Managing Diabetes: counselling supported by user data in a mobile platform

Diogo Machado[1], Inês Dutra[2], Pedro Brandão[1], and Vítor Santos Costa[2]

[1] Instituto de Telecomunicações, Porto, Portugal
[2] CRACS, INESC-TEC, Porto, Portugal
Faculty of Sciences, University of Porto, Porto, Portugal
{dmachado, ines, pbrandao, vsc}@dcc.fc.up.pt

Abstract. Diabetes management is a complex problem. The patient needs to monitor several parameters in order to react in the most appropriate way. Different situations require the diabetic to understand and evaluate different rules. The main source of knowledge for those rules arises from medical practice and is usually transmitted through medical appointments. Given this initial advice, most patient are on a continuous process of managing the disease, toward achieving the best possible quality of life.
Motivated by recent aadvances in diabetes monitoring devices, we introduce a diabetes support system designed to accompany the user, advising her and providing early guidance to avoid some of the many complications associated with diabetes. To accomplish this goal, we incorporate standard medical protocols , advice and directives in a Rule Based System (RBS). This RBS which we call Advice Rule Based System (ARBS) is capable of advising and uncovering possible causes for different occurrences.
We believe that this solution is not only beneficial to the patient, but may also may be of use to the clinitians advising the patient. The device has continuous contact with the patient, thus it can provide early response if/where needed, Moreover, the system can provide useful data, that an authorized medical expert can use while prescribing a particular treatment, or even when investingating this health problem. We have started to add data-mining algorithms and methods, to uncover hidden behavioural patterns that may lead to crisis situations. Ultimately, through refining the rule systems base don human and machine learning, our approach has the potential for personalising the system according to the habits and phenotype of its user.
The system is to be incorporated in a currently developed diabetes management application for Android.

Keywords: Diabetes, Rule-Based Systems, eHealth

1 Introduction

Diabetes is a chronic disease that in 2015 affected 415 million people around the world [9]. Despite all the efforts to avoid the growth of this disease, the number of people with diabetes is continuously increasing and it is estimated to reach 642 million in 2040 [9]. Correct management of diabetes leads to a normal life, without complications. Unfortunately, if glycaemic values are not controlled, diabetes can seriously decrease the quality of life. In the worst cases, the disease will lead to amputations, blindness or heart problems. Type 1 diabetic patients (insulin dependent) require a very strict control of their disease. They should collect data on glucose levels, carbohydrate intake, insulin, exercise, stress, illness and other factors that influence glycaemia. MyDiabetes[3] is an Android mobile application that started as a way for users to register their daily data. As with the dissemination of mobile smart devices, mHealth applications became a useful tool, which even clinicians recommend [4].

For users, having their records with them at all times can be very useful. These form the basis to make decisions and react to their daily context that affects their disease management. With this is mind, we propose to add to our existing MyDiabetes application the capability to advise the user, based on medical protocols and the records made by the patient. To achieve this goal, we developed a RBS composed of medical protocols and medical approved advice, translated to logical rules, which we called ARBS. These rules and protocols are being developed and implemented with the cooperation of the endocrinology service of the S. João's Hospital, Porto, Portugal.

To adapt to patients' specific context and characteristics we aim to add data mining techniques to create new rules based on the information registered. We have done early preliminary work to explore this avenue.

We aim to provide the diabetic patient with a mobile assistant capable of constantly collecting data and then applying medical knowledge and advances in pattern analysis to generate on-time personalised, prompt feedback to improve disease management. In this paper we address the application of medical knowledge in an RBS and describe our initial approach for using data mining to improve the rule system.

2 State of Art

RBSs have already entered the health field as expert systems, namely the diabetes topic. These systems were used in order to diagnose diabetes, advise treatment regimes and even advise the user to seek particular expert knowledge e.g. Ophthalmology. In this section we will present some approaches made to tackle the challenge associated with daily management of diabetes. acknowledgements

2.1 Expert systems

The proposal by [12] is implemented using VP-Ex-pert Shell, a tool for expert system's development. The knowledge acquisition for this project was obtained through direct interviewing of medical specialists and nurses from the diabetes'

[3] Web site at `https://mydiabetes.dcc.fc.up.pt/video.php`.

field and from the study of other related scientific resources. This system was evaluated by the internees and diabetes specialists of Hasheminezhad Teaching Hospital. The users were asked particular questions about their condition and diabetes' history like e.g. "Is your blood sugar rate equal or greater than 120?". With the answers from these questions the system is able to conclude a treatment advice. This approach was tested on 30 diabetics of various types. The results were compared with the diagnosis and advice given by the specialists. This project concluded that an expert system is able to facilitate the treatment of diabetic patients as the system's consistency was approved by the internees.

Another approach to diabetes' managements using an expert system was the Diabetes Advisor project [11]. Developed using the JESS expert system shell, it was designed as an interactive, menu based, system. At the end of the consultation, the Diabetes Advisor displays a summary of the main recommendations. If the situation justifies, the system may even recommend the user to seek medical counselling. This project was found to be useful even as a prototype that still requires improvements in the user interface and accessibility.

One of the main problems of the referred approaches is the interaction between the system and the user. We believe that our approach, by being implemented on a mobile device, has the advantage of being always close and available for the user to record relevant data. A second issue is that most expert systems are passive. They do not allow learning of patient behaviours or prediction of critical situations.

2.2 Data-mining algorithms

Data-mining research has broadened different medical fields such as cancer detection [5, 7]. In the diabetes' field, data-mining is also a useful tool for monitoring and diagnosis [8].

Lee, Gatton and Lee [10] aim to provide diabetes' management for diabetic patients, taking into account the glycaemic level. This proposal considers two different approaches to the diabetes monitoring problem: a rule-based solution and a data-mining solution. The data-mining solution is obtained through K Nearest Neighbor (KNN), a classification algorithm used in data-mining and machine learning, applied to a sample set. The main goal of this project is to obtain an optimum treatment recommendation. This project is implemented as Web Services and as a Personal Digital Assistant (PDA). The results presented show that the system was able to calculate calorie intake based on the patient's condition and was able to manage meals, exercise therapy and glycaemic values.

The method applied was: to recommend eating if the glycaemic values are low; to recommend exercise, if the system detects that the glycaemic values are high; and if the high glycaemic values persist, it is recommended "taking Insulin". This prescription treatment was compared using the KNN classifier in order to obtain the optimal treatment.

After comparing the two methods, the prescription algorithm using the KNN classifier was selected as the optimum treatment method, rather than the prescription algorithm based on rules. This conclusion was based on the fact that the prescription algorithm using KNN selects the optimum treatment method using the given sample data while the prescription algorithm method based on rules has the disadvantage of not being able to select any treatment method without previous diabetes knowledge [10].

The work proposed in [10] illustrates the value of rule-based algorithms working together with data-mining. One problem not discussed in this paper is how to obtain the data to mine. In this project the data was available, but when trying to monitor an individual, it is not possible to advise without first obtaining base data from them. We believe here is where the rule-based algorithm can be truly helpful. The rule-based algorithm, grounded on medical knowledge can advise without the need of initial data. After data is recorded (after the first uses by the user), in our approach on the mobile device, it is possible for the data-mining algorithm to intervene and give more personalized advice.

2.3 Mobile applications for chronic disease self-management

Mobile devices, namely smartphones, are increasingly becoming more accessible. This reality creates a unique opportunity for developing new and better platforms for the management of chronic diseases. In this field, applications like [1] and [14] propose the inclusion of RBSs as an expert knowledge element and as a tool to advise users.

The work developed in [14] proposes a patient diary, directed for patients with Atrial Fibrillation (AF). In its base, users of this application can record data such as blood pressure or complications related to their chronic disease. Adding to this feature, the authors implemented a rule engine composed of rules extracted from the Clinical Practice Guidelines for AF for the treatment of AF given by the Canadian Cardiovascular Society and the European Society of Cardiology. Similarly to our approach the inference system and the application are separate modules that can be updated individually. This application will be tested in a large-scale clinical trial (5000 AF patients).

Addressing diabetes, the proposal from [1] portraits an architecture capable of improving self-management of chronic diseases. This architecture was implemented on the MoSHCA diabetes application for diabetes management. The system uses the data inserted to advise the user given the specific needs of the user's diabetes type. The advice consists of treatment suggestions and notifications to remember the user of tasks like insulin/drug intake or glycaemia value verification. The proposal also focus on telemonitoring/telementoring where the health professional can access the patient's information and provide counsel by contacting the patient. The reasoning behind the advice system is not clear from the proposal, and it is not clear the type of suggestions given and their knowledge base.

Both approaches propose a system very similar to ours where medical knowledge and user's data are used in order to advise. Conclusions on the usefulness of these type of systems are still theoretical as both systems together with our approach are still waiting for validation through user testing.

Other approaches by [2] and [6] are also very relevant in our study. These articles tested the impact of gamification and visual feedback, respectively, on mobile applications for diabetes management on adolescents.

The first study applied gamification techniques in order to turn tasks into game objectives. After completing an objective, the user was rewarded with music and applications from iTunes. The implementation of this technique resulted in an improvement in the frequency of blood glucose monitoring.

The second study compares a mobile visual picture-based system to register different types of records such as physical activity or photos of food eaten with a web-based short message service (SMS) intended for users to communicate with their providers when faced with diabetes management doubts. This work states that visual representations seem well adapted to the maturation of the adolescent brain. Which helps the adolescent user to link theoretical knowledge and executive functions. This study supports that systems like the mobile visual picture-based system are more adequate for adolescents.

Although visual feedback and gamification are not the focus of our approach, we take in consideration the importance of these elements. Our approach through the Android Native Application (ANA) is testing the addition of gamification elements, but its discussion is outside the scope of this article. The ARBS advices displayed to the user still consist of plain informative text. In the future the importance of visual feedback will be taken in consideration for the display of advice.

3 System Architecture

Our approach is constituted by: the ANA, which is composed by the Core, the ARBS and a SQLite Database. These components work together in order to help the users' diabetes management. We will refer to the ANA when we describe interaction with the user, i.e., referring to the user interface. The ANA receives input from the user as records and retrieves the conclusions from the ARBS as advice messages and alerts. This is mediated by the Core application, which we refer to when addressing the Android code that intermediates information transfer and handling between the ANA and the ARBS. Both the Core and the ARBS have access to the records' database, but, for now, only the Core is able to write new information. The Core is able to call the ARBS to verify the existence of new advice. This communication process can be observed in Figure 1.

We use Prolog for our RBS and YAP [3] as our Prolog engine. The connections between the Core and the ARBS and the ARBS and the database, in a fully integrated system, are possible through a YAP port to Android, the yapDroid. yapDroid is still being developed[4], so the full integrated system has not yet been tested with volunteers. An early Beta version of yapDroid was used to test the connection between the Core and the ARBS. This Beta version still lacks the elements necessary to connect to the database. The tests made were related to

[4] See `https://github.com/vscosta/yap-6.3` for the development.

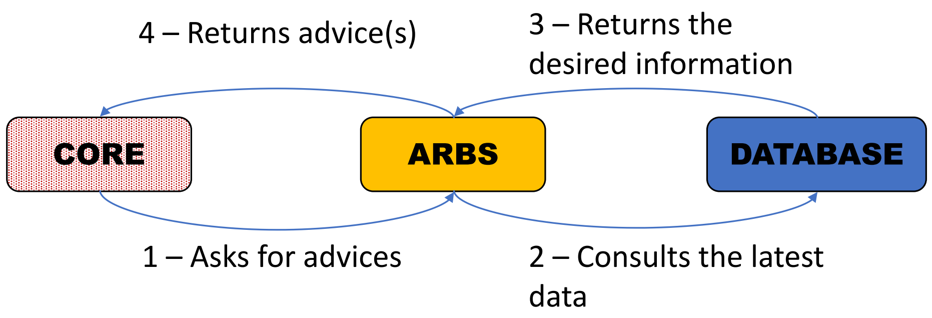
4 – Returns advice(s)
3 – Returns the desired information
CORE
ARBS
DATABASE
1 – Asks for advices
2 – Consults the latest data

Fig. 1. Example of the work flow when the ARBS needs to filter the advice list

the communication of the Core to the ARBS and the execution of the ARBS with fabricated facts that would be present on the database. The tests ran with full correctness, the system always responded with the expected answer, but to fully test the application we still need to first be able to connect to the database.

Since we still do not have the means to integrate YAP on an Android device, in order to prove our concept we developed non-fully integrated tests. For these tests we ran YAP on a computer, executing the query the Android Core would call. We then inserted on the mobile application the result obtained from YAP running on the computer. This allowed us to verify if the mobile application would behave as intended, i.e., generated the correct user interface alerts. For this, we created a test method on the mobile application that accepted the result of the ARBS (an atom) inserted as a string. The tests were successful as the system behaved as it was expected.

4 The advice rule-based system

As mentioned, the ARBS is a Prolog-based language set of rules, that analyses data, based on medical knowledge and guidelines. The main objective of this component is to receive the current situation and, based on the data from the database, return the advice defined for that situation. The implementation of the ARBS is divided in three categories, as we will discuss on the following sub-sections: system rules, register query rules and medical rules.

4.1 System rules

The *system rules* are the ARBS's core. For a medical collaborator the perfect work environment would be one where development could be done with clauses that can be read and understood as simple text. Predicates like `atom_concat/3`, while useful in building rule IDs, or rules that filter certain facts, must be avoided in medical collaboration, in order to avoid unnecessary confusion. Thus system rules include the rules that support the different sub-components of the ARBS, in a working unit that hides the underlying more technical predicates. The system rules have a specific rule called "master rule" constructed to be the *main* method of the ARBS. This is the predicate the Core uses to call upon the ARBS to verify the existence of a new advice. Included in the system rules are also other facts that define user preferences, e.g. the maximum number of advices the user wants to receive and other predicates used to sort and filter the resulting advice list; among other auxiliary facts that the master rule requires. The master rule starts by using a `findall/3` to search all the advice rules that trigger under a given situation. This situation will be further explained on the medical rules section 4.3, where the information of the current user status is crucial.

The predicate `findall/3` is used to obtain an array of triggered advice. The length of the advice array may be too long to be displayed. To avoid this, the master rule filters the array of advice, starting by ordering the array by level of urgency. If we are to only return a certain number of advice, we want to return the more urgent advice. The array filtering is dependent on a parameter given to the master rule. This parameter defines if the system wants an array of advices (`multi_advice`) or just the most urgent advice (`single_advice`).

As users gain experience using the application, some advice may become dispensable. With this in mind we allow users to block certain advice. When the user swipes the advice on the ANA, the Core inserts a new fact in the ARBS e.g. `block('advice_ 3h_meal')`, which would represent the advice that informs the user that eating every three hours is important. Blocking is not permanent, as the advice may become relevant again in the future. When the Core blocks an advice on behalf of the user, it also schedules the removal of the rules' blockage. The default duration of a blockage is, by default, one month but this value still requires further test and validation. The blockage removal is made through the Core's initiative, as the ARBS is not able to perceive time, nor contains internal schedulers.

The advice's text presented to the user is specified in different files according to the user's language. The Core, while requesting advice, informs the ARBS of what language is currently being used. The ARBS, through the system rules, then selects the corresponding advice file so that when advice terms are returned to the Core they have the chosen language. Currently the system contains English and Portuguese advice files. For easier addition of new rules to the system, the text files are composed of facts in the ARBS that will be described in section 4.3.

4.2 Register query rules

One of the most important elements in the ARBS is the connection to the database, as the main goal of the application is to know the user's current health state and advise accordingly. It is essential to be able to access the stored data. The *Register Query Rules* are Prolog facts created for accessing the database. They are the link needed to ascertain facts of the user's clinical condition such as the value of the last recorded glycaemia. In order to retrieve this value we have to first create a connection with the database; then by using the predicate 'getLastGlycaemia(Value)', shown bellow, we obtain the wanted value. These predicates use the `db_sqlite3_select/3` predicate in order to execute SQLite queries and obtain information from the database.

```
getLastGlycaemia(Value) :- db_sqlite3_select(con,'SELECT Value FROM
Reg_BloodGlucose ORDER BY DateTime LIMIT 1',[Value]).
```

These rules, used to access the contents of the DataBase (DB), are a direct use of the YAP's library capabilities of importing database tables as terms in the logical program's environment [13]. For now, the ARBS is only planned to be able to read information from the registers database. The writing task is only available for the Core component.

4.3 Medical rules

The ARBS, when called by the Core verifies if the patient recorded data fits a range of known clinical conditions. Medical Rules are clauses that represent different clinical conditions. Every one of these rules places the user in a particular situation e.g.: "the patient exercised recently (less than 2 h) and is going to take insulin". Since exercise affects the blood glucose values, the user should consider the exercise done while calculating the insulin dose. So, in this case, a rule will trigger an advice to warn the user for the need of taking precautions.

We called these rules `inRisk` given that if the rule triggers, it means the user is in risk of incurring in the situation we aim to prevent. These rules have four parameters: the record timing, the type of record, the urgency and an ID. While the first two parameters serve as a way of knowing more of the user's status, the last two parameters must be defined upon the rules' creation. The example bellow shows the skeleton of a risk rule.

```
inRisk(record_timing, record_type, urgency, id):- condition1_n.
```

There are two possible timings for the ARBS to be called: before the user inserts new records and after the user inserts new records. The timing after the user inserts information is obviously crucial, as the user inserted new data that may indicate a change to a status that may need advice. If the user, for example, inserts a flu condition, the system's reaction should be to warn the user to the increased risk of glycaemia fluctuations, which translate on a need to check glycaemic values more frequently. If the user inserts a glycaemia value under 70 (a hypoglycaemia state), the system displays specific guidelines the user needs to follow to revert the situation.

The timing 'before the insertion of new records', when the user opens the input window that allows the insertion of a certain type of record, at first does not seem to hold any meaningfulness. However, by adding information of the record's type, we can know the user's intention which is valuable information. The example shown at the beginning of this section illustrates this, "the user exercised recently" which is a simple condition defined by a register query rule, "and is going to take insulin" meaning the user is requesting the mobile application to insert a new insulin dose, but still has not inserted it. This gap between knowing the user's intention and the user's definitive insertion of a new record is when we call for the attention of the user to facts that may have been forgotten and would lead to a situation of crisis. The user, seeing the advice, has the opportunity to change or adapt the current behaviour in order to avoid a future crisis.

The `record_type` is the indication of what record is being assessed in the rule. It could be inserting `insulin`, making an `exercise`, etc.

Medical rules represent different clinical conditions with different levels of urgency. This level of urgency is reflected on the `urgency` parameter. This parameter's value may range from 1 to 10; 1 being the least urgent and 10 the most urgent advice to give. This attribute is specially important when trying to filter or sort the triggered advice. In our approach we target a variety of situations, some with particular conditions and others more abstract. While particular situations can not have conflicts between each other, more generic rules can. For example, if a rule is set to trigger at the start of a new exercise registry and has no further conditions, it will conflict with other particular rules that would also trigger in the same setting. The contrasting point between both rules is their urgency level. Abstract rules consist of generic "good advice" e.g.: "a rule that detects constant high glucose values". This rule is important since it calls the attention of the user for a fact that should be corrected. Despite its importance, this rule can never overlay a rule that advises in case of Hyperglycaemia. In other words, abstract rules are always less urgent than particular rules. In case of conflict, the particular rule is always returned as the urgency level is higher.

In its current state, the mobile application, displays only one advice in the "before timing" and the maximum of three advices at the "after timing". The presented numbers were selected, in one hand, to inform the user and, on the other hand, to not overwhelm them with too much information during a short period of time.

The connection between the analysis and the advice's text is made through the **ID** field. Every medical rule has a unique **ID**, which, when the rule is triggered, is used to search the corresponding advice in the localised language file.

Clinical conditions are represented by facts and symptoms e.g.: "Hypoglycaemia is defined by abnormally low blood glucose (blood sugar) levels, normally with values below 70 mg/dl"[5]. People with hypoglycaemia may show symptoms such as: shakiness, nervousness, anxiety, sweating, chills and clamminess. Likewise, Medical Rules are composed of conditions and symptoms. In this case every condition is a fact about the user's health state, and is represented by a Register Query Rule (RQR). If all the conditions of the Medical Rule are met, it denotes that the user is in a certain clinical condition.

Symptoms can be obtained by direct contact with the user such as shakiness or weakness/fatigue. But other symptoms such as blurred/impaired vision or headaches can only be obtained by asking the user about his current state. Either way, the system itself cannot obtain these symptoms, since it is not able to analyse its user. The solution encountered for this problem was the creation

[5] From American Diabetes Association.

of a particular interaction with the user. When a rule is to be evaluated (as per the timing mentioned previously) the system may need to ask the user if a certain symptom is present. If the response is affirmative, a new fact is added to the ARBS indicating that this symptom is present. With this new information the *medical rules* can determine the clinical condition more accurately.

In order to better understand medical rules, we are going to analyse an example. The following rule verifies if the user has a low glycaemia value, when the user has the intention of registering a new exercise:

```
inRisk(start,exercise,6,hasLowGlucoseBeforeExercise):-
     hasRecentValueLow(glucose).
```

Since exercise lowers blood glucose, it is important to take precautions before and after exercising, even more if the user already has low glycaemia values.

This medical rule has the following arguments: `start` defines the moment of intervention as the time the user enters the 'register new activity'; `exercise` defines that the system will only trigger this rule in an exercise (new) register situation; `6` is the indication of urgency[6] of this rule; `hasLowGlucoseBeforeExercise` was the chosen **ID**, in the advice's text file there is an advice with the same **ID** that will be returned if the rule is triggered. As for conditions, the system knows the user is going to insert a new exercise register (defined by the arguments), so it only needs to confirm if the user registered recently a low glycaemia value. This is evaluated by the RQR `hasRecentValueLow` with the argument `glucose`.

An overview of the system's work flow is represented in Figure 2. In this figure it is possible to see the actions from the moment the user selects a new register to the display of the advice.

The advices in the message files are not plain text. These advices are also prolog facts with particular information. The fact base constructor can be seen bellow:

```
msg(iD,[display_text, expanded_text,[advice_type, extra_params]]).
```

As previously stated each advice has a unique **ID**, used to link the rules with the advice. The other advice's attributes define its display on the mobile application. The display text is the text that first appears on the screen to the user; the expanded text is the text that is shown if the user clicks on the advice that appears on screen, which displays more relevant information about the occurring situation; the advice type and extra parameters define the type of display. Currently the system supports three types of advice display:

- Normal - the advice is a simple message with a display text and an expanded text, and does not require further action;
- Alert - the advice requires scheduling an alert to verify if the crisis is being correctly managed;
- Suggestion - the advice suggests a particular action to the user e.g. to register a record that the user has not updated in a long time.

[6] Values for urgency are not totally validated and still need medical confirmation.

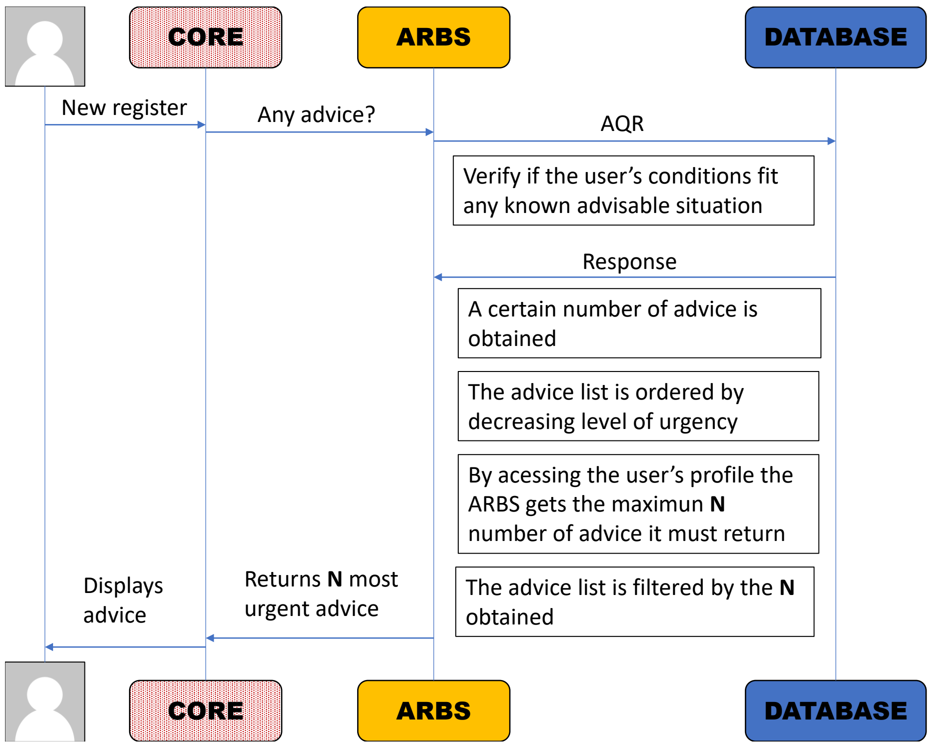
CORE
ARBS
DATABASE
New register
Any advice?
AQR
Verify if the user's conditions fit any known advisable situation
Response
A certain number of advice is obtained
The advice list is ordered by decreasing level of urgency
By acessing the user's profile the ARBS gets the maximun **N** number of advice it must return
Displays advice
Returns **N** most urgent advice
The advice list is filtered by the **N** obtained
CORE
ARBS
DATABASE

Fig. 2. Example of the work flow when the ARBS needs to filter the advice list

The extra parameters are bound to the type of register. These parameters vary from the time of scheduling of a new register or alert, to the the type of register the ANA should display in case the user accepts the suggestion. These parameters depend of the course of action the medical expert developing the rule suggests.

4.4 Crisis situations and possible causes

The ARBS functions as a guide to avoid crisis. Still, users will not always follow correctly the guidelines given. This behaviour reinforced by other external factors may lead the user to a crisis situation. The system, by itself, cannot avoid this, but it can be a source of information by enabling the user to pinpoint the reasons of a crisis. By being alerted, the user has the opportunity to rethink past actions and take measures to avoid future similar situations.

To deduce possible causes, medical knowledge is essential. The ARBS has prepared for each situation of crisis a list of possible causes, created with the tutelage of medical experts. MyDiabetes, when asked to, proceeds to evaluate each one of the possible causes for the occurred crisis. After this evaluation, the system searches for the best advice for the conclusions obtained. These possible causes are defined in the medical rules. To better understand these particular rules, one example is represented bellow:

```
possibleCause(hypoglycaemia, [(exercise,'+'), (meal,'-'),
(insulin, '+')]).
```

This particular rule stores the possible causes for a hypoglycaemia. It is constituted by a crisis type, in this case hypoglycaemia and a list of possible causes. The elements of the possible cause list are composed of a register type and a plus or minus symbol. These symbols represent the existence or absence of a recent register. In the case shown, the possible causes for a hypoglycaemia are: to have recently performed exercise, the absence of recent meals or a recent insulin intake.

The rule that verifies the existence or absence of a register is the RQR `hasRecent(RegisterType)`. The verification process of each of the conditions by the system is not visible to the medical expert, since these are system details. Each one of these actions, represented by the possible cause elements, affects the glycaemia levels in a different way so, all of them may have influenced the hypoglycaemia crisis situation. The possible outcomes are that none of these possible causes happen or one or more of these happen. Each different conclusion has an advice text informing from these possible causes what led to this current situation.

5 Data-mining integration

Data-mining is a useful tool able to uncover hidden information within data. In our approach we start to apply data-mining algorithms in order to uncover usage patterns that led to crisis situations. Our objective was to create new rules, given the patterns found, to alert the user and avoid future crisis. This process would also individualize the ARBS to the user's needs.

To achieve our goals, one of the main requirements for the use of data-mining is to have a large amount of data. Due to the lack of online relevant databases and information this is a hard requirement to fulfil. In order to achieve our study goals, through a partnership with the endocrinology service of the S. João's Hospital, we started collecting volunteer's information. The application used in this test was the base MyDiabetes application. The ARBS and the data-mining components were not embedded in the test application. At this point we have thirty one volunteers registered and from these eight sent information. From the eight, five sent information regularly.

With the data collected we obtained association rules and developed one bayesian network for each volunteer. These rules point to interesting characteristics such as "At Thursdays and Fridays afternoon you usually have hyperglycaemias". These facts could later be used by the user or even the medical expert accompanying the user, to pinpoint the origin of the problem and correct it. Note that the conclusions are still very crude with no explanation associated, and mostly based on statistic gathering on a small data set of registers. The bayesian networks obtained can be used to predict the probability of an user to have hypoglycaemia or hyperglycaemia given certain conditions. In the future we want to connect the ARBS to this data-mining component. The patterns uncovered by the data-mining component can be translated to "advice logical rules" and be used to call the attention of the user to a particular pattern that is developing.

If we take one of the conclusions obtained for example "At Wednesday in the afternoon your usual meal results in high glycaemic values." we can create a Medical Rule as:

```
inRisk(start, carbs,9,dMrule_wed_meal)
        :- isToday(wednesday, afternoon).
```

This rule triggers at the **start** of a new **carbohydrate** register when the current day is **wednesday** in the **afternoon**. By triggering the rule returns an ID with the value **dMrule_wed_meal** that will be converted to an advice. This advice warns the user to the conclusions from the data-mining component. The user, knowing this fact, is able to take precautions and avoid hyperglycaemia. This rule's ID would be connected to an advice that would warn the user for the need of taking precautions. If the user corrects this behaviour, the pattern disappears and the rule would be removed.

6 Conclusion

Education is a crucial element in diabetes management. Considering this, we believe that diabetes requires lucid care in order to avoid crisis. Our approach pre-emptively alerts the user to the risks of certain actions (recalling the doctor's advises), this way, avoiding possible crisis and educating the user. When a predicament does occur, our approach is also able to inform and guide the user. The advice displayed is the result of a medical rule that was triggered. This rule represents a given situation that necessarily occurred. From the advice given it is possible to know the user's situation, for a medical expert this is relevant information that may lead to changes in the user's current treatment. This may help pinpoint problems in the current behaviour, by analysing advices triggered. More so, it may provide new advices for the patient and/or for the system. By using data-mining we will be able to individualise our advice, adapting the system to our user's needs.

Our system still requires testing to be fully validated. As soon as the YAP port is available, we intend to test this new feature with our current volunteers. We believe our concept has usefulness and can bring a new light to the diabetes management, not only helping the user with advices and guidance, but also by hinting the health expert to flaws that need to be addressed.

By integrating both the ARBS and the data-mining component we expect to receive more data from the users. The more users insert records, the better the system can help them, and the more the system helps the users, the more relevant the system becomes.

Acknowledgement

This article is a result of the project NanoSTIMA Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics, NORTE-01-0145-FEDER-000016, supported by Norte Portugal Regional Operational Programme (NORTE 2020), through Portugal 2020 and the European Regional Development Fund.

References

1. Ambroise, N., Boussonnie, S., Eckmann, A.: A smartphone application for chronic disease self-management. Proceedings of the 1st Conference on Mobile and Information Technologies in Medicine. (2013), `http://www.mobmed.org/download/proceedings2013/mobilemed2013_paper_11.pdf`
2. Cafazzo, A.J., Casselman, M., Hamming, N., Katzman, K.D., Palmert, R.M.: Design of an mhealth app for the self-management of adolescent type 1 diabetes: A pilot study. J Med Internet Res 14(3), e70 (May 2012), `http://www.jmir.org/2012/3/e70/`
3. Costa, V.S., Rocha, R., Damas, L.: The YAP prolog system. TPLP 12(1-2), 5–34 (2012)
4. Demidowich, A.P., Lu, K., Tamler, R., Bloomgarden, Z.: An evaluation of diabetes self-management applications for android smartphones. Journal of Telemedicine and Telecare 18 (May 2012)
5. Ferreira, P., Fonseca, N.A., Dutra, I., Woods, R., Burnside, E.: Predicting malignancy from mammography findings and image-guided core biopsies. Int. J. Data

Min. Bioinformatics 11(3), 257–276 (Feb 2015), http://dx.doi.org/10.1504/IJDMB.2015.067319

6. Frøisland, H.D., Årsand, E., Skårderud, F.: Improving diabetes care for young people with type 1 diabetes through visual learning on mobile phones: Mixed-methods study. J Med Internet Res 14(4), e111 (Aug 2012), http://www.jmir.org/2012/4/e111/
7. Gorunescu, F.: Data mining techniques in computer-aided diagnosis: Non-invasive cancer detection. World Academy of Science, Engineering and Technology 25 (2007)
8. Han, L., Luo, S., Yu, J., Pan, L., Chen, S.: Rule extraction from support vector machines using ensemble learning approach: An application for diagnosis of diabetes. IEEE Journal of Biomedical and Health Informatics 19, 728–734 (May 2014)
9. International Diabetes Federation: IDF diabetes atlas (2015), www.diabetesatlas.org
10. Lee, M., Gatton, T.M., Lee, K.K.: A monitoring and advisory system for diabetes patient management using a rule-based method and knn. Sensors (Basel) 10, 3934–3953 (April 2010)
11. Mbogho, A., Dave, J., Makhubele, K.: Diabetes advisor–a medical expert system for diabetes management. In: International Conference on e-Infrastructure and e-Services for Developing Countries. pp. 140–144. Springer (2013)
12. Seyedeh Talayeh Tabibi, Tawfik Saeed Zaki, Y.A.: Developing an expert system for diabetics treatment advices. International Journal of Hospital Research 2, 155–162 (September 2013)
13. Soares, T., Ferreira, M., Rocha, R.: The MYDDAS programmer's manual. Tech. rep., Faculdade de Ciências da Universidade do Porto (2005)
14. Woensel, W.V., Roy, P.C., Abidi, S.R., Abidi, S.S.: A mobile & intelligent patient diary for chronic disease self-management. MEDINFO 2015: eHealth-enabled Health. pp. 118 – 122 (2015)