Configurable In-Database Similarity Search of Electronic Medical Records

Yuewen Wu[1,2], Yong Zhang[2](✉), and Jiacheng Wu[2]

[1] Princeton University, Princeton, NJ 08544, USA
[2] Tsinghua University, Beijing 100084, China
zhangyong05@tsinghua.edu.cn

Abstract. With the development of technology, Electronic Medical Records (EMRs) are widely used for medical analysis through methods such as similarity search. Typical EMRs contain attributes of different data types including string, enumeration and numeric data, and are commonly stored in a database. However, many EMR similarity search algorithms neither separate different data types nor conduct search directly in the database. In addition, for researchers and doctors who need similarity search but do not have strong programming background, a user-friendly interface is missing. Therefore, we design a tool "SIR" to solve the aforementioned problems. SIR can conduct configurable similarity search in high dimensions (within 0.0931 and 0.7824 s respectively using its basic and advanced version), can be embedded directly in the database, and has an intuitive interface.

Keywords: EMR · UDF · GUI · Similarity search · Database

1 Introduction

Real-World Evidence (RWE) is widely used by those who develop medical products or who study, deliver, or pay for health care [11]. With the development of technology, Electronic Medical Records (EMRs), one category of RWE, are frequently archived and subsequently extracted for analysis [2]. EMR is proven to be useful in many aspects. Besides making better judgements by comparing similar patients, researchers even manage to leverage similar EMRs to predict possible future medical concepts (e.g., disorders) [7]. The global electronic health records market size was valued at USD 26.8 billion in 2020 and is expected to witness a compound annual growth rate (CAGR) of 3.7% from 2021 to 2028 [10]. This expanding market shows both the potential and necessity of EMR research. One application of EMR is patient similarity search, which uses collected medical data to find a cohort of patients with similar attributes to any given patient [3]. Sometimes, patients share similarity in terms of a combination of different attributes rather than only one. With similarity search, multiple dimensions of an EMR can be taken into account and reflected as a straightforward number, allowing doctors and lab researchers to find the hidden connections between patients. Therefore, the goal of this paper is

https://doi.org/10.1007/978-3-030-87571-8_6

to present a tool "SIR", standing for "Similarity In Records", which helps users to conduct configurable in-database similarity search of EMRs.

Despite the great progress made by EMR similarity search, existing researches still have their own defects. A typical EMR consists of diverse attributes including patient ID, gender, diagnosis and so on. These attributes are categorized into three main data types: string, enumeration and numeric data. However, early academic researches barely consider this multi-dimensional characteristic. For example, Gravano et al. [5] investigate the distance calculation algorithm for EMR's string data, but they leave the other data types undiscussed. A decade later, He et al. [6] design a system that accepts EMRs collected from hospitals as input, goes through a series of process, and eventually calculates the similarity of any two EMRs. Nevertheless, most hospitals and labs store EMRs in databases, while He's work conducts similarity search outside the database. Tashkandi and Wiese [13] notice this synthetic condition and thus propose to processes the majority of the workload within the database. Nonetheless, they calculate the distance between every EMR pair, while in reality, only the distances between the query patient and other EMRs are needed. Furthermore, McGinn et al. [9] point out that real-world users are concerned about the ease to use similarity search. Hence, a user-friendly interface that hides complicated calculation details for real-world users is essential, but current academic work rarely recognizes this need.

Our work consists of three major contributions: 1) an EMR similarity search algorithm that separates different data types, 2) an in-database search algorithm through MySQL's UDF and 3) an user-friendly interface[1] that eliminates technical barriers.

The rest of the paper is organized as follows: Sect. 2 defines the problem and Sect. 3 introduces the architecture of SIR. Section 4 presents the basic design of SIR, which utilizes nmslib to conduct external similarity search, while Sect. 5 presents the advanced design of SIR, which utilizes UDFs to conduct in-database similarity search. The implementation details are explained in Sect. 6. Section 7 explains the evaluation of both versions of SIR. Section 8 discusses some related work on patient similarity search. Section 9 concludes the paper with directions for further improvements.

2 Problem Definition

An EMR is a list that consists of n attributes, with n_s string attributes, n_e enumeration attributes and n_n numeric attributes. In order to calculate the distance between each data type, an EMR is partitioned into three small lists, each containing attributes of one data type. For string data, the list contains n_p long strings and n_q short strings. When calculating the distance between an EMR pair, each attribute has a weight value w_i. In addition, each data type has a weight value

[1] SIR is user-friendly in the sense that no heavy coding is required, but users are supposed to have knowledge in patient similarity search and know how to make their own customized search settings, such as weight configurations.

w_s, w_e, w_n, so users can calculate distance based on one data type. In this way, the distance D between two EMRs are defined as:

$$D = w_s \cdot d_s + w_e \cdot d_e + w_n \cdot d_n \tag{1}$$

where d_s, d_e and d_n are distances between string, enumeration and numeric data of two EMRs.

Each string within the string list will be partitioned into tokens (word separated by space, comma or period) and put into a string set. For the convenience of illustration, A_{pi} and A_{qi} represents the query EMR's strings sets containing long strings and short strings; B_{pi} and B_{qi} represents other EMR's strings sets. We use J and E to represent Jaccard distance and Edit distance. Jaccard distance is the disjunctive union of A, B divided by the size of the union of the sample sets[2],

$$J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} = 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2}$$

and edit distance is the minimum number of operations required to transform one string into the other. Hence, d_s is defined as:

$$d_s = \sum_{i=0}^{n_p} w_i \cdot J(A_{pi}, B_{pi}) + \sum_{i=0}^{n_q} w_i \cdot E(A_{qi}, B_{qi}) \tag{3}$$

d_e measures the distance between enumeration data, where distance is 0 if two labels are the same and 1 if not.

d_n measures the distance between numeric data, where x_i is the i^{th} attribute of the numeric data:

$$d_n = \sum_{i=0}^{n_n} w_i \cdot \frac{x_i}{max(x_i) - min(x_i)} \tag{4}$$

In order to aggregate the distances of each data type into a weighted sum D, d_s, d_e and d_n need to be normalized to the same range $[0, \omega]$. In our work, we set $\omega = 10$. If a range is too small, the difference between distances will not be obvious, but if a range is too large, computing spaces will be wasted.

3 System Architecture

The architecture of SIR is demonstrated in Fig. 1. The grey area represents the system of SIR, which consists of a EMR database, an attribute level, an EMR level and a service level. Users such as doctors and lab researchers only interact with the service level. They specify the query EMR ID, select a Top-K Value (number of similar patients they want) and adjust each attribute's weight (details explained in Sect. 6.3). Afterwards, SIR passes these information to the attribute

[2] Reference: https://deepai.org/machine-learning-glossary-and-terms/jaccard-index.

level and then the EMR level to find the distance between the query item and other EMRs in the database. Based on the distances, SIR sorts the records in the EMR database in ascending order, and the closest neighbors are returned to users as search results.

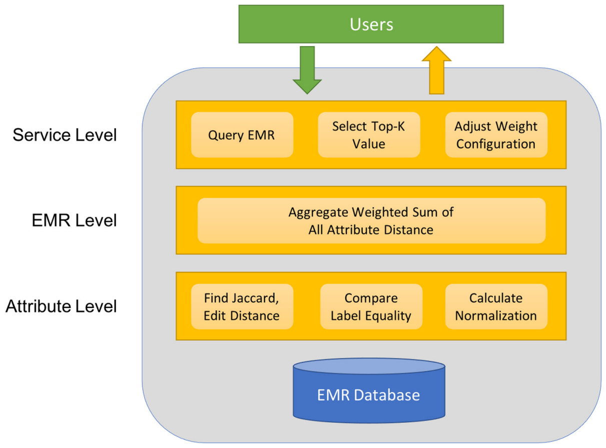


Fig. 1. System architecture of SIR

SIR has two versions that both accomplish the expected tasks. The first version (Fig. 2.a) is a basic design, which uses the Non-metric Space Library (nmslib) in an external program to perform similarity search outside the database. EMR data are first downloaded to local file for the external program, and the search results are then passed back to the interface for demonstration (details explained in Sect. 4). The second version (Fig. 2.b)) is an advanced one, which embeds the search algorithm inside the database system by User-Defined Functions (UDFs) in MySQL. SIR passes search specifications to pre-defined UDFs, and finds similar patients within the database (details explained in Sect. 5).

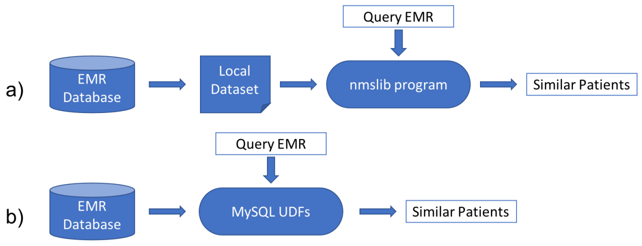


Fig. 2. Comparison of SIR's two versions

4 Basic Design

The essential idea behind the basic design is combining an external program with the user interface. The external program is in charge of similarity search while the user interface receives search specifications from users and demonstrate search results. Section 4.1 explains how the similarity search is done, and Sect. 4.2 explains how the external program cooperates with the interface.

4.1 Non-metric Space Library (nmslib) for Similarity Search

As aforementioned, a typical EMR contains many attributes about a patient. For instance, diagnosis and complaint are string data; sex and gender are enumeration data; age and heartbeat are numeric data. These attributes are categorized into different data types and need to be dealt with separately. This need of multi-dimension calculation makes nmslib a suitable choice.

nmslib is an open source similarity search library [1,8] for evaluation of k-NN methods for generic non-metric spaces. Each data type of EMR can be treated as a new dimension, or even each attribute can be treated as a new one. In this way, the distance between each dimension can be calculated separately with different methods and later aggregated into an overall value. Upon completion of the calculation, all the EMRs will be sorted in ascending order.

4.2 External Program Embedding

The decision to do similarity search as an external program is made after experiments with alternative plans. It turns out that nmslib needs to call many other local files and libraries, and several failed attempts to embed it into MySQL suggest that running nmslib as an external program is more realistic.

The external program is embedded in SIR like a specialized toolkit. It takes in the query EMR ID, the Top K value (number of similar patients needed), an EMR data file (in .csv format) and a configuration file. Then it returns the EMR ID of similar patients. Since the external program has no access to EMRs stored in the database, all the data need to be downloaded to local in .csv format in advance. When a similarity search is triggered, SIR passes the query EMR ID and the Top K value to the external program. In terms of the configuration file, there exists an initial default file, and SIR updates the file based on customized settings by users. Now, all four parameters needed for the external program become available and similarity search can be done within seconds. Upon completion of the search, the external program sends the results back to SIR for demonstration.

5 Advanced Design

5.1 User-Defined Function (UDF) for Similarity Search

Although the basic design of SIR could perform EMR similarity search, it cannot conduct synchronous search. In order to solve this problem, MySQL's UDF is employed in the advanced design of SIR. UDF stands for User-Defined Function, which is a customized function that can be called directly by MySQL like a built-in function. A UDF can return string, integer, or real values and accept arguments of those same types. Simple functions operate on a single row while aggregate functions operate on a group of rows of a database table.

Three UDF "strd", "equa" and "norm" are created to calculate the distance between string, enumeration and numeric attributes respectively. Users can specify each attribute's weight by passing parameters to the corresponding UDF. Each UDF then takes in the customized weight and returns a value from 0 to 10 that reflect the distance. Afterwards, SIR adds up the three distance values based on the overall weight proportion to get an aggregate final distance. Based on this value, SIR can sort all the records in the database to determine the most similar EMRs. Note that we assume the structure of the EMR, such as the number of string attributes, is known and fixed. The following subsections introduce the detailed algorithm of each UDF.

UDF "strd" for String Data. The UDF "strd" returns the Jaccard distance and Edit distance between string data. Edit distance calculates the distance between strings by characters, so it tolerates typos or different writing styles. However, this algorithm is relatively slow, so it only applies to strings shorter than 10 phrases. Jaccard distance is used for longer strings.

The function works as follows. Initially, each string attribute of the EMR is partitioned into string tokens (word separated by space, comma or period). Tokens from the same string attribute are then put into a string set. Next, for each string set, "strd" finds the distance between the query's set and other EMRs'. Finally, "strd" combines the distance between each set into one single value based on customized weight. Note that empty strings in the query EMR are ignored in the calculation by setting its distance with any other string set to 0, because empty strings contain no information.

UDF "equa" for Enumeration Data. The UDF "equa" returns the distance between enumeration data. Enumeration data are labels such as "male" and "female", so the distance between these data is done directly through equality comparison. Basically, the distance between two labels is 0 if they are equal, 1 if not. Likewise, "equa" finds the overall enumeration distance by adding up all the distances according to their weight.

UDF "norm" for Numeric Data. The UDF "norm" returns distance between numeric data. For each numeric attribute, "norm" first finds the largest possible range. Then, it uses the range to divide the difference between the query and the compared EMR. The ratio is considered as the distance for this numeric attribute. The overall distance is found similar to the previous two UDFs.

5.2 UDF Embedding and Combination

After the three UDFs are created, they need to be embedded into MySQL. As instructed by MySQL's manual[3], UDF is initially written in C++ as a dynamically loadable file. Next, the programming file needs to be complied into a sharable library file. Then, the shared object needs to be copied into the server's plugin directory so that MySQL can locate it and install the corresponding UDF.

After the installation of the UDFs, SIR can call these functions through its interface directly. In this way, SIR receives the distance for string data, enumeration data and numeric data separately, and then combines them according to their weight proportion. Based on this overall distance, SIR then instructs MySQL to sort all the EMRs in the database and thus finds the most similar EMRs. A sample MySQL instruction is shown below:

```
sql = "select ID, Age, Sex, Diagnosis from translate_data order by" + "
    (" + enum_prop + "*equa('" + query_sex + ",,,,," + query_race +
    ",,,,," + query_allergy + ",,,,," + query_blood + ",,,,," +
    query_condition + ",,,,," + query_status + "', " + " concat(Sex,
    ',,,,,', Race, ',,,,,', Allergy, ',,,,,', Blood_type," + " ',,,,,',
    Health_condition, ',,,,,', Heartbeat_status), '" +
    enum_weight_string + "') + ......"
```

*Instructions in purple are normal sql command. Parameters in black are weight proportions specified by users (or by default). Attributes of the same data type form one string separated by five commas (',,,,,'). Due to space limitation, only enumeration data's distance calculation is shown.

6 Implementation

6.1 Environment

System. All of the programs are written, complied and run on Ubuntu 20.04.

MySQL. Initially, MySQL 8.0 is used, but it turns out that UDF works more stably in older versions, so MySQL is degraded to 5.7 instead. Theoretically, both versions could work, but some coding details of UDF are different in the two versions.

Programming Language. The interface is programmed using Tkinter, a Python binding to the Tk GUI toolkit. Python version is 3.8.5. nmslib is an open source library [1,8]. Most of its programs are written in C++, so SIR's similarity search algorithm is written in C++ as well. As required by MySQL's manual, all UDF codes are written in C++. Complier g++-9 is used, but theoretically, any version higher than g++-7 could work.

[3] https://dev.mysql.com/doc/extending-mysql/5.7/en/udf-features.html.

6.2 Dataset

The dataset used contains 1908 EMRs obtained from Beijing Tsinghua Changgeng Hospital. Each EMR includes the following attributes (missing attributes treated as NULL):

- 5 string attributes: diagnosis, complaint, medical history, smoking history, drinking history
- 6 enumeration attributes: sex, race, allergy, blood type, health condition, heartbeat status
- 3 numeric attributes: age, number of visits to the hospital, heartbeat

6.3 Interface

The interface consists of three different windows as demonstrated in Fig. 3.

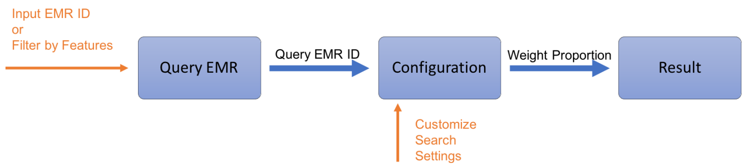
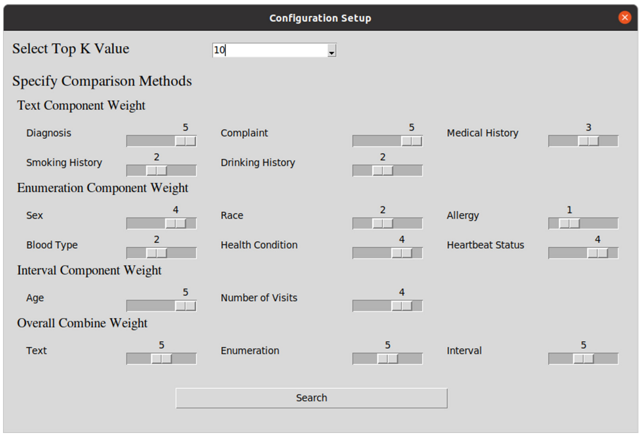


Fig. 3. The process of interface usage

EMR ID Search Window. This window is the first level that gets the query EMR ID. If users already know the query EMR ID, they can enter the ID and go to next level. Otherwise, they can select up to three features to locate the query ID. All potential EMRs filtered by the database will then pop up on the screen and users can click on one candidate to enter the next level. As soon as users click on "next", the query EMR ID will be stored backstage for further usage.

Configuration Window. This window allows the users to make customized configuration settings (shown in Fig. 4). As aforementioned, each EMR has three different type of data (string, enumeration and numeric), and each data has several attributes. When calculating distance for similarity search, each attribute contributes in a different proportion. As demonstrated, users can make adjustments by dragging the slider bars on the basis of a default setting. As soon as they click on "search", the current configuration file will be updated backstage for similarity search.



Select Top K Value 10

Specify Comparison Methods

Text Component Weight

Diagnosis	5	Complaint	5	Medical History	3
Smoking History	2	Drinking History	2		

Enumeration Component Weight

Sex	4	Race	2	Allergy	1
Blood Type	2	Health Condition	4	Heartbeat Status	4

Interval Component Weight

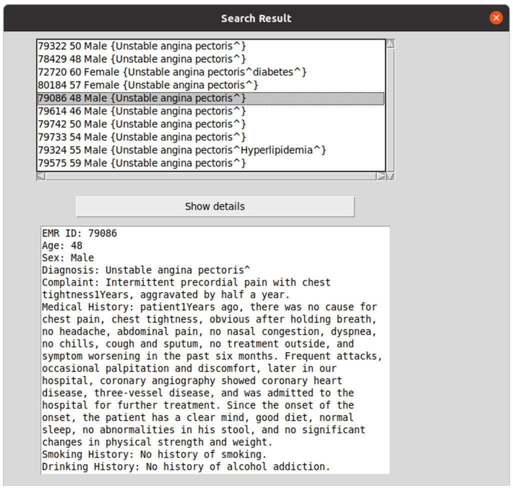
Age	5	Number of Visits	4

Overall Combine Weight

Text	5	Enumeration	5	Interval	5

Fig. 4. Configuration Window

Result Window. This window conducts similarity search backstage and then displays the search results (shown in Fig. 5). Upon entering this window, SIR has collected all the necessary information for similarity search. It passes the query EMR ID and weight configuration to either the external executable (basic design) or MySQL (advanced design), and finds the most similar EMRs. Afterwards, SIR returns the result with a few key information, and users can click on the record of their interest to see more details.



```
79322 50 Male {Unstable angina pectoris^}
78429 48 Male {Unstable angina pectoris^}
72720 60 Female {Unstable angina pectoris^diabetes^}
80184 57 Female {Unstable angina pectoris^}
79086 48 Male {Unstable angina pectoris^}
79614 46 Male {Unstable angina pectoris^}
79742 50 Male {Unstable angina pectoris^}
79733 54 Male {Unstable angina pectoris^}
79324 55 Male {Unstable angina pectoris^Hyperlipidemia^}
79575 59 Male {Unstable angina pectoris^}
```

```
EMR ID: 79086
Age: 48
Sex: Male
Diagnosis: Unstable angina pectoris^
Complaint: Intermittent precordial pain with chest
tightness1Years, aggravated by half a year.
Medical History: patient1Years ago, there was no cause for
chest pain, chest tightness, obvious after holding breath,
no headache, abdominal pain, no nasal congestion, dyspnea,
no chills, cough and sputum, no treatment outside, and
symptom worsening in the past six months. Frequent attacks,
occasional palpitation and discomfort, later in our
hospital, coronary angiography showed coronary heart
disease, three-vessel disease, and was admitted to the
hospital for further treatment. Since the onset of the
onset, the patient has a clear mind, good diet, normal
sleep, no abnormalities in his stool, and no significant
changes in physical strength and weight.
Smoking History: No history of smoking.
Drinking History: No history of alcohol addiction.
```

Fig. 5. Result window

7 Evaluation

7.1 Running Time Evaluation

In order to test the running time of SIR comprehensively, both versions conduct similarity search under different situations. Overall, the running time for both versions takes less than 1 s, but basic version of SIR runs much faster than the advanced version, because nmslib is a mature library for high-dimensional computation, while the UDFs for this work use relatively naive algorithms. The rest of this section discusses detailed analysis of the running time data.

On one hand, the number of attributes taken into consideration does not have distinct effect on the basic version of SIR, and it takes around 0.0931 s to find 5 most similar patients to a query patient. On the other hand, the running time of the advanced version varies a lot, and the data is shown in Table 1. The dataset consists of 5 string attributes, 6 enumeration attributes and 2 numeric attributes, and 10 different situations are considered. The distance calculation for pure enumeration data and pure numeric data almost takes time (both smaller than or around 0.01 s), while string attributes require much longer time to compare their distance. For the purpose of demonstration, String Attribute, Enumeration Attribute and Numeric Attribute are abbreviated as SA, EA and NA.

Table 1. Running time of the advanced version of SIR under different situations.

Combination of attributes	Running time (second)
1 SA	0.4018
1 EA	0.0093
1 NA	0.0077
2 SA	0.4151
2 EA	0.0101
1 SA + 1 EA + 1 NA	0.4121
2 SA + 2 EA + 1 NA	0.4258
2 SA + 3 EA + 1 NA	0.4258
3 SA + 3 EA + 1 NA	0.7624
All Attributes	0.7824

7.2 Interface Evaluation

Three doctors from Beijing Tsinghua Changgeng Hospital are invited to try out SIR and give feedback accordingly. They use both traditional methods and SIR to locate 5 similar patients to a query patient. For evaluation purpose, a small dataset consisting of 100 EMRs is used.

It turns out that SIR is at least 20 times faster than manual filtering methods. The doctors only took 3 min to set up the similarity search, and received the search results within seconds. However, if they manually looked for similar patients, it could took them 1 h or more.

Moreover, the doctors are especially satisfied with the interactive interface. They said that without strong programming background, using MySQL or other similar tools to filter similar patients used to be a huge challenge. However, SIR was very intuitive and required no technical experience. They only needed to input their requirements and could leave the complicated calculation to the computer. Also, SIR allowed them to assign weights to each attribute, which greatly enhanced the accuracy of the results and made the search more specific.

8 Related Work

Besides the methods introduced in this paper, EMR similarity search can be done in many other ways. Graph representation is a popular approach, as it is fast and requires very few storage spaces [4]. When the reference database is very large or the distance computation is expensive due to hardware limitation, it is more practical to find the approximate nearest neighbor [12]. nmslib is an open source similarity search library [1,8] that finds approximate nearest neighbors, though it does not have direct application in EMR similarity search yet.

Although many studies have made great progress in EMR similarity search, a few shortcomings still exist. To begin with, Gravano et al. [5] discuss the distance calculation for string attributes within the database. They develop a technique for building approximate string join capabilities on top of commercial database. However, they only consider the string attributes and give up using UDF for in-database calculation due to algorithm inefficiency. A few years later, He et al. [6] design a more comprehensive system that calculates the similarity of any two EMRs. However, the calculation is done in synthetic conditions: data are extracted from the EMR database and processed in external tools. Therefore, Tashkandi et al. [13] propose to processes the majority of the workload within the database. However, their method finds the distance between all EMR pairs inside the database. In reality, the query patient is usually known, so only its distance with other patients is needed. Furthermore, McGinn et al. [9] conducts a survey among real-world users, and finds that they are hesitant to adopt EMR similarity search due to design and technical concerns, ease of use and interoperability. Hence, a user-friendly interface is necessary if researchers want to make real-world impact, but not much work has covered this topic yet.

9 Conclusion

In this paper, both basic and advanced designs of SIR are introduced. SIR is invented as a tool for EMR similarity search that comes with a user-friendly interface. Its basic version conducts similarity search using the external nmslib, while the advanced version employs UDFs for in-database similarity search. The interface requires almost no technical background to use. It helps users to locate their query patient, allows them to make customized search configuration and shows them search results with detailed information.

However, SIR can still be improved in a few aspects. First, an adaptive interface that can automatically fit with different EMR structures is needed. In addition, the efficiency of UDFs in basic version still needs enhancement. Lastly, SIR's compatibility with other applications requires further study.

Acknowledgement. This work was supported by National Key R&D Program of China (2020AAA0109603), State Key Laboratory of Computer Architecture (ICT, CAS) under Grant No. CARCHA202008.

References

1. Boytsov, L., Naidan, B.: Engineering efficient and effective non-metric space library. In: Brisaboa, N., Pedreira, O., Zezula, P. (eds.) SISAP 2013. LNCS, vol. 8199, pp. 280–293. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41062-8_28
2. Celi, L.A., Charlton, P., Ghassemi, M.M., et al.: Secondary Analysis of Electronic Health Records. Springer, New York (2016)
3. Eteffa, K.F., Ansong, S., Li, C., Sheng, M., Zhang, Y., Xing, C.: Application of patient similarity in smart health: a case study in medical education. In: Ni, W., Wang, X., Song, W., Li, Y. (eds.) WISA 2019. LNCS, vol. 11817, pp. 714–719. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30952-7_72
4. Eteffa, K.F., Ansong, S., Li, C., Sheng, M., Zhang, Y., Xing, C.: An experimental study of time series based patient similarity with graphs. In: Wang, G., Lin, X., Hendler, J., Song, W., Xu, Z., Liu, G. (eds.) WISA 2020. LNCS, vol. 12432, pp. 467–474. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60029-7_42
5. Gravano, L., Ipeirotis, P.G., Jagadish, H.V., et al.: Approximate string joins in a database (almost) for free. VLDB **1**, 491–500 (2001)
6. He, Z., Yang, J., Wang, Q., Li, J.: A method of electronic medical record similarity computation. In: Xing, C., Zhang, Y., Liang, Y. (eds.) ICSH 2016. LNCS, vol. 10219, pp. 182–191. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59858-1_18
7. Le, N., Wiley, M., Loza, A., et al.: Prediction of medical concepts in electronic health records: similar patient analysis. JMIR Med. Inform. **8**(7), e16008 (2020)
8. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. CoRR abs/1603.09320 (2016). http://arxiv.org/abs/1603.09320
9. McGinn, C.A., Grenier, S., Duplantie, J., et al.: Comparison of user groups' perspectives of barriers and facilitators to implementing electronic health records: a systematic review. BMC Med. **9**(1), 1–10 (2011)
10. Grand Viewer Research: Electronic health records market size, share and trends analysis report, 2021–2028, April 2021. https://www.grandviewresearch.com/industry-analysis/electronic-health-records-ehr-market
11. Sherman, R.E., Anderson, S.A., Dal Pan, G.J., et al.: Real-world evidence–what is it and what can it tell us. N. Engl. J. Med. **375**(23), 2293–2297 (2016)
12. Sundari, P.S., Subaji, M., Karthikeyan, J.: A survey on effective similarity search models and techniques for big data processing in healthcare system. Res. J. Pharm. Technol. **10**(8), 2677–2684 (2017)
13. Tashkandi, A., Wiese, I., Wiese, L.: Efficient in-database patient similarity analysis for personalized medical decision support systems. Big Data Res. **13**, 52–64 (2018)