International Journal of Computer Applications (0975 – 8887)
Volume 177 – No. 13, October 2019

Cohort Search, Representation and Prediction: Application to Medical Data

Adebayo Salaudeen
Grenoble Institute of Technology

Ahmad Abubakar
Gipsa Lab
University Grenoble, Alpes

Ganiyu Saheed
University Grenoble, Alpes

ABSTRACT

In this paper, we are interested in cohort search, representation, and prediction. Information retrieval and text mining technique were proposed based on Term Frequency Inverse Document Frequency (TF.IDF) to extract important terms. Also, a formal and algorithmic model was formulated to compute: readable, concise cohorts of patients and find similarities between patient trajectories.

Finally, Patient health trajectories were analyzed using a Deep Learning architecture from intensive experimental processes based on two parallel Minimal Gated Recurrent Unit networks, working in a bi-directional manner. The obtained result shows an improvement in the performance of computer-aided medicine and serves as a guide in designing artificial neural networks used in prediction tasks.

General Terms

Data Mining, Health Informatics, Artificial Intelligence, Machine Learning.

Keywords

TF-IDF, EMR, Cohort, Neural Networks, Deep Learning, Patient Trajectory

1. INTRODUCTION

1.1 Motivation

Electronic Medical Records (EMR) is the Electronic format of traditional paper-based medical records that stores patient health information in a digital format. Integrating or sharing EMR helps to facilitate the collaboration between healthcare entities and also enhance the efficiency and quality of care through enhanced information-sharing capabilities. The primary advantage of using EMR is “more and better access to patient information” in compare to Paper-based Patient Record [1]. The EMR will strengthen the position of the patient because he can have more and easier control over his health information and can follow the progress of his own illness [2]. Electronic Medical Record systems (EMRS) aim to provide an environment for safely sharing and exchanging patients’ records.

Hospitals have a Health Information System (HIS) centered on each patient. Several hospitals are using the same set of tools. To facilitate health management and treatment, those tools must be enriched with patient similarity. All hospital admissions, doctor visits, administrative data, traceability records, and events are recorded in these tools. On the other hand, due to the increase in health-care data in various medications such as hospital admissions, doctor visits, administrative data, etc. medical experts require an effective system to assist them to better understand and analyze the evolution of their patients’ health. They also need to know how important a term is in a cohort of patients, similarities between sets of patients with respect to time and finally what will happen next to a patient in a cohort-based on their past. A cohort analysis system helps to augment treatment effectiveness, patient satisfaction, and health-care revenue [3]. A medical cohort system assist the expert in answering two frequently asked questions: “what is happening?” and “what happens next? [4]. The first question is about patient illness diagnosis whereas the second is about predicting future medical risk.

1.2 Problem Statement

Despite the benefits of the techniques mention above, they were unable to compute similarities between patients. Such similarities would enable the retrieval of patients with common demographics such as age and geographic location but also patients with similar medical histories, e.g. those who underwent comparable treatments. This research work was aimed at assisting medical experts to identify the most frequent terms present in patient files, extract main terms from a list of patient’s documents in a search query using a word cloud to visualize them. It also assists in building a common story for list of patients with respect to their medical actions and occurred time. And finally using a Neural network (deep learning) approach to predict future outcomes of patients' disease progression and future risks from their historical admissions and present health states.

The main contribution of this work are listed below:

a) Classifying or grouping terms in patients' data (cohort), extracting meaningful/important terms and improving patient information retrieval.

b) Building common stories for different patient groups (patient cohorts) from their trajectories.

c) By learning from an individual patient’s medical history and present status, we applied the Deep Learning method to build a medical prognosis method using recurrent artificial neural networks.

The outline of this paper is divided into sections as Section II describes the data model, which formally defines the problem of cohort search, representation, and prediction. The experiment was carried out in Section III. A set of extensive experimental results is analyzed in Section IV. Discussion of results in Section V. And conclusion in Section VI.

2. DATA MODEL

2.1 Data Issues

Admissions and diagnoses were explored to predict the trajectories. A patient’s admission refers to a set of diagnostic codes that describe what happened during a hospital stay. The real Health dataset used has some challenges due to its cardinality and it uses ICD-10 encoding which is highly granular.

These challenges were tackled using the idea in the research work of [5] as illustrated below:

a. Cardinality: The total number of admissions in the dataset is 58,976. However, the distribution of patients considering the number of admissions is skewed, see Figure 1 – 38,983 patients have one single admission. This fact severely reduces the dataset as only data of patients with at least two admissions is useful for trajectory prediction. Besides that, a few admissions do not have any related ICD-10 codes, and others are not meaningful (negative time duration). With these restrictions, the number of admissions falls to 19,911; the number of patients falls from 46,520 to 7,483.

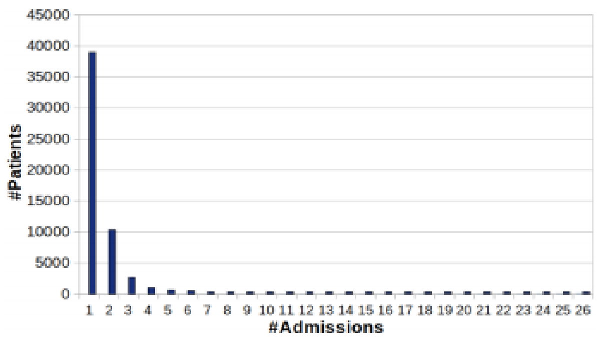


Fig 1: Distribution of the number of patients with respect to the number of admissions in the real Health dataset

b. ICD-10 encoding: Concerning the encoding of diagnoses codes, the drawback comes from the high cardinality of the ICD-10 standard, whose number of diagnosis codes sums up to 68,000. This cardinality refers to the granularity of details, which describes a disease along with its possible clinical manifestations. In the real Health dataset, a total of 8,114 codes appear in the database instance – Figure 7 presents the distribution of the number of codes with respect to the number of admissions

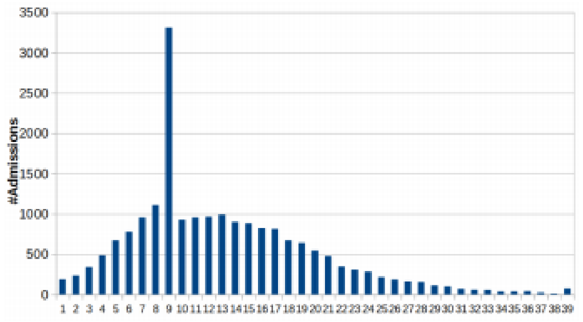


Fig 3: Distribution of the number of admissions with respect to the number of medical codes in the real Health dataset

c. Alternative ICD-10 encoding: The high granularity of the ICD-10 standard is a problem not restricted to this work; rather, ICD-9 is a recurring problem in several research activities. The Healthcare Cost and Utilization Project (HCUP), a North-American association dedicated to healthcare research has tackled the problem by issuing the Clinical Classifications Software (CCS) encoding [6]. Their classification scheme (not a software) defines a specialist-established tabular mapping from ICD-9 to a less granular descriptive standard, the CCS. The goal is to ease statistical analysis and reporting. This classification scheme has not been extended to ICD-10, therefore CCS could not be used on the ICD-10 codes. ICD-10 code first three characters categorize the generic name (CCS) of injury while the fourth to sixth gives details about the injury for example, instead of using code I44.2 for Atrioventricular block, complete, we used only I44, which stands for Atrioventricular and left the bundle-branch block.

Thus, only the CCS was returned and neglect the remaining details in this research. Table 1 illustrates the mapping of the complications of bariatric procedures from ICD-10 to CCS ICD-10 – in this example, 24 ICD-10 codes become 1 CCS ICD-10 code.

Table 1. Example of ICD-10 to its CCS Mapping

ICD-10 code	**ICD-10 description**	**CCS**	**Generic Description**
K94.00	Colostomy complications	K94	Digestive system
K94.01	Colostomy hemorrhage	K94	Digestive system
--	--	---	--
K94.32	Esophagostomy infection	K94	Digestive system
K94.33	Esophagostomy malfunction	K94	Digestive system

Before preprocessing, only the diagnosis code of the real Health dataset was used for the prediction phase, which contains 8,897 admissions regarding a sample of 8,209 complete de-identified patients; the data is related to 3,047 diagnoses encoded in the ICD-10 standard. After filtering out inconsistencies and patients with one single admission, we ended up with only 1,457 admissions related to 669 patients; encoding in the real Health dataset uses 3,047 diagnosis codes with their respective textual descriptions based on the ICD-10 standard. The granularity was reduced at the cost of losing details but gained in prediction performance by reducing the size of the problem in Table 1. Consequently, this decision reduced the number of codes in the real Health dataset to 559.

2.2 Problem Modeling

The first problem in this research is defined as a cohort c of patients' $p(c) \subseteq \mathcal{P}$, extract the important terms from the patient documents. A cohort c has a tuple $(cond, outcome)$ that contains a set of patients'. The outcome is the outcome of $p(c)$ while $conds$ include one or several conditions to be met by members and each condition is a tuple $conds = (a, v)$ where a is an attribute and v is its value. $D(p) \subseteq D$ with $p \in p(c)$ are patient documents in $p(c)$ and $T(d) \subseteq T$ with $d \in D(p)$ are terms in the patient documents. Therefore, the function $R: p(c) \times D \times T \longmapsto [0, 1]$.

Remarks: TF-IDF is used to extract the important terms from the cohort. The closer the value of R is to 1 the better.

The second research problem is a cohort c and a confidence threshold σ, return a readable and succinct representation of events for $c's$ members $\mathcal{E}_c^*$, where for each confidence $(e) > \sigma$. A trajectory matching m between two events $e1 \in T(p1)$ and $e2 \in T(p2)$ is used to compute the similarity score between $T(p1)$ and $T(p2)$ i.e. $m = (action, time, value) \equiv m.action = e1.action = e2.action.$

For the intuitive representation of time, mean was introduced to achieve this

$$m.time = [mean\ (e1.time, e2.time)] \quad (1)$$

$$m.value = \frac{max(|T(p1)|,|T(p2)|) - |e1.time - e2.time|}{\binom{|p(c)|}{2} \times max(|T(p1)|,|T(p2)|)} \quad (2)$$

and the value was assigned to m so that the more frequent matches get a higher value using equation (2). In other to achieve readable and succinct representations of events for $c's$ members $\mathcal{E}_c^*$, a confidence threshold σ is used where a higher σ increases succinctness.

The final research problem is stated as: given a patient's sequence of admissions from an EMR, predict the most probable diagnoses that will appear in the next admission of this patient at a given time $t+1$. A patient's admission refers to a pair $a_i = (t_i, d_i)$, in which i is the temporal order of the admission, t_i is the timestamp stating when the admission occurred, and $d_i = \{d_{i,0}, d_{i,1}, \ldots, d_{i,n-1}\}$ is an unordered set of $|n|$ diagnoses codes, so that $d_{i,j} \in D$, in which D is a standard set of codes such as ICD-10

3. EXPERIMENTS

3.1 Keyword Extraction

As explained in section 1.3, keyword extraction helps medical experts identify the most important terms present in patient files or extract main terms from the list of patient's documents in a search query using the tag cloud to visualize them.

3.1.1 *IR Method used to show the Importance of Terms in Documents*

One of the best ways of weighting or scoring terms in a corpus in information retrieval and text mining is using TF-IDF. TF-IDF is a state of the art algorithm that assigns a weight to how important a word or term is to a document in a collection or corpus [6]. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus Variations of its weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

Taking a peek at the dataset, the result below shows a one per line JSON string from all patientDoc $D(p) \subseteq D$, read into *Pandas Dataframe* and some random samples were printed. The dataset consists of patient "personal details", "health history", "prescription" etc.

```
                                         file.contenu

                                 Nombre d'enfants : 0
Ressource : non; Nombre d'enfants : 0; Retrait...
Intervenant id : 25464; Intervenant txt : RAYM...
Id de l'intervenant de création : 23396; Id de...
```

3.1.2 *Data Cleaning*

After reading the dataset in a structured format, the next step is to perform some data cleaning which includes removing tags, removing special characters and digits, converting to lowercase and stemming. Other cleanings were performed by parsing the dataset through a special function written mainly for this process (data cleaning) before the actual computation is done on them.

```
'ressource non nombre d enfants retraite habitation
appartement propriétaire locataire locataire sécurité
sociale oui mutuelle oui ressources'
```

3.1.3 *Parsing Stop-Words, Creating Vocabulary and Word Counts for IDF*

A custom stop word list was generated and parsed to remove unnecessary words from the dataset. $CountVectorizer$, from Sci-kit-learn a python library was used to create a vocabulary from all the text in the dataset, then counting of words in the vocabulary which are returned in the form of term-document matrix e.g.in this searched query result, we have 77,743 documents (the rows) and the vocabulary size is 98,526. A few of the vocabularies are shown below.

```
['ressource', 'retraite','habitation','appartement','propriétaire',
 'locataire','sociale','mutuelle','ressources','intervenant']
```

3.1.4 *Compute Inverse Document Frequency (IDF)*

Computing the IDF values was done by invoking $tfidf\_transformer.fit()$ to the sparse matrix from $countVectorizer$ above and returns the IDF values.

```
array([11.56802948, 11.56802948, 11.56802948, ..., 11.16256437,
        7.70729977, 11.16256437])
```

3.1.5 *Computing TF-IDF and Extracting Important Terms*

Once the IDF computed, computing the TF-IDF and extracting top keywords from the TF-IDF vectors. The TF-IDF value for all the terms in the query result from the dataset is computed by invoking $tfidf\_transformer.transform$ which return the weight/value of each term of the document.

```
locataire 0.622
ressources 0.411
propriétaire 0.31
appartement 0.277
retraite 0.273
```

3.1.6 *Filtering with Medical Dictionaries*

Hospitals have various departments with their respective dictionaries and medical experts prefer to see how important a term is with respect to the different departments. Six medical dictionaries were available and all the terms in them were used to compare terms in the dataset in order to avoid the problem in [7], i.e. to show relevant words or capture minor lexical variations, human error or absence of an accent were likely cases that can hinder the extraction of important words. To overcome this, a sequence similarity metrics known as Levenshtein distance is used, which measures the difference between two strings of characters and calculates the minimum number of characters that must be deleted, inserted or replaced to move from one channel to another [8]. For instance, if a two-word sequence differs by only a single word, then their Levenshtein distance is one i.e. only one substitution is needed. This was used to calculate the distance between a term in the dataset and terms in the medical dictionaries, which is better than using the comparative equation which does not consider the position of each letter in a term. A threshold of ≤ 1 was used to get a better result, the lower the distance the better the similarities i.e. (0 means no changes and 1 means just a string changed). All length of terms less than 2 were neglected before using the Levenshtein distance to prevent more noise in the tag cloud. The algorithm below iterates through all the terms in the real Health dataset compared it with the terms in the medical dictionaries and increase the scores within the threshold while other scores remain the same, resulting in a new *TF-IDF* which is saved in R. The complexity of the algorithm is $O(NM)$

Algorithm 1: Levenshtein distance

```
Input: Cohort c, a threshold
Output: //a list of terms and their scores
R: []
patientDoc <- c.getAllDocument
for each term in patientDoc:
    for a word in MedDict:
        cmpFile=jellyfish.levenshtein_distance (terms, word)
        if (cmpFile <= min_dissim):
            min_dissim = cmpFile
        if min_dissim == 0:
            break
    new_tf_idf=tf_idf + ((threshold - min_dissim) / threshold)
    R.append((terms, new_tf_idf))
return R
```

3.1.7 Generating Tag Cloud for Visualization

Finally, a visual representation (tag cloud) of text data was used to show important keywords from patient documents. Using the scores of each term in the dataset, the tag cloud increases the font size or color of terms with higher scores and thus seen as the most important terms in the cohort (fig.3).

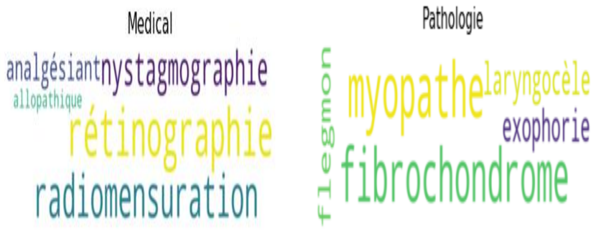


Fig 3: Tag Cloud to Visualize Important Terms from a Cohort

All processes involved in this phase is summarized in fig. 4

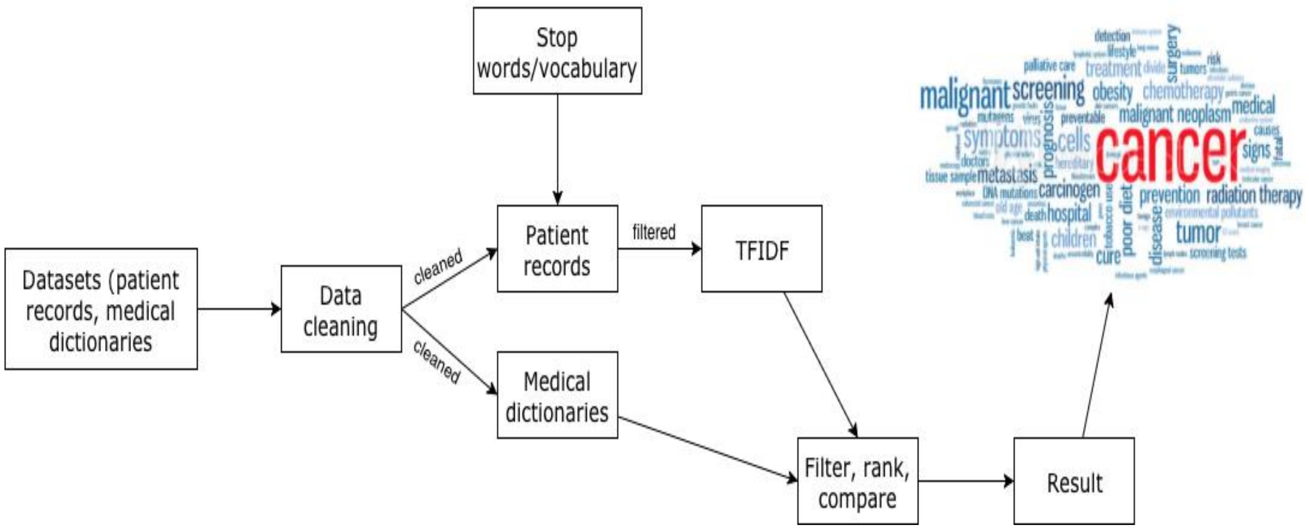


Fig 4: Flow chart of Keyword Extraction

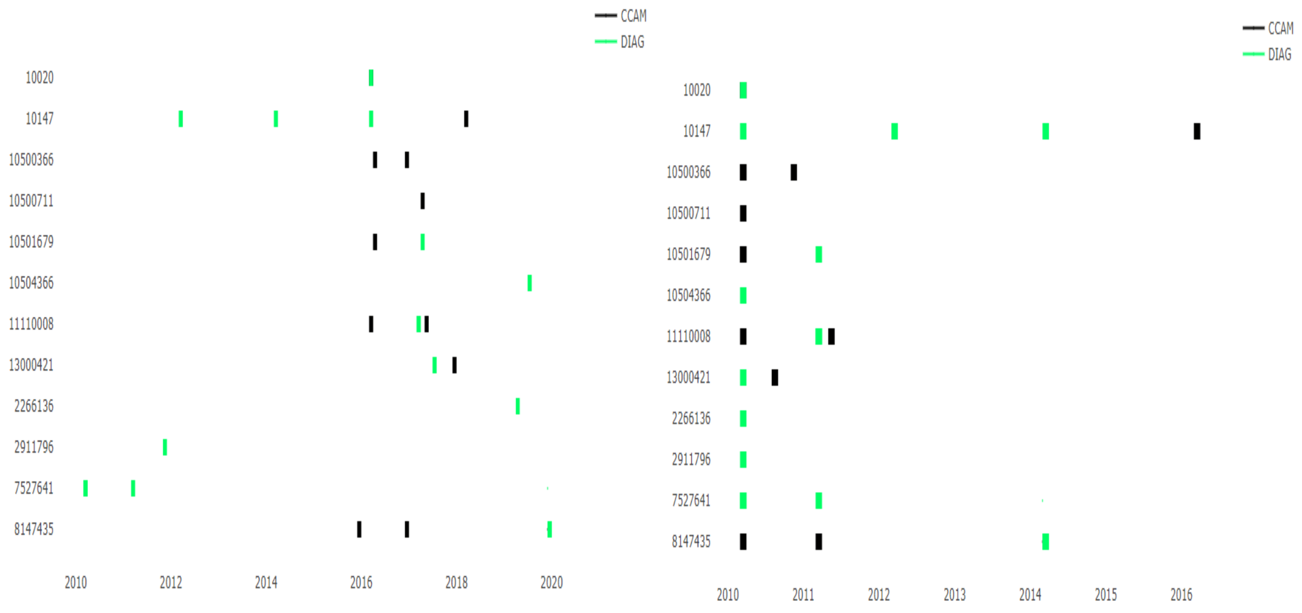


Fig 5: Trajectories of the Cohort's Members

3.2 Building Common Stories between Patients over Time

The second phase of this research is to build a common story from sets of patient stories in a cohort. The main challenge is that the cohort mostly consists of many patients with various types of events.

We applied the algorithm 2 of Cohort Representation used in [3] to the dataset to get the collective behaviors of patients over time. We defined five key steps in effecting this algorithm on our dataset which is explained below;

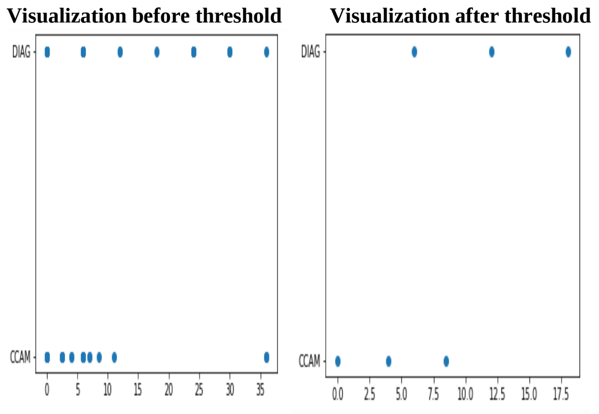
i. Get each patient trajectory from the dataset with respect to their actions and timestamps; some conditions were given to the query to generate data for the cohort. The cohort consists of n patient trajectories (is a list of temporary sorted events i.e. $T(p) = (\{p, action, time\} \in \mathcal{E})$). A flat representative of the trajectories of all n patients was shown using a Gantt chart in figure 5, which shows that the number of patients can be large and flattening which is a bad idea. An ideal cohort representation should describe a single end-to-end storyline for the cohort, be limited to events which matter the most to the cohort (i.e., succinct), and align those events in such a way that it is readable [3].

ii. Using time zero technique in statistical analysis [9] to align all patient trajectories on their first action in the dataset to time zero in figure 5-right. i.e. the exact sequence and distance between events of every single trajectory are preserved and matching can be performed. We also used a monthly time bins for the time granularity due to the real Health dataset structure i.e. any event that occurs within a month, e.g. 2nd and 25th is mapped to time bin of the first month and we calculated the length of a patient trajectory by calculating the difference between the first month and the last month of events in the hospital i.e. $|T(p)| = month_{between}(first_{day}, last_{day})$

iii. Check for common matches between patient trajectories; due to the manner of a patient timestamp in figure 5-left. There is variation in the patient events and no order of transition between them, it was difficult to match using the idea of [10] but [11] idea was extended in order to find global matching between patient trajectories which considers relations before and after successive events.

iv. Equation (1) was used to get the intuitive representation of matching time and equation (2) assigns higher matching value to more frequent matches, all match events between different patient's trajectories pairs with respect to time were recorded in a matching table which consists of the match events, their mean-time and matching values are shown in figure 6b.

v. A representation matrix whose rows are events and columns are times was plotted in figure 6a; a threshold is given by the expert to prevent noisy events and get a single-line story cohort. A running example from a real health dataset consisting of 141 patients is shown below; a full confidence value is assigned to matches at time 0 because it occurs at the same time in the trajectories while the confidence for other events is lower, as their corresponding matches are not frequent among all trajectory pairs.



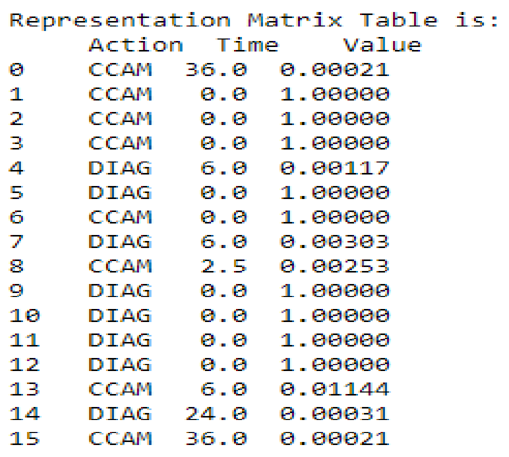

```
Representation Matrix Table is:
      Action  Time    Value
0     CCAM    36.0   0.00021
1     CCAM     0.0   1.00000
2     CCAM     0.0   1.00000
3     CCAM     0.0   1.00000
4     DIAG     6.0   0.00117
5     DIAG     0.0   1.00000
6     CCAM     0.0   1.00000
7     DIAG     6.0   0.00303
8     CCAM     2.5   0.00253
9     DIAG     0.0   1.00000
10    DIAG     0.0   1.00000
11    DIAG     0.0   1.00000
12    DIAG     0.0   1.00000
13    CCAM     6.0   0.01144
14    DIAG    24.0   0.00031
15    CCAM    36.0   0.00021
```

Fig 6a-b: Running Example of Cohort Representation

3.3 Prediction from Patient Trajectory

After extensive testing, low cardinality of the datasets poses a great challenge for diagnosis prediction. As a consequence, we had to test a broad set of artificial neuron cells aiming to achieve high predictive performance. We focused on recurrent neural networks, which are recognized for their ability to deal with sequences in time. This makes our settings susceptible to a number of parameters; every time a significant number of parameters (layers of neuron nodes) are added, the recall would pointedly fall. We hypothesize that the small number of instances of the dataset made it difficult to have the network learn the underlying patterns, therefore reducing the recall for both training and testing. To cope with that, we tested many techniques.

Among the recurrent networks found in the most-accepted literature, the one with the smallest number of weights is Jordan's network; the one with the biggest number of weights is Google's LSTM. The GRU network demonstrated higher performance than Jordan's by using more weights, but with less performance than Google's LSTM. The Minimal GRU (MGRU) network, introduced by Zhou et al. in 2016, uses even fewer weights than the classical GRU and, just as demonstrated by its authors, it did not lose performance despite using fewer gates – its performance was slightly superior to Google's LSTM and classical GRU, but demanding less processing time. Hence, MGRU was chosen to be the core of our architecture; its equations are:

$$\boldsymbol{f}_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \qquad (3a)$$

$$\hat{\boldsymbol{h}}_t = tanh(W_h x_t + U_h(f_t \odot h_{t-1}) + b_h) \qquad (3b)$$

$$\boldsymbol{h}_t = (1 - f_t) \odot h_{t-1} + f_t \odot \hat{\boldsymbol{h}}_t) \qquad (3c)$$

where $\sigma = \frac{e^x}{e^x + 1}$ is the sigmoid activation function; $tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is the hyperbolic activation function; the W*'s* and U*'s* are the weights to be optimized, together with the biases indicated with $b's$. We initialize the squared matrices (U) using identity, and the other matrices (W) using Xavier initialization [12], that is, using a Gaussian distribution with $zero\ mean$ and $variance = \sqrt{\frac{2}{|input| + |output|}}$.

Despite the intuition that more layers should lead to better performance, our experiments demonstrated, for every tested technique, that the higher the number of hidden layers the worst the performance. We observed lower performance in both training and testing, so it was not a matter of overfitting, but of capacity to learn the underlying function. This result was intriguing, especially because many authors advocate their technique to be immune to the vanishing/exploding gradient problem or to support deeper networks satisfactorily. We did not further investigate the underlying reasons, but just verified that these claims were not valid for our problem setting – notwithstanding, this problem has been a topic of active research [13]. our results are in sync with a recently awarded work, by Frankle and Carbin [14], who state that neural networks can be as much as 90% smaller without losing performance. In our tests, hence, we designed architecture with one input layer, one MGRU hidden layer, and one standard output layer before the softmax probability distribution. Despite the satisfactory results, we hypothesized that more weight could help because they can detect more about the underlying patterns. However, since stacking more layers did not help, we explored using the principle of bidirectional recurrent neural networks [15].

Bi-directional recurrent neural networks connect two processing flows computed in opposite directions in relation to the temporal dimension of the temporal data. As a result, the output layer gets information regarding past and future states simultaneously. For our problem setting, this architecture represented significant performance gains. We used the parameterized Rectified Linear Unit (LReLU) at the feed-forward stage of the network which demonstrated superior results with respect to recall and speed of convergence as compared to functions sigmoid, hyperbolic tangent, and classical Rectified Linear Unit (ReLU).

4. EXPERIMENTAL PERFORMANCE EVALUATION

4.1 System Settings for keyword extraction (Search) and Representation

Due to the unavailability of medical experts, we were only able to provide one set of experiments i.e. the performance aspects of our algorithms. All experiments in this phase were carried out using python3.6 and are conducted on a 3.30GHz Intel Core i5 with 16GB of memory on the Windows10 Enterprise operating system.

A real health-care dataset with 802,206 events for 275,307 patients with a focus on diagnosis problems was used for this research. Some of the patient's actions are treatment, compliance, etiology, fatigue marker, BMI marker, and hospitalization. In the dataset, an average trajectory has a length of 41 months and contains 11.12 events.

4.1.1 Performance Study with respect to Search and Representation

The algorithm was used on three different cohorts from the real Health dataset. The smallest cohort has 29 old female patients that live in Aoste, France with 240 events; the Medium cohort has 50 patients living in Aoste, France with 724 events while the last and biggest cohort has 203 young patients that live in Genay with 1638 events. We compared the execution time in relevance to the number of patients in the cohort and secondly, we checked the relevance of the important words displayed by the tag cloud.

Our experiment shows that cohort with the lowest patients executed faster than those with a higher number of patients because it has fewer terms in it, so it was able to run faster than others with more terms. Also, the medical dictionary has a lot of terms which makes our algorithm runs a bit slower. Our result also shows that cohort with the small patient has some noise in it because its documents are very small, some unimportant words surface while those with higher patient shows better results. Generally, our algorithm works better for the cohort with many documents.

For Cohort representation, we consider threshold σ=0.03 in order to obtain at most 10 events in our final representations (based on maximal human perception capacity explained in [16]).

Table 2. Successive Execution Time with respect to Cohort size and its Standard Deviation

Cohort Size	Threshold	1st	2nd	3rd	Mean Time	Standard Deviation
Small	0.03	0.562	0.546	0.547	0.55166667	0.009
small	0.3	0.547	0.562	0.563	0.55733333	0.009
small	0.8	0.547	0.547	0.547	0.547	0
medium	0.03	1.266	1.297	1.297	1.28666667	0.0179
medium	0.3	1.313	1.312	1.266	1.297	0.0269
medium	0.8	1.281	1.313	1.297	1.297	0.016
large	0.03	3	3.031	3	3.01033333	0.0179
large	0.3	3	2.969	2.984	2.98433333	0.0155
large	0.8	2.984	3	2.984	2.98933333	0.0092

For the performance study, we measure the efficiency of our algorithm and verify the influence of input parameters on execution time. We experimented with our algorithm on the three cohorts (small, medium and large), ran the algorithm three times to get a mean execution time and the standard deviation (Table 2) using different thresholds as presented in figure 7. From the left chart of figure 7, it was observed that the execution time was not affected by the varying threshold. This is true because the threshold is post-processed. However, the cohort size and number of events in each cohort have a strong influence on the execution time. The right chart of figure 7 shows that execution time grows bigger as the cohort increases from small, medium and large i.e. as the number of events increases. This is realistic because the matching algorithm has to compare all trajectory pairs where the number of pairs grows to the size of the cohort is quadratic.

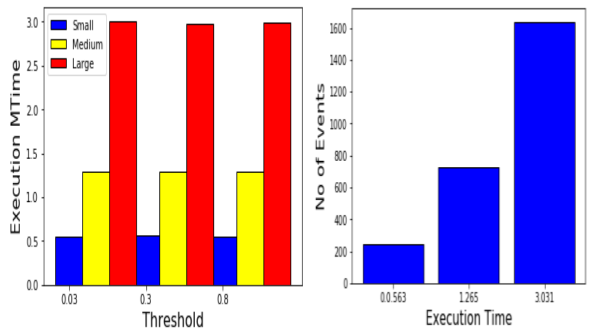


Fig 7: Impact of Threshold and Size of the Cohort on Execution Time

4.2 System Settings for Prediction

For training and testing, 90% and 10% of the patients were used respectively. The training occurred until the model recorded 10 consecutive epochs without improvement as measured by reductions in the cross-entropy loss.

4.2.1 Comparison to Related Works

We reproduce the results in related works to prediction, this is because none of them is completely reproducible due to code or data access – in every case, and we considered the best-reported results. We compare other works to our results, the comparison is meant for a perspective of our work with respect to others; however, it must be carefully interpreted – all the former works use private non-accessible datasets on specific diseases. The ideal comparison should be over the same dataset and in similar conditions, which is not possible.

In Table 3, we see that the best results of [17]. are worse than what we obtained for the real Health dataset. Also in the real Health dataset ICD-10 encoding obtained F1-Score 0.413 against 0.4 of [17]; considering AUC-ROC, their best result using their private data was 0.9, which is a bit better than what we got in our settings. Again, this is not absolutely conclusive because of adverse experimental conditions. The work of [4] recommended both the use of an embedding layer and of cell LSTM; but, our result demonstrated that this is not the case for our low-cardinality settings – embedding, in particular, was a very bad design choice. Furthermore, [4] report on their DeepCare system and on a Markov-based methodology using metric *Precision @ (1, 2, 3).* As shown in Table 3, their best results over their private datasets are better than what we got.

DeepCare and the Markov based method, shows at first glance, that our result was not comparatively so good. However, after preprocessing, their mental health dataset had 52,049 admissions and 247 diagnosis codes, and their diabetes dataset had 53,208 admissions and 243 codes – a problem much easier than ours, which is less specific, more granular, and with lower cardinality. Actually, their dataset's characteristics are comparable only to Mimic-III; moreover, the real Health dataset has generated from outpatient (non-specialist) admissions, Mimic-III from critical care admissions, and DeepCare for mental health and diabetes. The patient trajectory prediction depends on the context of healthcare, being more challenging when it is not specific. In Table 3, we also report the best results of *Doctor AI* over their private dataset; their results in this configuration are not that better than ours, but better for *Recall@30*. This performance is explained by the characteristics of their dataset, with over 14 million admissions and 1,183 codes, which provides strong training support for neural networks. [18]. also mention Mimic-III using ICD-9, which achieved a ratio of 0.55 for *Recall@30*, and 0.64 for *Recall@30* after transferring the learning of their private dataset training. Our best result for the real Health dataset using ICD-10 was 0.766 for *Recall@30* – also presented in Table 3.

Table 3. Metric Comparison to Related Works.

	Rajkumar et al. Mimic-III ICD	**Real-Health Dataset**
F1-Score at discharge	0.4	0.413
	Rajkomar et al. (private dataset)	Real-Health Dataset
Frequency weighted AUROC	0.9	0.882
	Pham et al. (4) DeepCare / Markov (best results - all datasets)	**Real-Health Dataset**
Precision@1	0.662 / 0.551	0.603
Precision@2	0.596 / 0.341	0.513
Precision@3	0.537 / 0.243	0.402
	Choi et al. DoctorAI (private dataset)	**Real-Health Dataset**
Recall@10	0.643	0.688
Recall@20	0.743	0.751
Recall@30	0.795	0.766
	Choi et al. - DoctorAI Mimic-III	**Real-Health Dataset**
Recall@30	0.64	0.766

5. DISCUSSION OF RESULTS

As earlier explained, our model was successfully ran on three different cohorts with different sets of documents. One of the limitations of TF-IDF was its inability to identify slight changes in its tense, this was overcome in our research by stemming all terms to its root words. Another limitation in earlier research was that TF-IDF could not check the semantics of text in documents, we used Levenshtein distance to get important terms in the real Health dataset that are the same or slightly different from terms in our medical dictionaries. This prevents unexpected results and improves the result of our experiment. Though, there is still more to be done especially in the aspect of Levenshtein distance which mostly considers the deletion, insertion or modification of strings in words but does not consider the semantic meaning of the words.

In addition, adding as much as possible words to the stop words document will filter and remove more unwanted words in the dataset before data processing [6]. Regarding keyword extraction, knowing the domain of treatment for a set of the patient cohort will go a long way in showing only important documents related to that domain and the algorithm will process faster.

During the prediction phase, our first finding was that the recurrent network techniques do not accomplish many of the claims that abound in the respective literature. Specifically, we verified that they do not support the stacking of layers – the more layers, the worse the performance; effectively, our design ended up with one single layer. We also noticed that the networks whose cells used more gates, like LSTM and GRU, had the same performance as of the network based on the much simpler Minimal GRU cells, which we ended up choosing for our design. When one considers the theoretical claims on why each gate is part of a given network, the facts do not support the theoretical premises – for our specific settings, the fewer gates, and the better. Complexity seemed no to be the path to follow. These two findings were not exhaustively investigated, notwithstanding, our results alert that some assumptions taken for granted must be revisited.

6. CONCLUSIONS

The goal of this research internship is to build a system that will enable medical experts to easily search for important terms from a cohort and view a common story between a set of patient actions in the cohort. We used an information retrieval and text mining technique called TF.IDF to assign weight or values with respect to how important each term is in a set of documents and we used the approach of [3] to get common stories between patients in the same cohort. We also conducted broad experimentation over a real health dataset using the approach of [5]. During this phase, we considered only the DIAG (diagnoses) codes because it's coherent and it also enables easy comparison with other current research results due to its popularity in this field of research. This work was based on predicting the possible next action of a single patient, we plan to extend it to predicting the next possible action in a set of a patient (cohort) in future research work and also extending it to other medical codes.

Finally, using a parallel architecture demonstrated to be a promising design decision; even further, this design can be extrapolated to more than two parallel networks, each one benefiting from different characteristics of the data. We also suggest that the whole methodology be experimented over more specific datasets, as for predicting finer onsets, like heart failure, or strokes; and also, for predicting when the next onset might take place, as the data is rich with respect to temporal information.

7. ACKNOWLEDGMENTS

The authors wish to thank all the respected professors who helped during the experiment, development and writing phase of this paper.

8. REFERENCES

[1] E. Balka, “Electronic patient records,” Nurs. Manage., vol. 5, no. 4, pp. 39–39, 2016.

[2] J. G. Beun, “Electronic healthcare record; a way to empower the patient,” Int. J. Med. Inform., vol. 69, no. 2–3, p. 191—196, Mar. 2003.

[3] B. Omidvar-Tehrani, S. Amer-Yahia, and L. V. S. Lakshmanan, “Cohort representation and exploration,” Proc. - 2018 IEEE 5th Int. Conf. Data Sci. Adv. Anal. DSAA 2018, pp. 169–178, 2019.

[4] T. Pham, T. Tran, D. Phung, and S. Venkatesh, “Predicting healthcare trajectories from medical records: A deep learning approach,” J. Biomed. Inform., vol. 69, pp. 218–229, 2017.

[5] S. Rodrigues-Jr, JF; Spadon, G; Machado, BB; Amer-Yahia, “Patient trajectory prediction in the Mimic-III dataset challenges and pitfalls,” 35ème Conférence sur la Gest. Données – Principes, Technol. Appl., 2019.

[6] S. Qaiser and R. Ali, “Text Mining : Use of TF-IDF to Examine the Relevance of Words to Documents Text Mining : Use of TF-IDF to Examine the Relevance of Words to Documents,” no. July 2018.

[7] K. Hans, “Das System Kaliumsulfai-Kaliurnsulfid,” Zeitschrift für Anorg. und Allg. Chemie, vol. 164, pp. 45–56, 1931.

[8] E. S. Ristad and P. N. Yianilos, “Learning string-edit distance,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 5, pp. 522–532, 1998.

[9] A. Heuser, M. Huynh, C. Joshua, and J. C. Chang, “Asymptotic convergence in distribution of the area bounded by Kaplan-Meier curves using empirical process modeling,” 2014.

[10] W. David and D. H. Rich, “Book, Software, and Web Site Reviews,” no. 11, pp. 2219–2221, 2005.

[11] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” J. Mol. Biol., vol. 48, no. 3, pp. 443–453, 1970.

[12] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, vol. 9, pp. 249–256.

[13] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, “How to Construct Deep Recurrent Neural Networks,” pp. 1–13, 2013.

[14] M. Carbin, “THE LOTTERY TICKET HYPOTHESIS :” pp. 1–42, 2019.

[15] M. Schuster and K. K. Paliwal, “Bidirectional Recurrent Neural Networks,” vol. 45, no. 11, pp. 2673–2681, 1997.

[16] G. A. Miller, “Human memory and the storage of information,” {IRE} Trans. Inf. Theory, vol. 2, no. 3, pp. 129–137, 1956.

[17] A. Rajkomar et al., “Scalable and accurate deep learning with electronic health records,” npj Digit. Med., vol. 1, no. 1, p. 18, 2018.

[18] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, “Doctor AI: Predicting Clinical Events via Recurrent Neural Networks,” JMLR Workshop Conf. Proc., vol. 56, pp. 301–318, Aug. 2016.