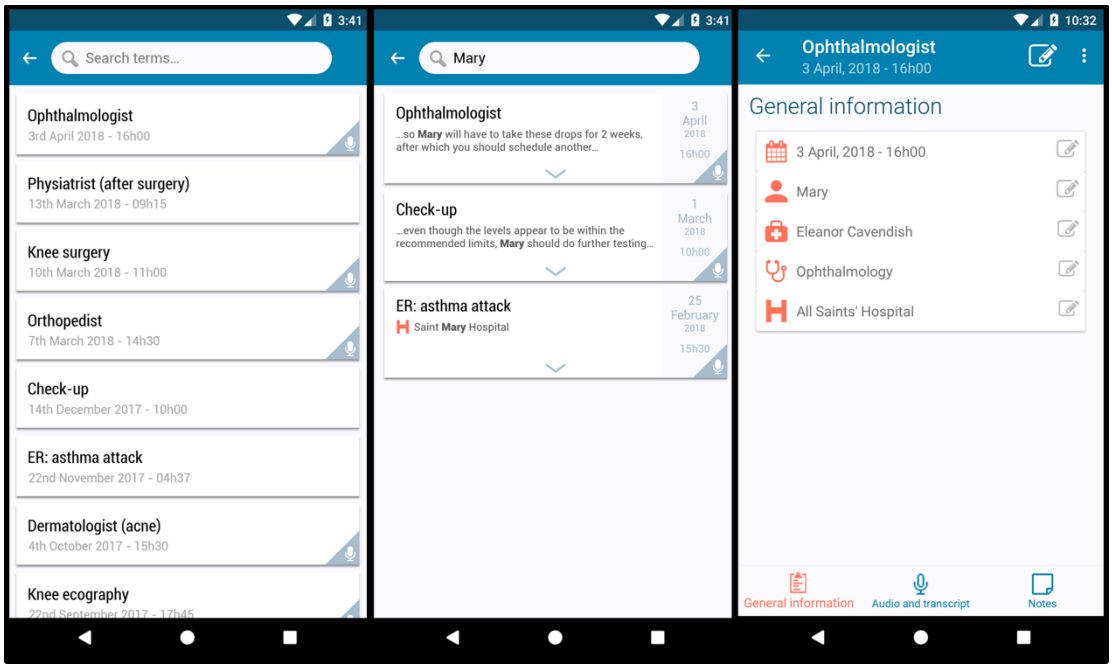

Figure 2. Mockups for the new search engine. From left to right: the user opens the list of appointments; then they search for the term ”Mary” and the list of appointments updates to only include appointments related to that search; and lastly the user clicks on the appointment they were looking for to see.

The downside of stemming algorithms is that they are language-specific; on the other hand, 4-gram tokenization has comparable performance to language-specific stemming algorithms in European languages, sometimes even surpassing them [11]. Based on these results, we implemented 4-gram character tokenization [12] on each token.

B. *Ranking*

Predicting situations where several results match the given query, we decided to rank the results using TF-IDF (Term Frequency - Inverse Document Frequency). The values are calculated for each term-document pair. For the inverse document frequency (IDF), we query the virtual table for the total number of documents (N) and the document frequency (DF) for each term. To compute the term frequency (TF), we use the *matchinfo* function of FTS4. This function returns a blob array with (1) the number of terms in the query, (2) the number of columns we have defined in the virtual table, and (3) the frequency of each query term in each document. This information allows us to calculate the TF-IDF values for all term-document pairs in our result. We store the results in arrays that contain the values of every term for each document. This data structure is then used to compute the cosine similarity [13] between each document and the query.

C. *User interface*

When developing a search interface for a solution that people with reduced technology skills might use, it is essential to use a familiar and straightforward interface. That was something we took into account when developing HealthTalks. Regarding search, we also aimed to make it seamless. Users are accustomed to more straightforward solutions such as those used in the major web search engines, which do not need any input besides the search terms to retrieve pertinent information. Hence, we decided to simplify the search interface since a more advanced search engine would permit less input from the user without compromising the search results. We only ask for the search terms; when the user writes them, we list the relevant results displayed in order of decreasing relevance. Fig. 2 shows the mockups for this user journey.

V. Preliminary Evaluation

To evaluate the usefulness of our search engine implementation, we created a mock database with 105 appointment summaries with date, physician, hospital, and transcription. These summaries always include a diagnosis or the results of an exam. Often, they included prescribed medication or recommended therapy.

Although this number of documents may seem relatively low, especially compared to the usual collections in information retrieval scenarios, it is probably more extensive than the number of appointments a regular user will have. Based on these summaries, we created seven information needs based on the work of Borlund [14], ensuring the realism and control of the experiment. These information needs are presented in Table I. We assessed the relevance of every appointment summary on a 0 (not relevant) to 2 (totally relevant) scale for each information need.

We then asked 13 volunteers to make queries, available in Table I, for each information need. Their ages ranged from 19 to 62 years old (average 34); 8 were female and four were male; 69% (n=9) had a Master’s degree, 2 had Bachelor’s degrees, one had completed up to the 12th grade, and another the 9th grade. They had no prior access to the database we created.

TABLE I. INFORMATION NEEDS AND QUERIES SEPARATED BY SEMICOLONS.

IN	Information Need	Queries
IN1	You have had myocardial problems for a few years. Last June, you went to an appointment at Cliria where the doctor prescribed benuron and suggested a specific treatment. Now you will have a cardiology appointment with another doctor, and you need to tell him the benuron dosage and the recommended treatment.	Benuron+dosage; benuron june 2018; Benuron dose by weight; benuron care; Consultation cairia cardiology benuron posology; Benuron dosage and recommendations myocardial problem; cliria myocardium; Benuron dosage; Cliria, benuron; appointment, cliria, benuron; Benuron cliria myocardium; Cliria benuron; benuron cardiomyopathy
IN2	Your name is Bruno. Last July, you had an appointment with Dr. Rita. You recall the appointment was on a Friday because, on the following day, you went on vacation with your family. At that appointment, the doctor explained to you, in some detail, the differences between neoplasm, tumor, and cancer. Now, out of curiosity, you would like to reread that explanation.	Neoplasm+tumor+cancer; Dr Rita July; Tumor vs Cancer Rita; Difference between neoplasm, tumor and cancer; Dr Rita July sixth appointment neoplasm tumor cancer; Consultation Bruno Friday differences between diseases; cancer Friday; Difference between neoplasm tumor and cancer; July, appointments, Dr. Rita; appointment, July, rita; appointment rita cancer; Rita tumor; Dr. Rita tumor, cancer
IN3	Last summer, you had an appointment with Dr. Odete Orvalho in her private office. She analyzed your blood test results and referred you to the neurosurgeon Albina Afonso due to a cyst in your brain. You need to find that appointment to show it to another doctor and get a second opinion.	Consultation+albina; Dr Odete Orvalho Albina Afonso; brain albina; Blood tests; appointment Albina Afonso neurology; Consultation dr Odete Orvalho cyst brain; brain analyses; Consultation Dr Odete Carvalho; Consultation, Dr Odete; appointment, odete; blood brain analysis; Odete Carvalho; Dr Odete orvalho
IN4	In June, you went with your son to an appointment with Dr. Roberto Pintassilgo at Hospital da Trofa. He was diagnosed with a disease whose name he cannot remember, only that it starts with "neuro". You want to know the name of that disease.	Appointment +pintassilgo; Dr Roberto Pintassilgo Hospital da Trofa; Neuro pinta; Appointment June, Dr. Roberto; Consultation July child pintassilgo neuro*; Appointment June dr Roberto pintassilgo Hospital Trofa; neuro Trofa; Appointment dr pintassilgo; Dr. Roberto, neuro; appointment, June, roberto; Roberto trofa neuro; Neuro Trofa; Dr Roberto, Trofa, neuro
IN5	Dr. Fernandes prescribed you amiodarone, and you forgot the dosage. You need to find that appointment to remember the dosage.	Amiodarone+fernandes; Dr Fernandes amiodarone; Amiodarone dosage; Amiodarone; Consultation Fernandes amiodarone posology; Dr Fernandes posology amiodarone; Fernandes; Amiodarone dosage; Consultation, Dr. Fernandes, amiodarone; appointment, fernandes, amiodarone; amiodarone Fernandes; Amiodarone Fernandes; Dr Fernando, amiodarone
IN6	You went to the hospital to see Dr. Pedro because of a bruise, and now you want to know what day that appointment was.	Date+ appointment +Pedro; Dr. Pedro bruise; Pedro date; Consultation with Dr. Pedro; Day appointment bruise pedro; Day appointment dr Pedro bruise; hospital bruise; Appointment Dr Pedro; Consultation, Dr. Peter, bruise; appointment, pedro; hospital bruise pedro; Bruise ecchymosis Pedro; Dr Pedro bruise
IN7	After having a CVA, you went to see Dr. Carolina Cardoso on August 30th for a check-up. Now you need to know what her recommendations were at that appointment.	Appointment +30th of August; CVA August 30th; Cardoso CVA; August 30; August 30 appointment recommendations; CVA recommendations dr Carolina Cardoso 08/30; CVA; Dr Carolina appointment; Appointment, Dr. Carolina; appointment, carolina; check-up Carolina cva; Carolina Cardoso August 30th; Cva, Dr Carolina Cardoso

We submitted the 91 queries to the search engine and retrieved, on average, 28.69 results per query.

To assess the performance of our search engine, we computed the Normalized Discounted Cumulative Gain (NDCG) [15], the first rank position of a totally relevant result with and without 4-gram character tokenization, and the query latency. Table II shows the results for these measures, aggregated for all the queries.

TABLE II. AGGREGATED VALUES FOR EVALUATION METRICS. MEDIAN FOR RANK POSITION AND AVERAGE FOR OTHER MEASURES.

NDCG@10	First rank position of a totally relevant result		Query latency (ms)
	with 4-gram	*without 4-gram*	
0.97	1.9	1.95	27.66

The average NDCG@10 is 0.97, close to the ideal 1; only six queries in 91 (6.6%) had NDCG@10 values lower than 0.90. Most of those were in cases where dates were used in non-abbreviated ways (e.g., “30th of August”). The 4-gram tokenization and TF-IDF ranking are probably causing this. In the previous example, they take “August” and match it with proper names such as “Augustine”, which, due to their rarity, have a higher IDF and appear on top of the results. This happens because, even though we normalize dates, we still treat them as regular tokens; a possible solution is to consider them exclusively as dates.

We noted the position of the relevant documents for each information need in the list of results for the query. The average rank position of the first totally relevant result for each query is 1.90, which is not the ideal 1.00. However, we still consider it an excellent value, mainly considering the average number of retrieved results. By removing the 4-gram module, that result is slightly worse, going up to 1.95. There were some specific cases in which removing it was beneficial, especially when the query was small and generic (such as “analysis brain”). On the other hand, cases where the user misspelled the medication’s name would not have yielded results without the 4-gram. A paired sample t-test showed no significant differences in using 4-gram (p-value = 0.05).

Regarding the performance of each module, TF-IDF computation is the most demanding at an average of 9.05 milliseconds in a total average of 27.66 milliseconds for the time interval between the arrival of the query and the generation of an answer, taking, as predicted, noticeably more time in longer queries. We believe each module we implemented should remain since the time they use is negligible.

VI. Conclusions and Future Work

We described the implementation of a full-text search engine in the context of a mobile app that should be as light as possible and work offline. This implementation used SQLite's Full-Text Search 4 module and traditional information retrieval techniques, some specialized in the medical domain.

As the search engine will work offline with each user's collection of documents, the collection size will be small. Yet, an automatic search mechanism is advantageous even with few documents, as manually scanning transcriptions and notes associated with dozens of documents is time-consuming.

We evaluated this search system with a straightforward experiment to understand its performance. Results show that we can still improve specific areas such as date processing for optimal performance.

We noticed a prevalence in searching for the word "date" when their information need involved a date; or "appointment" when the goal was to retrieve an appointment. Query formulation habits in online search engines probably influence this behavior. In future developments of the search system, we envision treating these special terms as indicators of what the searcher is looking for and perhaps highlight the desired information on the results page.

It would also be interesting to implement the BM25 ranking function available in FTS5 and compare it with the one implemented in this work. We found that 4-gram tokenization has not significantly affected the results. Therefore, we would like to compare its use with stemming and lemmatization approaches and evaluate their impact on the system's performance.

At the moment, HealthTalks does not suggest words while the user is typing, but in the future, we want to suggest words based on parameters such as recent searches and terms in the index.

This evaluation was probably more demanding than a real-world situation would be. We have used more than 100 documents, a number of documents that not every user will have. Moreover, the queries were formulated by people that did not have any prior knowledge of the appointments. This setup and results give us confidence in the system's good performance. Nevertheless, we plan to conduct an additional evaluation experiment, including real data and more users, to obtain further insights. The existence of real data will allow us, for example, to understand better what users are generally looking for and how many documents are likely to be indexed on average.

Acknowledgment

This work was developed under the project "NORTE-01-0145-FEDER-000016" (NanoSTIMA) that is financed by the North Portugal Regional Operational Programme (NORTE2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF). This work is also financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020.

References

[1] Pew Research Center, "Mobile Fact Sheet," Pew Research Center, Tech. Rep., 2021. [Online]. Available: http://www.pewinternet.org/fact-sheet/mobile/

[2] J. M. Monteiro and C. Teixeira Lopes, "HealthTalks - A Mobile App to Improve Health Communication and Personal Information Management," in *Proceedings of the 2018 Conference on Human Information Interaction&Retrieval - CHIIR '18*. New York, New York, USA: ACM Press, 2018, pp. 329–332.

[3] C. Middleton and R. Baeza-Yates, "A Comparison of Open Source Search Engines," 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.119.6955

[4] O. Alonso, J. Strotgen, R. Baeza-Yates, and M. Gertz, "Temporal information retrieval: Challenges and opportunities," in *WWW2011 Workshop on Linked Data on the Web, Hyderabad, India*, March 29, 2011, ser. CEUR Workshop Proceedings, C. Bizer, T. Heath, T. Berners- Lee, and M. Hausenblas, Eds., vol. 813. CEUR-WS.org, 2011, pp. 1–8. [Online]. Available: http://ceur-ws.org/Vol-707/TWAW2011-paper1.pdf

[5] H. Joho, A. Jatowt, and B. Roi, "A survey of temporal web search experience," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13 Companion. New York, NY, USA: Association for Computing Machinery, 2013, p. 1101–1108.

[6] N. Kanhabua, R. Blanco, and K. Nørvå̊g, "Temporal information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 9, no. 2, pp. 91–208, 2015.

[7] D. Lewandowski, "Date-restricted queries in web search engines," *Online Information Review*, vol. 28, no. 6, pp. 420–427, 2021/10/14 2004.

[8] National Library of Medicine, "CHV (Consumer Health Vocabulary) - Synopsis," https://www.nlm.nih.gov/research/umls/sourcereleasedocs/current/CHV/index.html, accessed: 2021-10-15.

[9] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to information retrieval*. Cambridge University Press, 2008. [Online]. Available: https://nlp.stanford.edu/IR-book/

[10] C. Moral, A. de Antonio, R. Imbert, and J. Ramírez, "A survey of stemming algorithms in information retrieval," *Information Research*, vol. 19, no. 1, 2014. [Online]. Available: http://InformationR.net/ir/ 19-1/paper605.html

[11] J. Mayfield and P. McNamee, "Single n-gram stemming," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 415–416.

[12] P. McNamee and J. Mayfield, "Character n-gram tokenization for european language text retrieval," *Information Retrieval*, vol. 7, no. 1, pp. 73–97, 2004.

[13] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, no. 2, pp. 6–es, jul 2006.

[14] P. Borlund, "The IIR evaluation model: a framework for evaluation of interactive information retrieval systems," *Information Research*, vol. 8, no. 3, 2003. [Online]. Available: http://InformationR.net/ir/19-1/paper605.html

[15] Y. Wang, L. Wang, Y. Li, D. He, and T.-Y. Liu, "A theoretical analysis of ndcg type ranking measures," in *Proceedings of the 26th Annual Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, S. Shalev-Shwartz and I. Steinwart, Eds., vol. 30. Princeton, NJ, USA: PMLR, 12–14 Jun 2013, pp. 25–54. [Online]. Available: https://proceedings.mlr.press/v30/Wang13.html