Université de Montréal

An intelligent real-time help system for clinical reasoning in virtual reality environment based on emotional analysis

Par

Qiang Ye

Département d'informatique et de recherche opérationnelle, Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de de Maîtrise ès sciences (M. Sc.) en Maîtrise en informatique, option intelligence artificielle

Juin 2020

Université de Montréal

Unité académique : Département d'informatique et de recherche opérationnelle, Faculté des arts et des sciences

---

Ce mémoire intitulé

An intelligent real-time help system for clinical reasoning in virtual reality environment based on emotional analysis

Présenté par

Qiang Ye

A été évalué(e) par un jury composé des personnes suivantes

François Major
Président-rapporteur

Claude Frasson
Directeur de recherche

Esma Aimeur
Membre du jury

Résumé

Le raisonnement clinique est l'une des compétences les plus importantes de la pratique médicale. Hypocrates est une plateforme logicielle d'évaluation médicale et d'analyse émotionnelle construite sur un environnement de réalité virtuelle. Grâce à cette plateforme, les étudiants en médecine peuvent évaluer leurs connaissances médicales par des cas cliniques virtuels bien conçus à partir d'une base de données de cas. Pendant le processus d'évaluation, les signaux d'électroencéphalogramme sont collectés simultanément pour évaluer l'état émotionnel de l'élève, ce qui aide les chercheurs à étudier le changement émotionnel de l'élève pendant tout le processus d'évaluation. Des études antérieures montrent que le maintien d'une émotion paisible et positive est nécessaire à de bonne performance d'évaluation. Pour maintenir un état émotionnel positif, une possibilité est d'aider les élèves de manière à éviter des émotions négatives (frustration, stress, confusion, etc.) qui peuvent découler des erreurs d'évaluation.

Dans cette recherche, nous avons étudié, conçu et développé un système d'aide en temps réel et l'a intégré dans Hypocrates pour former sa prochaine version: Hypocrates +. Le système d'aide est intelligent car il fournit un contenu d'aide personnalisé et hautement associé lorsque la plate-forme estime qu'un contenu d'aide est nécessaire pour maintenir le statut émotionnel paisible et positif d'un élève. Le système d'aide est en temps réel car il a un délai de réponse très court afin que l'étudiant puisse obtenir très rapidement des connaissances médicales utiles. Le contenu de l'aide est généré à partir d'Internet, ou plus précisément, à partir de pages Wiki en ligne, pour que le contenu de l'aide soit toujours à jour.

Le langage C # et Visual Studio IDE ont été utilisés pour développer le système d'aide en temps réel. Une application console fonctionne comme un serveur fournissant des services à ses clients et constitue une partie de la plateforme Hypocrates+. Des techniques telles que la recherche d'informations, l'intelligence artificielle, la programmation réseau UDP ont été largement utilisées pour aider au développement d'un serveur intelligent. Avec le système d'aide en temps réel souhaité intégré, Hypocrates+ est passé à une plateforme virtuelle de formation médicale au lieu

d'une simple plateforme d'évaluation, qui devient de plus en plus populaire dans la formation médicale moderne.

Des tests et des expériences ont été effectués sur le système d'aide en temps réel et Hypocrates+ pour étudier la qualité et l'utilité du contenu d'aide généré et le temps de réponse. La réponse est très rapide avec un temps de réponse moyen de 1,5 seconde. Les résultats de l'analyse émotionnelle montrent que le contenu d'aide a réduit l'émotion négative de 4 participants sur 5. Nous concluons qu'un système d'aide en temps réel avec une bonne qualité de contenu d'aide enrichit les fonctionnalités d'une plate-forme de formation médicale en réalité virtuelle.

Mots-clés: intelligence artificielle, analyse émotionnelle, formation médicale, temps réel, réalité virtuelle

Abstract

Clinical reasoning is one of the most important skills in medical practice. Hypocrates is a medical evaluation and emotional analysis software platform built on a virtual reality. By this platform, medical students can evaluate their medical knowledge by well-designed virtual clinical cases from a case database. During the process of the evaluation, electroencephalogram signals are collected simultaneously for evaluating the emotional status of the student, which helps researchers study the emotional change of the student during the whole evaluation process. Previous studies showed that maintaining a peaceful and positive emotion is necessary for good performance, and one possible way to maintain such emotional status is to help avoid negative emotions (frustration, stress, confusion, etc.) that can arise by errors during the evaluation.

In this research, we studied, designed and developed a real-time help system and integrated it into Hypocrates to form its next version: Hypocrates+. The help system is intelligent as it provides personalized and high related help content when the whole platform believes such help content is necessary to maintain a student's peaceful and positive emotion status. The help system is real-time as it has a very shot delay of response so that the student can obtain useful medical knowledge very quickly. The help content is generated from internet, or more precisely, from online Wiki pages, to keep the help content is always up to date.

C# language and Visual Studio IDE were used to develop the real-time help system: a console application functions as a server providing services to its clients: other part of Hypocrates+ platform. Techniques like information retrieval, artificial intelligence, UDP network programming were widely involved to help developing the intelligent server. With the desired real-time help system integrated, Hypocrates+ upgraded to a virtual medical education platform instead of just an evaluation one, which is becoming more and more popular in modern medical education.

Tests and Experiments were performed on the real-time help system and Hypocrates+ to investigate the quality and usefulness of the generated help content and the response time. Results show that the content quality is quite good. The response is very quick with average response time of 1.5 seconds. The results of emotion analysis show that help content reduced 4

of 5 participants' negative emotion. We conclude that the real-time help system with good quality of help content enriches the functionality of a virtual reality medical education platform and it probably helps medical students reduce negative emotion during clinical reasoning.

Keywords: artificial intelligence, emotional analysis, medical education, real-time, virtual reality

Table des matières

Résumé 5

Abstract 7

Table des matières 9

Liste des tableaux 13

Liste des figures 15

Liste des sigles et abréviations 17

Remerciements 19

Chapitre 1 – Introduction 21

1.1 Review of medical education 22

1.2 Clinical reasoning 23

1.3 Virtual reality technology in medical education 24

1.4 Hypocrates 25

1.5 A real-time help system: upgrading to Hypocrates+ 27

Chapitre 2 – Internet Searching 28

2.1 Internet search engine 28

2.1.1 Crawling 28

2.1.2 Indexing 31

2.1.3 Creating ranked results 33

2.2 Wiki API for OpenSearch 35

2.3 Internet searching to real-time help system in Hypocrates+ 36

Chapitre 3 – Hypocrates+ 39

3.1 Architecture of Hypocrates+ 39

3.1.1 Clinical Case Database Component 40
3.1.2 Case Manager Component 40
3.1.3 Report and Log Component 40
3.1.4 Virtual Environment Component 41
3.1.5 Improved Intelligent Agent 42
3.1.6 Real-time Help System 42
3.2 Artificial Intelligence in Hypocrates+ 42
Chapitre 4 – Real-time Help System for Hypocrates+ 45
4.1 Architecture of RTHS 45
4.1.1 Message Mechanism Between RTHS and its Clients 46
4.1.2 Intelligent Information Retrieval Component 50
4.2 Key Implementation of RTHS 54
4.2.1 WikiSearchServer 55
4.2.2 ClientApp 57
4.2.3 WikiAPI 58
4.2.4 HtmlAgilityPack 59
4.3 Tests 59
4.3.1 Accuracy Test 60
4.3.2 Response Time Test 61
Chapitre 5 – Experimental Setup and Results 63
5.1 Objective of the experiment 63
5.2 Criteria for inclusion and exclusion 64
5.3 Scenario of the experiment 64
5.4 Results and discussion 66

5.4.1 Participant profile 66
5.4.2 Accuracy and usefulness of help information 67
5.4.3 Negative emotion analysis 68
Chapitre 6 – Conclusion 71
6.1 The development of real-time help system 72
6.2 Benefits and disadvantages of real-time help system for Hypocrates+ 73
Références bibliographiques 75
Annexes 79
Annex1 An XML segment of a clinical case 79
Annex2 Recruitment poster for experiments 82
Annex3 Paper accepted by 2nd BFAL international conference 83
Annex4 Detailed data for negative emotion analysis 91

Liste des tableaux

Table 1.1 Main Parameters of Wiki API for OpenSearch 35
Table 4.1 The common reason of lower quality response information 61
Table 4.2 Basic Statistics of Response Time (s) 61
Table 5.1 Basic information of 5 participants 67
Table 5.2 The accuracy and usefulness of the help information 67
Table 5.3 Average frustration level of the participants 69

Liste des figures

Figure 1.1 Hypocrates architecture 25
Figure 1.2 Hypocrates virtual environment and its information panels 26
Figure 3.1 Architecture of Hypocrates+ 39
Figure 3.2 Two typical case scenarios in virtual environment 41
Figure 4.1 General architecture of RTHS and the communication with its clients 46
Figure 4.2 Definition and fields of the request and response messages 46
Figure 4.3 Data structure of message in message queue 49
Figure 4.4 Message flow In Hypocrates+ 49
Figure 4.5 Mechanism of intelligent information retrieval component 50
Figure 4.6 Structure of the solution and code snapshot in Main function 54
Figure 4.7 A Windows Form UI client and RTHS console server. 57
Figure 5.1 Histogram of mean frustration level before and after the help of 5 participants 69

Liste des sigles et abréviations

EEG : electroencephalogram

RNN: recurrent neural network

RTHS: real-time help system

IIRC: intelligent information retrieval component (of RTHS)

UDP: user datagram protocol

NLU: natural language understanding

DLL: dynamic link library

HTML: hypertext markup language

DOM: document object model

XML: extensible markup language

XPATH: XML path language

XLST : extensible stylesheet language transformations

Remerciements

I wish to express my sincere appreciation to my supervisor, Professor Claude Frasson, who guided and encouraged me to be professional and do the right thing during the research. Without his persistent help, the goal of this project would not have been realized.

I wish to show my gratitude to Hamdi Ben Abdessalem, a PhD project member who provided me many good suggestion and direction on my research. I would also like to thank Marwa Boukadida, an assistant researcher in the group who also spent lots of time in the project. I deeply enjoyed the teamwork and will remember all the hardworking yet joyful time with you.

I would also like to thank all the friends I met in the Heron lab. Daily conversations and discussions with you all have greatly inspired my research work and also brought me much help in life, which I really appreciated.

I must thank my wife and two daughters, and their support has enabled me to devote myself to my research work and to enjoy it as well.

I thank NSERC-CRD (National Science and Engineering Research Council- Cooperative Research Development) and BMU, two institutions, that funded this research.

Chapitre 1 – Introduction

Hypocrates+ is the next generation software system of Hypocrates, combining virtual reality and real-time help technologies to help junior medical students and residents better learn medical knowledge by well designed virtual electronical clinical cases. Compared with its old version: Hypocrates (Ben Abdessalem and Frasson 2017), Hypocrates+ imported a real-time help system to quickly help to correct learners' mistakes when they get medical training by Hypocrates+.

Medical education includes two aspects: the theoretical education in schools and the clinical practice in clinics or hospitals during most of its four stages: premedical, undergraduate, postgraduate, and continuing education. Hypocrates focuses on the theoretical education while providing virtual clinical cases to help students strengthen their knowledge so that they can be more comfortable and confident with their practice in the future. In the stage of theoretical education, medical students understand a disease mainly by memorizing the description of causes, symptoms and signs, diagnosis criteria, treatment of the disease from lectures and textbooks. Although some new pieces of knowledge could be inferred by other known ones, there are still many knowledges need to be memorized before they can be correctly used to solve a real clinical problem. Therefore, it's a huge challenge for a student to memorize all the knowledge correctly in very short period just by reading books.

It is proven that vivid visual information such as animation, three-dimensional objects, compared with images or plain texts, are more easily to be accepted by people, and can last very long in memory (Grady et al. 1998). Vivid visual information usually excites much more area of human brains simultaneously and can be more easily evoked by other information because of such wider connection (DeYoe and Raut 2014). Repetition is also a good method for people to learn and master a knowledge or a skill (Tabibian et al. 2019). By repetition, transient memory will be converted to persistent memory by strengthen related inter-neuron connection in human brain (Majerus 2013). Once a permanent memory is formed, it could last years or even the whole life of a person (Thompson and Kim 1996). As permanent memory lasts very long time, it is very important for a person to have a correct knowledge in brain, so is to correct a wrong knowledge

as early as possible. One of the most efficient ways to correct a wrong knowledge is to follow a person's study process and provide real-time help to guide the person when a mistake is made. Researches found that real-time help system (Dyer, Swartzlander, and Gugliucci 2018) can reduce the times of making mistakes, promote building correct knowledge map in brain, and increase the inference ability of a person.

Hypocrates, a virtual medical platform, was developed earlier to provided junior medical students opportunities to practice and evaluate their medical knowledge on virtual clinical cases, but without a real-time help system. Because of the importance of a real-time help system to virtual medical education. This thesis focused on how to design, develop and integrate a real-time help system to the existed Hypocrates to form the Hypocrates+.

1.1 Review of medical education

It is usually considered that medical education derives from ancient Greek where the practice of observation and reasoning regarding disease was first taken place and the famous Hippocrates' oath was proposed (Fulton 1953). However, modern medical education was established no early than mid-19th century (Fuller 1906). In 1910, the Carnegie Foundation for the Advancement of Teaching published a report by the educator Abraham Flexner (Stahnisch and Verhoef 2012). In this report, he pointed out that "medical education actually is a form of education rather than a mysterious process of professional initiation or apprenticeship". As such, medical education needs an academic staff, laboratories, libraries, teaching rooms, and teaching hospitals. From then on, formal medical education eventually developed into a process that involved four generally recognized stages: premedical, undergraduate, postgraduate, and continuing education.

In most countries, the premedical courses include physics, chemistry, and biology. These are required as the basis of subsequently courses in anatomy, physiology, biochemistry, and pharmacology with precision and economy of time to students prepared in scientific method and content. During undergraduate education, medical schools usually begin their work with the study of the structure of the body and its formation: anatomy, histology, and embryology. Concurrently, or soon thereafter, come some functional studies like physiology, pharmacology,

biochemistry, pathological anatomy, bacteriology, immunology, Usually, medical students will spend two or more clinical years of an effective curriculum learning these subjects in classrooms and laboratories. Clinical study begins with general medicine and surgery and then expands to obstetrics and gynecology, pediatrics, ophthalmology, psychiatry and other topics. During this period, students will also have opportunities to go to the clinics and hospitals to observe how certified and experienced doctors perform their clinical practice so that they gain some intuitions about clinical practice. On completion of medical school, physicians usually seek graduate training and experience in a hospital under the supervision of competent clinicians and other teachers. Generally, there is not a clear boundary between undergraduate and graduate education. In most countries, a medical license is required for a physician or a surgeon to perform clinical practice independently. During the long time of clinical practice, a continuing education like conferences, short training on a specific topic etc., is periodically provided to physicians and surgeons to keep their medical knowledge up to date.

1.2 Clinical reasoning

Clinical reasoning (Gruppen 2017; Khin-Htun and Kushairi 2019), also known as clinical judgement, is the process by which clinicians collect symptoms and signs of a patient, understand the patient's medical situation or problem, make diagnosis, plan and implement appropriate medical interventions, evaluate outcomes, and learning from this entire process.

Usually, the clinical reasoning process includes the following four main phases:

- Observe and collect symptoms, signs, and other clinical findings of a patient by talking with the patient and some basic physical examination.
- Analyze collected information and make appropriate diagnosis, treatment, and prevention plan.
- Deliver the above plan efficiently and accurately.
- Evaluate the plan's outcomes, analyze its effectiveness and side effect on this specific patient, and improve the plan for the patient and for future patients.

In real clinical practice, the fourth phase of clinical reasoning process is as equal important as the others, as it provides opportunities to rethink the whole process, find better solution, and refine scientific questions for further research, which is an engine to promote medical science. However, For medical students, this phase is not as important as in real clinical practice because students are in a training process where the cases usually are well designed and with definite results, which means students have seldom opportunities to observe the outcomes of patients for evaluating and reflecting if they are not facing real patients.

In a nutshell, medical professionals use clinical reasoning to consider the various aspects of patient care and make a relevant and appropriate decision aimed at prevention, diagnosis, and treatment of a patient's problem: a critical aspect of strong clinical skills and quality care. Clinical reasoning is often considered the most important aspect of a clinician's skill set as it determines the quality of a patient's health care. Because of the importance and the long-time process of achieving a good clinical reasoning skill set, experienced clinicians, medical education experts, and developers designed and developed many classic clinical cases for training medical students to build a good clinical reasoning skill set (Braun et al. 2018). No doubt, information technology is making these cases more and more lively and realistic than just written as texts on paper (Crowe et al. 2011).

1.3 Virtual reality technology in medical education

Significant progresses have been made in how to let medical students gain their medical knowledge more effectively and correctly. Medical education experts innovated the teaching materials, from the traditional textbooks to multimedia resources where more information can be obtained by sounds, images and videos instead of texts on paper. Along with the update of the teaching materials, new teaching and learning styles have also been proposed. Student-centered learning (Dong et al. 2019) becomes a useful supplement of traditional teacher-centered teaching; problem-based learning(PBL) (Yew and Goh 2016) and case-based learning (McLean 2016) are two new approaches for medical students and residents to gain their medical knowledge in more connecting scenarios.

As the style of medical education changes and the development of the virtual reality (Bohil, Alicea, and Biocca 2011; Dyer, Swartzlander, and Gugliucci 2018) technology, case or problem based virtual reality medical training system becomes very popular. Virtual reality technology offers new opportunities for medical educator to realistically simulate a wide range of clinical situations; meanwhile, it provides opportunities for medical students and residents to make their beginner mistakes in the virtual environment, not on real patients. In 2018, Queen's University launched Canada's first virtual reality medical training center (Queen's University 2018), partnered with virtual reality innovators *SimforHealth* and *HTC VIVE*. Reports said that "the training center allows medical students and residents to gain experience caring for patients in a realistic but completely safe environment. "

1.4 Hypocrates

From 2015, our laboratory (Higher Educational Research ON emotional, HERON) started to research and develop case-based virtual reality medical education system: Hypocrates (Ben Abdessalem and Frasson 2017). In Hypocrates, clinical cases in text and image were designed by medical educators and then transformed into virtual reality environments powered by Unity 3D game engine. Hypocrates system contains three main modules: a virtual environment, a case manager and an intelligent agent (Figure 1.1).

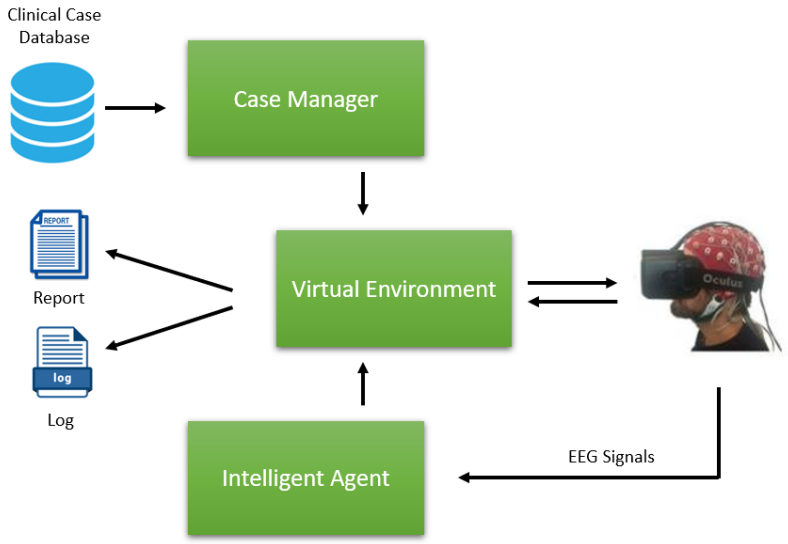

Figure 1.1 Hypocrates architecture

Hypocrates is built of three main components, indicated by three green boxes in the figure. The case manager chooses clinical cases from case database, produces a collection of cases, and displays the cases in the virtual environment for interacting with a platform user. The virtual environment component is responsible for providing an immersive virtual environment to platform user. The intelligent agent component collects the electroencephalogram (EEG) signal of the user and provide the emotional status of the user in the environment simultaneously. Once an interaction is completed, reports and logs will be saved.

During a medical student's interaction with the system, the system tracks the student's emotional state by collecting and analyze electroencephalogram (EEG) signals in real-time, tries to change the emotional state by providing some designed hints based on the interaction, and analyzes the student's subsequent reactions after she/he makes mistakes in clinical reasoning (Figure 1.2).

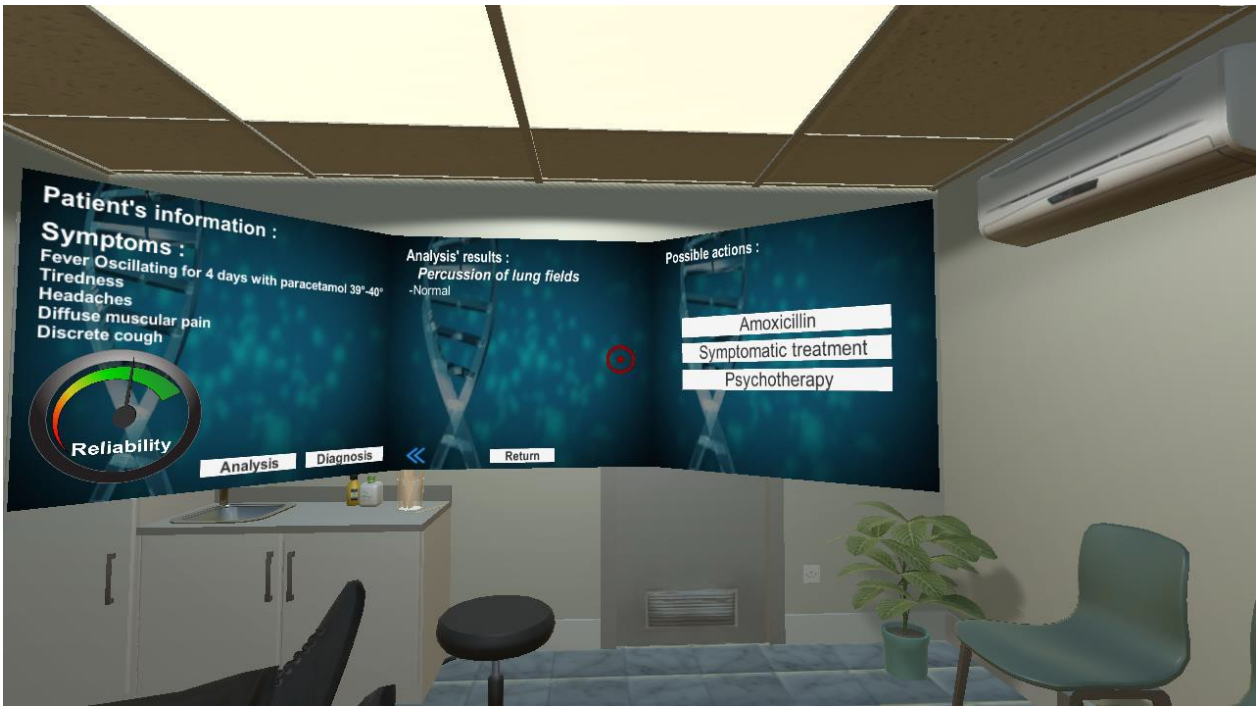

Figure 1.2 Hypocrates virtual environment and its information panels

In a virtual clinical environment, case information and questions are displayed in three huge information panels. The left panel is used to display the patient's information, a gauge of reliability indicating the emotional status of the platform user, and "Analysis" "Diagnosis" buttons. Patient's information in this panel is read-only, and there is no question in this panel. By clicking one of the buttons, middle or right panel will be shown. In these two panels, a user can interact and provide feedbacks to the questions displayed. In the middle panel, several analytical declarations about the case are listed; several of them are

true, some are false. The user is asked to choose the true one(s). In the right panel, the user is first asked to choose one best diagnosis of the case and then to choose the correct items in the treatment plan.

1.5 A real-time help system: upgrading to Hypocrates+

This research focuses on developing and integrating a real-time help system to our existing Hypocrates platform to build Hypocrates+: an intelligent virtual reality based medical education platform, so that the whole platform can provide wider range of virtual medical education instead of medical evaluation only. The main mission of the real-time help system is to provide highly related medical information to the platform user: a medical student, when the intelligent agent component in the platform considers the emotional status of the user is no longer peaceful when mistakes are made during the clinical reasoning with virtual medical cases and a real-time help is required.

The real-time help system functions as a server, providing service to its client: intelligent agent of the platform. The client asks the real-time help system for a help request, and it communicates with the real-time help system by network communication. Once receiving a help request, the real-time help system first searches the online Wikipedia web pages, analyze each suggested page by Wiki search engine, generate a short response, and finally sends the information back to the client who asked for it.

Chapitre 2 – Internet Searching

The main purpose of the real-time help system for Hypocrates+ is to rapidly provide accurate and useful information to its users from Wikipedia. From this perspective, the real-time help system provides an internet searching service. In this chapter, we review the techniques widely used in internet searching fields and the Wikipedia web search engine.

2.1 Internet search engine

Internet search exists everywhere. When people search on internet, they are using the service powered by an internet search engine. An internet search engine is an online tool that searches for results in a database based on a search query indirectly generated by an internet user. Usually, the search results are a list with ranked websites that match with the search query. The result list is generally called the search engine results page (SERP) (Clarke and Clarke 2014).

Although there are many search engines on the internet, the search mechanism of how these search engines work is almost the same. Generally, the search process includes three steps: scour the internet for contents, store and organize snapshots of the contents found, and provide a list of snapshots that the engine think best matches a user's query. These three steps are called: crawling, indexing, and creating ranked results, respectively.

2.1.1 Crawling

Crawling, the first step for building a search engine, is the process of visiting internet contents by a crawler or several crawlers. A crawler is an internet program also often called an internet bot. It systematically scans websites on the world wide web starting with a list of seed URLs to visit. When a crawler visits a Web page, it detects all the hyperlinks within that page and may push these links into the list of URLs based on a certain policy for future visit. A policy defines whether a hyperlink is going to be pushed to or removed from the URLs list and what the priority of a hyperlink in the URLs list. The URLs list, like a priority queue, will be dynamically updated under a policy: visited URLs are removed, and new related URLs are imported with a certain priority. In

many cases, an archive process is performed when a crawler visits each URL so that a snapshot of that page can be stored, and there is a repository to manage the collection of the snapshots. Usually, the repository only stores the most recent version of HTML pages.

There are many challenges (Castillo 2005) for a crawler to visit the internet in an efficient and rapid way. Crawling on the whole internet is a process that costs a lot time, meaning that within a given period, a crawler can only visit limited web pages and download a limited number of snapshots; some visited URLs could be pushed again to the list; some web pages are auto generated by users' different combination of HTTP GET parameters, which increase the difficulty of crawling. All these challenges indicate that at each step, a crawler must carefully choose what is the next page to visit so that it can visit expected amount of web pages in a limited time. Following crawling policies were proposed to address these challenges:

Selection policy

There is an extremely huge amount of web pages on internet, and the number is still increasing everyday. Even the most powerful search engine in the world can only cover a restricted portion of the whole internet pages. Steve Lawrence and C. Lee Giles (Lawrence and Giles 1999) showed that no search engine could visit more than 16% of the Web in 1999. Therefore, it is very important and necessary for a search engine to select pages that are most related to users' interest and visit them. One practical approach is to prioritize Web pages and give a metric of importance for them. The factors that influence the importance of a page include its intrinsic quality, popularity in terms of links. Based on this, several methods such as restricting followed links, URL normalization, path-ascending crawling, focused crawling, field-focused crawler, semantic focused crawler were developed to improve a crawler's page selection.

Re-visit policy

Many web pages are updated in irregular period by huge amount of web developers, which means when visiting a web page suggested by a search engine based on a former web content, the users may not always find the information needed because of the update or removal of that web page. In this case, the crawler needs to revisit that page so that it can provide better suggestions for users, or it removes the snapshot of that web page from repository if that page

can't be accessed anymore. Freshness and age are two measures developed to help designing a re-visit policy for a crawler.

Freshness is to measure whether the local copy of a web page is accurate or not. It is a binary value that will be set to 1 if the page is the same as the local copy at a specified time, and 0 otherwise. Age is another measure that indicates how outdated a local copy is. The age of a page p in the repository, at time t is set to 0 if p is not modified at time t and set to "t-modification time of p" otherwise. The objective of the crawler is to keep the average freshness of pages in its collection as high as possible, or to keep the average age of pages as low as possible.

Politeness policy

Crawlers may harm the general network community because they may require considerable bandwidth during a long period of time. Some web sites may receive requests with high frequency from a crawler, some may even crash due to poor coded crawlers. To address issues like this, the robot exclusion protocol was proposed for web administrators to judge which parts of their web services is not accessible for crawlers. Some other standards were also proposed to decide the interval between two consecutive connections so that a crawler can not visit the same website in a very high frequency. These intervals could be 10 – 15 seconds (Cho 2003), or even less than 1 second (Dill et al. 2002). A famous adaptive politeness policy indicates that if it took *t* seconds to download a resource from a specified sever, the crawler waits for 10 x *t* seconds (Heydon and Najork 1999) before downloading other resources from that server.

Parallelization policy

Crawling can be parallel. A parallel crawler can run multiple processes simultaneously. Usually, the purpose of utilising a parallel crawler is to maximize the speed of download while minimizing the overhead from parallelization and to avoid a redundant download of the same page. Cho and Garcia-Molina (Cho and Garcia-Molina 2002) studied two types of policies: dynamic assignment and static assignment. Dynamic assignment uses a central server to assign new URLs to different crawler processes, which allows the central server to dynamically balance the workload of each crawler process. Static assignment uses a hashing function to map a URL to a number that represents the index of a crawler process. No matter which assignment is applied, an effective

assignment should have a balance property which means each crawling process should get approximately the same number of hosts, a contra-variance property which means the number of hosts assigned to each crawling process should decrease if the number of crawling processes increases, and a dynamic editable property which means the assignment should be able to add and remove a crawling process dynamically.

2.1.2 Indexing

Resources extracted from Web pages vary a lot. They can be plain texts, HTML pages, PDF documents, and of other formats. The purpose of Indexing is to create a standard representation of the documents for further searching and ranking. In theory, well-designed features that can differentiate one web page from others is good for indexing. Many techniques were developed to find features of web pages, such as bag-of-word, TF-IDF model, advanced vector model, etc.

Bag-of-word

A web page talks something: academic study or everyday life, happy or sad, news or history. One method to categorize web pages is to classify it based on sentiment or to verify whether it is focused on specific subjects. In this way, we can represent a web page by a bag of words, as if it were an unordered set of words, while ignoring their original position in the page.

Bag-of-words model (Harris 1954) relies on the term frequency, defined as the number of times that a word occurs in a given document. Term frequency information is very important since it indicates the topic of a document in some respects, and it is also useful for engineers to find additional features such as the number of positive lexicon words.

Most Web search engines use keywords and metadata (Zhang and Dimitroff 2005) to provide a more useful vocabulary for Internet or onsite searching. In metadata web indexing, keywords or phrases are assigned to web pages or web sites within a metadata tag (or "meta-tag") field, by which the web page or web site can be easily retrieved by a search engine that is customized to search the keywords field.

TF-IDF model

TF-IDF algorithm (Korfhage 1997) is by far the dominant method of weighting co-occurrence matrices in natural language processing, especially in information retrieval. For a document collection *D* which has *N* documents, the TF-IDF weight $W_{t,d}$ for a term *t* in a document *d* is calculated as the product of the term frequency and the inverse document frequency:

$$W_{t,d} = tf_{t,d} \times idf_t$$

where term frequency $tf_{t,d}$, meaning the frequency of a term *t* in the document *d*, and one of definitions of the inverse document frequency is defined by:

$$\mathrm{idf_t} = \log\left(\frac{N}{1 + d_t}\right)$$

Where N is the total number of the documents, and d_t is the count of term *t* in the whole document set *D*. There are many modifications for the calculation of IDF.

TF-IDF is a good metrics to evaluate the significance of a term for a specified document in a whole document set. To fully understand TF-IDF, let's first look deeper at TF. Usually, if a term occurs in a document very frequently, the term frequency (TF) will be high. In such situation, we might say that the term is important to that document, or one of the topics of the document might be that term. In fact, this might not always be true. Think about those very popular, widely used term in life like "world", "eat", etc. It can occur in a document with a high frequency, yet the topic of the document may be totally different thing. This tells us that term frequency is not an adequate measure for the its in a document set. The use of term frequency alone in calculating the term weights in a document set does not always provide adequate high quality of information retrieval. Some terms that are with high frequency in one document can also be a high frequency term in other documents in that set, which hints that if this term is used for information retrieval, the whole document set may be presented to uses, which probably leads to a poor performed information retrieval system.

If a term occurs in a document with a high frequency yet with low frequencies in other documents of the document set, the weight of that term should be high and a good fit for that document. Inversed document frequency (IDF) was imported to address this issue (SPARCK JONES 1972). Based on its definition above, IDF indirectly evaluates collection frequency of a term in the whole

document set. Combined TF and IDF, the Weight of a term in a document is more reasonably reflected. TF evaluates the weight of a term in one specified document, while IDF plays roles in a document set. If a term is common in one document and very rare in the whole document set, both TF and IDF of the term will be heavy, but the TF of that term in other documents will be very small.

TF-IDF works well in many circumstances, yet it still has disadvantages especially when documents in a document set have large differences of document length. Longer document may have a high term frequency for a term because the term has higher opportunity to be repeated times in the document. Longer document may also have many other terms that may affect the retrieval process. Research shows that longer documents have higher probabilities to match users' multiple queries (Salton and Buckley 1988). To reduce the affection of document length, a normalization factor could be used to make the weight more effective.

Advanced vector model

Vector model (Castillo 2005) is another document or word representation techniques that becomes more and more popular as Deep learning algorithms develop. Features in a Vector model could be automatically obtained by a language model using (deep) neural network algorithms. Although the values of features representing a document may not be easily interpreted (Bordes et al. 2014), they are still very meaningful and interesting in indexing. Combining the techniques of cosine similarity, vector model is currently the best approach to building a computational model that successfully deals with the different aspects of word meaning including senses, antonym, synonym, similarity, etc. (Castillo 2005). The disadvantage of using vector model is it usually takes a lot time to train a neural network before a good vector model can be applied and the effectiveness of a trained vector model depends on the training data, both of which limit it from being widely used in general internet searching.

2.1.3 Creating ranked results

When someone performs a search, search engines scour their index for highly relevant content and then orders that content before presenting them to searchers. The process of ordering search

result is known as ranking. In general, the higher a website is ranked, the more relevant the search engine believes that site is to a query.

The PageRank (Brin and Page 1998) algorithm, named after Lawrence Page, who is one of the founders of Google, is a way of measuring the importance of website pages. It produces a probability distribution representing the likelihood that a person's random clicking on links arrives at any page. The algorithm, by analyzing the incoming and outgoing links of a web page and assigning a numerical weight t each of the pages, measures a page's relative importance in the page set; World wide web can be considered as such a page set. The numerical weight assigned to a given Element E is called as the PageRank of E and denoted by PR(E).

The PageRank algorithm uses a graph, where each node represents each Web page and edges represent the hyperlinks between pages, to describe the connections between Web pages. The rank value of a page represents the importance of that page, and the number of the hyperlinks pointing to a page can be considered as a vote of support to that page. The rank value of a page is calculated recursively and depends on the number and PageRank metric of all pages that link to it. A page that is pointing to or hyperlinked by many pages with high PageRank value also has a high rank value itself.

In the original paper (Brin and Page 1998) that introduced the PageRank, the rank value of a Page A ($PR(A)$) is defined as follows:

$$PR(A) = (1 - d) + d\left(\frac{PR(T1)}{C(T1)} + \cdots + \frac{PR(Tn)}{C(Tn)}\right)$$

Where T1, …, Tn are the pages which has hyperlinks pointing to page A. the parameter d is a damping factor with the range between 0 and 1, usually set to 0.85. *C(A)* is defined as the number of links going out of Page A. All web pages' PageRanks will sum up to one. $PR(A)$ can be calculated by implementing an iterative algorithm.

2.2 Wiki API for OpenSearch

OpenSearch (MediaWiki 2020) is a set of technologies by which search results in suitable format can be published for syndication and aggregation. Wikipedia opens its API for OpenSearch, which enables public to retrieve interested contents from Wikipedia by search keywords.

The main endpoint for Wiki search is: https://en.wikipedia.org/w/api.php. The API also provides some parameters that can be set by users to personalize Wiki search. The most frequently used parameters and their descriptions are listed in Table1.1.

Table 1.1 Main Parameters of Wiki API for OpenSearch

Parameter	**Default value**	**Description**
Action	“opensearch”	Opensearch protocol.
Namespace	“0”	Namespace to search.
Search		search string, this parameter is required.
Limit	10	Maximum number of results to return, must between 1-500.
Format	json	The format of the output.

For example, if we want to search 5 most related Wiki pages by keywords “cataract”, we can input the following address to a browser:

https://en.wikipedia.org/w/api.php?search=%22cataract%22&action=opensearch&namespace=0&limit=5&format=json

Wikipedia will return data like the following:


```
[
        "\"cataract\"",
        ["Cataract","Cataract surgery","Cataracts of the Nile","Cataract Dam","Cataractonium"],
        ["","","","","",""],
        [
                "https://en.wikipedia.org/wiki/Cataract",
                "https://en.wikipedia.org/wiki/Cataract_surgery",
                "https://en.wikipedia.org/wiki/Cataracts_of_the_Nile",
                "https://en.wikipedia.org/wiki/Cataract_Dam",
                "https://en.wikipedia.org/wiki/Cataractonium"
```

```
        ]
]
```

The first element of the returned data is the search keywords; the second element is a list that indicates the titles of 5 most related Wiki pages; the third element usually consists of a list of the pages description; the fourth element is the list of the address corresponding to each page title.

Executing the following codes written by Python come to the same result:

```
#!/usr/bin/python3

"""
    opensearch.py
    MediaWiki API Demos
    Demo of `Opensearch` module: Search the wiki and obtain
   results in an OpenSearch (http://www.opensearch.org) format
    MIT License
"""
import requests
S = requests.Session()
URL = "https://en.wikipedia.org/w/api.php"
PARAMS = {
    "action": "opensearch",
    "namespace": "0",
    "search": "Cataract",
    "limit": "5",
    "format": "json"
}
R = S.get(url=URL, params=PARAMS)
DATA = R.json()
print(DATA)
```

The API also provides alternative parameters which enables to modify the format of searching string and some other setting.

2.3 Internet searching to real-time help system in Hypocrates+

The objective of the Real-time help system is to retrieve useful short plain text that can be presented to Hypocrates+'s users. Given a keywords list, it's easy to get the address of each related Wiki page and then the HTML contents. Wiki API can help to complete most of the crawling work that a search engine will do. But the real-time help system will do more than just crawling. It needs to analysis each HTML page, extract useful short plain texts and provide helpful

one to Hypocrates users. Analyzing HTML page and Providing helpful text can be regarded as the indexing and ranking work for an internet search engine. The principles and techniques introduced in this chapter will play important roles in the whole research.

Chapitre 3 – Hypocrates+

Hypocrates+, compared with its previous version: Hypocrates, provides extra real-time help service to help a medical student expand their medical knowledge and correct mistakes during the interaction with the platform. In chapter1, we briefly introduced the architecture of Hypocrates. In this chapter, we will detail the novelties of Hypocrates+ as well as other components originally from Hypocrates.

3.1 Architecture of Hypocrates+

Hypocrates+ utilises most of the components and resources of its previous version: Hypocrates, including clinical case database, case manager, report and log system, virtual environment, and an improved intelligent agent. Real-time help system, instead, is a new component which will provide a real-time help service to the improved intelligent agent, which finally represents the help information to the platform user through the interaction with the virtual environment. EEG signals will also be recorded in real-time during the whole interaction. Figure 3.1 shows the architecture of Hypocrates+.

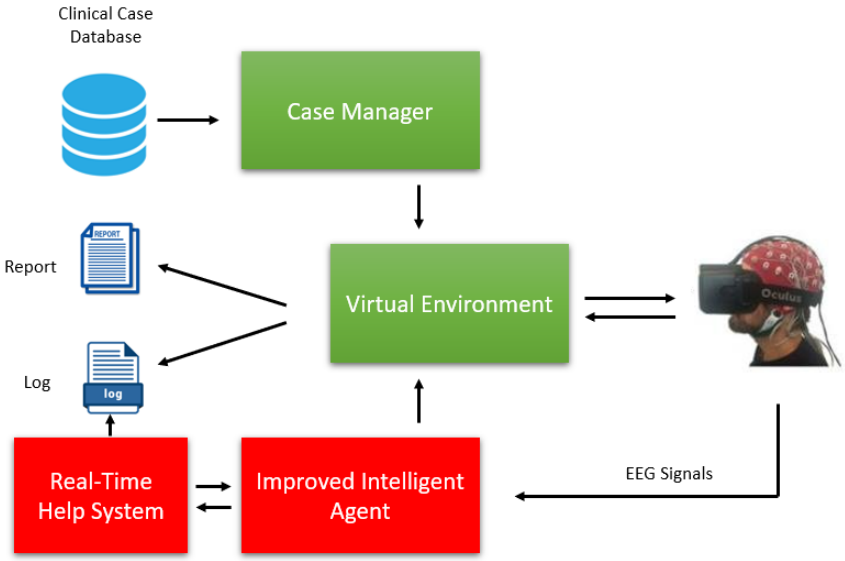

Figure 3.1 Architecture of Hypocrates+

Compared to the Architecture of Hypocrates, Hypocrates+ has an extra component called real-time help system and an improved intelligent agent, both of which are marked with red boxes. In addition to evaluate the emotional status of the platform user, the improved intelligent agent also decides whether and when it will make a help request to the real-time help system.

3.1.1 Clinical Case Database Component

Clinical cases are stored in Clinical case database in XML format. Each case, discussing a patient diagnosed by a specified disease, is represented by the following sections: general description, symptoms and signs, analysis, diagnosis, and treatment. General description section stores the general information of a patient, including the age, gender, the diagnosed disease name, etc. Symptoms section lists the symptoms provided by the patient. The analysis section lists possible clinical or laboratory examinations for the patient, some of which are necessary for diagnosis while some others are not. The diagnosis section lists one correct diagnosis and several false diagnoses. The treatment section, which is similar to diagnosis section, records one or several correct treatment plans and several false ones. Annex1 shows how an example of a complete clinical case was organized in XML format.

3.1.2 Case Manager Component

Case manager component, originally existed in Hypocrates, builds a list of several clinical cases from the case database for an interaction between a user and the platform, manages the data during such interaction including the details of the user's response, and provides the final report to the user after the interaction.

3.1.3 Report and Log Component

Report and log component is responsible for recording and saving all the information during the interaction, including a list of clinical cases that was provided by case manager component and used for the interaction with a medical student, EEG signal, all communication details between the improved intelligent agent component and the help system, etc. All this information will be saved in local files for query, analysis and research in future. This component was improved in Hypocrates+ since Hypocrates+ records more information than Hypocrates.

3.1.4 Virtual Environment Component

The virtual environment component is able to present various medical cases provided by case manager in real-time and produce realistic 3D objects and sound in order to make the interaction immersive. The virtual environment is composed of three main scenes: introductory scene to explain how the consequent interaction will go through and the possible actions a user can perform, case scene to present different kinds of virtual clinical environment according to the case content, and summary scene to briefly show the final report to user when the interaction is completed. From the final report, the user will see how many questions was correctly answered and how many helps was provided. Based on some features of clinical practices, two main scenarios (Figure 3.2) for case scene were developed: ICU scenario and Doctor's office scenario. If a virtual case is discussing an emergency, the user will find himself/herself in a virtual emergency room; otherwise, the platform will provide a virtual doctor's office to the user. In each scenario, the case information will be presented in several big panels. The user reads the information from the panels and make clinical reasoning by clicking buttons in the panels. A gauge displayed in one of the panels monitors the emotional status of the user during the interaction. The gauge value will change based on a specific algorithm when the user makes a correct or wrong choice.

(a) An emergency room scenario (b) A doctor's office scenario

Figure 3.2 Two typical case scenarios in virtual environment

In Hypocrates and Hypocrates+, clinical cases are displayed by texts in huge panels in virtual environment scenarios of either an emergency room or a doctor's office. With a VR glasses on, users will have an immersive experience of medical environment as if they are in a real medical environment.

3.1.5 Improved Intelligent Agent

Improved intelligent agent collects the EEG signals of a user during the interaction in real-time, evaluates the user's emotional status by analyzing the EEG signals, and intelligently decides whether a message of requesting a help is needed be sent to the real-time help System or not. In Hypocrates, the agent already exists but it only evaluates the emotional status by analyzing EEG signal. As Hypocrates+ is both a medical education and brain function research platform which is to be equipped with a real-time help system, we need a procedure to judge whether and when a help is needed for a user. In general, different uses of Hypocrates+ require different algorithms for the judgement procedure, and the algorithm may even change due to different scientific research purpose. Based on our design, the intelligent agent component took charge of the judgement procedure and evolved to "improved" intelligent agent.

3.1.6 Real-time Help System

The real-time help system is responsible for providing medical information to the improved intelligent agent. When the intelligent agent makes a judgement that an interacting user is needing a help, it will send a message to the real-time help system requesting for a help information. Once the help system receives this message, it will parse the message and search highly related information from the internet, currently more precisely, from Wiki pages. Once the help system finds some useful information that matches with the request, it will send the information back to the intelligent agent, and the information will finally be presented to the platform user: a medical student by the agent. Real-time help system in a completely new component in Hypocrates+, and design and implementing this system is the topic of this thesis.

3.2 Artificial Intelligence in Hypocrates+

There are two main components where artificial intelligence technology is involved: improved intelligent agent component and real-time help system component. In improved intelligent agent component, EEG signals are collected in real-time then converted to several values that represent the emotional status. This component also monitors the change of the emotional status and uses different algorithms to make decisions on whether some help is needed or not for the user based on different type of usage of Hypocrates+ platform. How the algorithms were implemented and

applied in this component goes beyond of this thesis. We will focus on how the real-time help system component is designed and implemented so as it can provide fast and useful real-time help service to the intelligent agent. All these will be fully explained in next chapter.

Chapitre 4 – Real-time Help System for Hypocrates+

Real-time help system (RTHS) is the most important component that differs Hypocrates+ from its previous version: Hypocrates. Provided the service by real-time help system, Hypocrates+ turns to a medical educational platform where medical students can learn and expand their medical knowledge and get help when they make mistakes during clinical reasoning through this platform. It also provides a platform where researchers can study the impacts of different strategies in providing real-time help information on human learning process. Intelligence and real-time service are the two main characteristics of the system, which will be fully discussed in this chapter.

4.1 Architecture of RTHS

RTHS functions as a server, providing services to the intelligent agent component of the Hypocrates+ platform; the latter can be considered a client of RTHS. The two components communicate with each other via user datagram protocol (UDP). RTHS itself has two main components: message queue component and intelligent information retrieval component, and an auxiliary log component which is a part of the report and log component of Hypocrates+. Figure 4.1 shows the general architecture of RTHS and the communication with its clients. Message queue component receives request message from clients and maintains a message queue; intelligent information retrieval component dequeues message from the message queue when the queue is not empty, parses the message, intelligently searches information from Wiki pages, and sends the search result back to the client who requested it; the log component records all the communication details and each search result. Due to the design of message queue, RTHS is capable to provide the service stably and robustly to multiple clients.

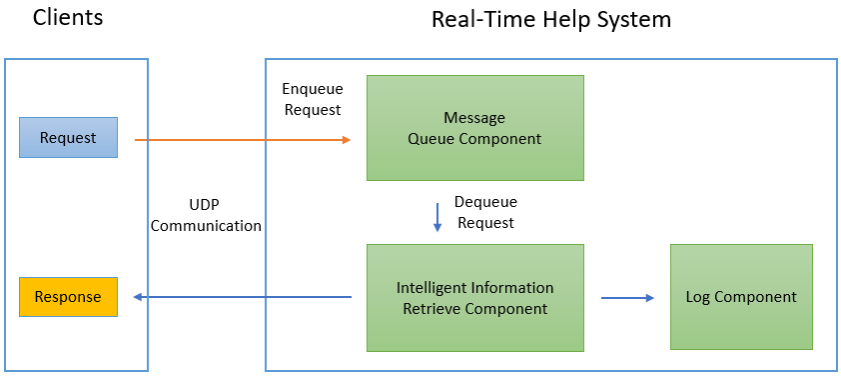

Figure 4.1 General architecture of RTHS and the communication with its clients

Real-time help system has two main components: message queue component and intelligent information retrieve component, and an auxiliary log component. Message queue component maintains a message queue, enqueues new request message and dequeues unsolved message to intelligent information retrieve component for information retrieval. The log component will save log all the communication data. Real-time help system communicates with its clients via UDP.

4.1.1 Message Mechanism Between RTHS and its Clients

It is through messages that RTHS communicates with its clients. Each time when a client believes a help is required for the platform user, it will send a request message to RTHS via UDP; when a help content is produced, RTHS will send a response message back to the endpoint who requested it. Both the request and response messages are predefined and composed of different fields. Figure 4.2 shows the complete fields of the two kind of messages.

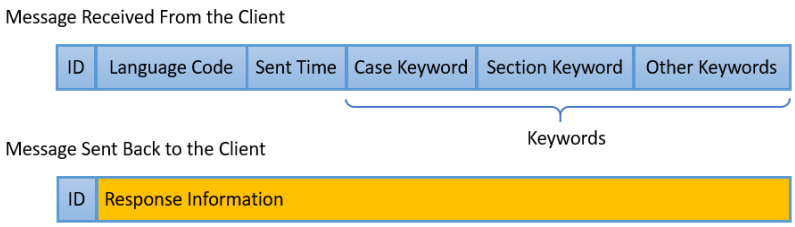

Figure 4.2 Definition and fields of the request and response messages

A request or a response message is a string that contains several above fields values separated by a “#” character.

An example of a request and response message can be the following:

Request: *"0012#fr#12:03#hypertension#analysis#Bandelette urinaire"*

Response: *"0012#Une bandelette urinaire ou tigette urinaire sert à réaliser des analyses médicales rapides permettant le dépistage de certains problèmes de santé, dont les infections des voies urinaires, la jaunisse, ou certains problèmes rénaux. La bandelette urinaire réactive immergée brièvement dans l'urine est lue par le praticien ou le particulier en la comparant à une échelle colorimétrique. Selon le type de réactifs, elle permet de déterminer le pH et de rechercher la présence dans les urines de glucose, de corps cétoniques, de leucocytes, de nitrites, de protéines, de sang, d'urobilinogène et de bilirubine."*

The request message tells that the intelligent agent needs some help information about the "b*andelette urinaire"* used in the *analysis* of the disease *hypertension*. The request is provided in French indicated by value of the language code: "*fr"*; the help information should also be in French. The intelligent agent marked that message was sent at *12:03* with a unified identity code *0012.*

The response message is very simple; it only included the help information and a repeated identity code *0012* so that the intelligent agent could recognized which request the help information is for in case that the agent send many requests in a very short time.

The following is the Definition and description of the fields:

- **ID**. the value of ID, determined by a client, is the identity of each message, helping the RTHS and the client to identify and manage each message. It is a unique character type.
- **Language Code**. This is a character type field telling RTHS in which language the help content will be provided so that RTHS supports multiple languages. It is a character type.
- **Sent Time**. This Date Time type field, determined by a client, records the date and time when a client sends a help request.
- **Keywords**. This character field records the value of three sub fields: case keyword, section keyword and other keywords. Case keyword tells RTHS the topic of the case that a medical student is interacting with the platform. Usually it can be a disease name in a specified language. Section keyword tells RTHS the topic of the question that a medical student is interacting with the platform. For example, a case keyword can be "hypertension", "cataract", or other disease name; a section keyword can be "Symptoms", "Causes",

"Diagnosis", "Analysis", "Treatment", "Prevention", etc. Other keywords sub field tells RTHS extra information about a detail point where a medical student gets stuck. It can be a formatted phrases or natural language content, such as "surgical approach in cataract treatment".

- **Response Information.** This character type records the information that RTHS generates and will send back to the client who requests it.

RTHS also defined some other fields to record the address and time information during the communication. These are:

- **Request Received Time.** This Date Time type field, determined by RTHS, records the date and time when RTHS receives a help request from one of its clients.
- **Response Sent Time.** This Date Time type field, determined by RTHS, records the date and time when RTHS sends back a response message to the client who requests it.
- **Endpoint (IP address and port).** This field records the IP address and corresponding port of a client, so that RTHS knows where it will send the response message.

RTHS uses message, an instance of the following data structure, to store the information concerning all the above fields in one request and response round (Figure 4.3). A message queue manages the enqueue and dequeue operations and fill the field values in proper time. Such design of message queue enables RTHS to handle multiple clients requests orderly and to log the interaction history easily.

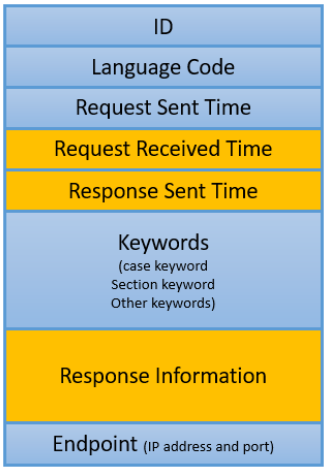

Figure 4.3 Data structure of message in message queue

Figure 4.4 shows the message flow of Hypocrates+ from which we can see when the value of a message field will be generated. When a client starts a help request, it will provide the information of request ID, language code, keywords (the combination of case keyword, section keyword, and other keywords), request sent time, and endpoint. When RTHS receives a help request, it first generates the information of request receive time, then enqueue the message to a message queue. The intelligent information retrieval component will dequeue a message from the queue, then generate the value of the response information field. When RTHS sends the response information back to the client specified by the endpoint information field, it will record this time to the field of response sent time. Thus, each field of a message is complete, and the log component will record the complete information for future query or analysis.

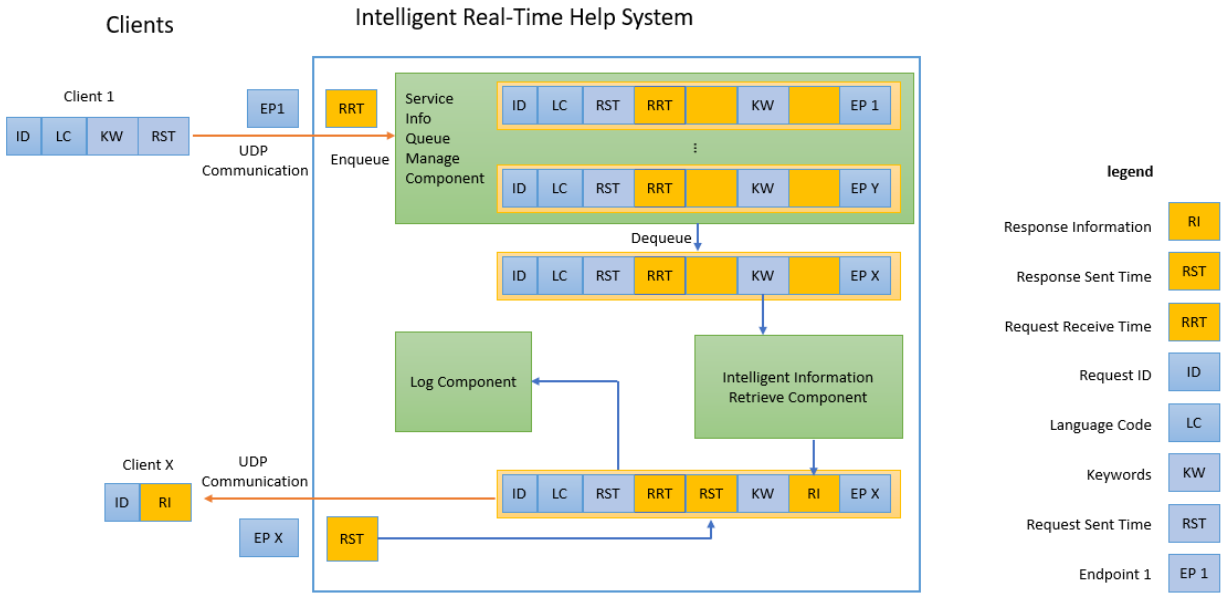

Figure 4.4 Message flow In Hypocrates+

This figure shows the workflow of a message in Hypocrates+ and its clients. Each small or yellow box represents a field of a message. blue field means the value is determined by a client; yellow box means the value is determined by real-time help system (RTHS). The complete name of each field is explained in the legend in the right.

4.1.2 Intelligent Information Retrieval Component

Intelligent information retrieval component (IIRC) is the core component that generates intelligent response information to a client; The only thing that IIRC needs and uses to produce response information is the keywords value from a message. Figure 4.5 shows the IIRC mechanism.

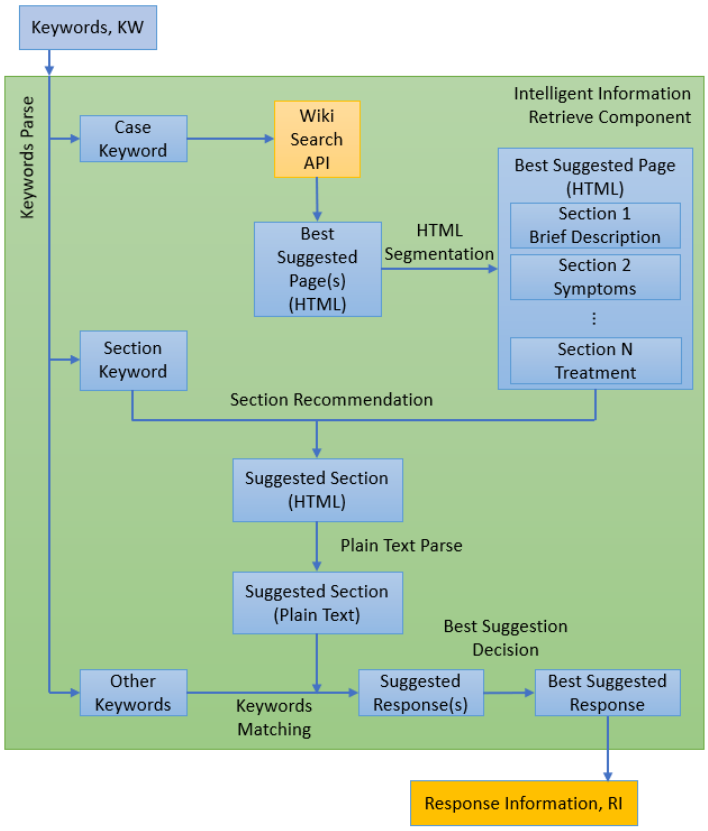

Figure 4.5 Mechanism of intelligent information retrieval component

This figure briefly describes the workflow of the intelligent information retrieval component (IIRC). Starting from parsing the keywords from a request message, IIRC first produces several suggested pages by wiki open search API, then for each suggested page, it divides the page into several sections, compares the section name with the section keyword to do section recommendation. In a suggested section, IIRC convert the html text to plain text and find short and useful plain text by keywords matching. Finally, IIRC will select the best short plain text from several suggested responses from different pages and output the best information. The whole process is very similar to how a web search engine works.

A complete workflow of IIRC includes the following main steps:

- Wiki Search API

When keywords information is dequeued from a message by message queue component and sent to IIRC, it will be parsed to three parts: case keyword, section keyword, and other keywords respectively. IIRC first uses open search API provided by Wikipedia to obtain several best suggested html formatted Wiki Pages based on the case keyword value.

- Levenshtein Distance for Section Recommendation

To speed up the search process, considering the general structure of medicine related Wiki pages, IIRC then analyzes each suggested html Wiki page and segment it to different sections. The number and the names of sections in each suggested Wiki page may vary a lot, which means not all Wiki pages have same sections like “symptoms”, “diagnosis”, “treatment”; some may or may not have one of the sections like “prevention”, “scientific research”, “prognosis” etc. Besides, as Wiki pages are contributed by publics, a section that focuses on same section topic may have different names. For example, the section “symptoms” may occur as “symptoms and signs” in some Wiki pages. The names are different, but both sections provide similar aspect of a disease. To address this problem, IIRC uses section recommendation mechanism to select the most related section for further analysis. One algorithm that can be used to implement this mechanism is comparing the Levenshtein Distances between the section keyword and the section names in a suggested Wiki page and selecting the section that has lowest Levenshtein Distance with the section keyword.

Levenshtein distance is a string metric for measuring difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. It is named after Vladimir Levenshtein, who considered this distance in 1965.

Mathematically, the Levenshtein distance between two strings a, b (of length $|a|$ and $|b|$ respectively) is given by $\mathrm{Lev}_{a,b}(|a|, |b|)$ where:

$$Lev_{a,b}(i,j) = \begin{cases} max(i,j) & if\ min(i,j) = 0, \\ min\begin{cases} Lev_{a,b}(i-1,j) + 1 \\ Lev_{a,b}(i,j-1) + 1 \\ Lev_{a,b}(i-1,j-1) + \mathbf{1}_{(a_i \neq b_j)} \end{cases} & otherwise. \end{cases}$$

Where $\mathbf{1}_{(a_i \neq b_j)}$ is the indicator function equal to 0 when $a_i \neq b_j$ and equal to 1 otherwise, and $Lev_{a,b}(i,j)$ is the distance between the first i characters of a and the first j characters of b, where i and j are one-based indexes.

Wagner-Fischer algorithm, a dynamic programming algorithm developed in 1974, can be used to calculate the Levenshtein distance between two strings. Given two strings: a of length m, and b of length n, the following pseudocode provides an implementation of Wagner-Fischer algorithm.

```
function WagnerFischerLevenshteinDistance(string a, string b):
    // the lengths of a and b are m and n respectively
    // all indexes start from 0
    declare d[m+1, n+1] and initialize all its elements to 0
    for i from 0 to m:        // m included
        d[i, 0] = i
    for j from 0 to n:        // n included
        d[0, j] = j

    for j from 1 to n:        // n included
        for i from 1 to m:    // m included
            if a[i] == b[j]:
                substitution_cost = 0
            else:
                substitution_cost = 1

            d[i, j] = min(d[i-1, j]+1, d[i, j+1]+1, d[i-1, j-1] + substitution_cost))

    return d[m+1, n+1]
```

The Wagner-Fischer algorithm has both the time and space complexity of $O(mn)$.

Levenshtein distance is just one measure to calculate the distance of two strings. In RTHS, the Levenshtein distance may not always accurately tells the difference of the meaning of two words to find the best match section in a Wiki page. We have solutions, which will be explained later, to solve this issue.

- HTML text to Plain Text

For each suggested Wiki page, a most related section will be recommended based on the above section recommendation mechanism. However, the recommended section is too long, and it is a

HTML formatted string which needs to be converted to plain text. Besides, some tiny tags such as reference tag like “[1]” and edit tag like “[Edit]” also need to be removed. All these works can be done by some string operation using regular expression techniques.

- Shortening the plain text

The purpose of this step is to provide a shorter but precise text from suggested section of plain text. There are several techniques can be applied for this purpose, from the simplest keywords matching to the most advanced natural language understanding (NLU). Considering NLU needs very large amount of trainable data which currently we don’t have, Keywords matching technique is applied at this moment, but an interface for more advanced techniques are well predefined during implementation, which will be explained later.

- Final selection

For each suggested Wiki page, step 2 to step 4 will be performed; therefore, if more than one Wiki page are provided by the search API, we will finally come to a list of suggested short plain text. As the pages provided by Wiki search API are ranked by the relativity, usually the first element in the list is preferred as the final selection to the client.

There may also occur a situation where there is no suggested short plain text after all the suggested Wiki pages are analyzed. It is rare, but it does occur. To address this situation, RTHS will send the “other keywords” as a case keyword to Wiki search API for suggested Wiki pages. In such circumstances, section keyword will not be used; instead, RTHS will only extract the “Brief Description” section of suggested pages and perform the remaining steps as usual.

Even though, RTHS may also come to a result of no proper response. Sometimes, there might be a failure to access the internet, or too long time is spent before a response is return. In all these cases, RTHS will send corresponding messages telling the client special response code, and the client decides what information will be displayed to the platform user based on response code. The most common codes are: NM, TO, and WF, meaning “no good matched response”, “time out”, or “wiki access failure” respectively.

4.2 Key Implementation of RTHS

The programming language for developing RTHS is C# with Visual Studio 2019 as the integrated development environment. In the whole solution, four projects were created: WikiSearchServer, ClientApp, WikiAPI, and HtmlAgilityPack. In Figure 4.6, the right panel shows the structure of the four projects in the solution.

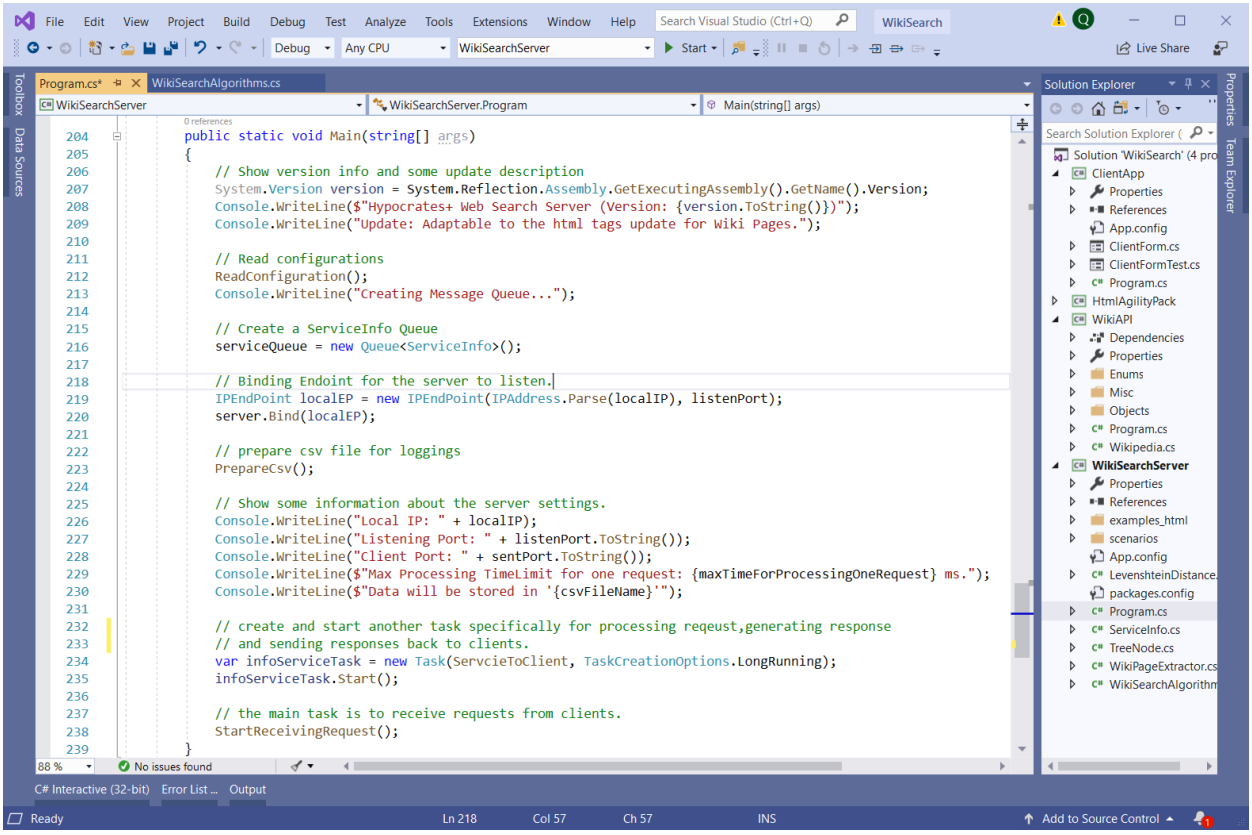

Figure 4.6 Structure of the solution and code snapshot in Main function

The right part of this figure shows the solution has four projects; the left part shows the complete codes in Main function of the project WikiSearchServer. In the Main function, after displaying the version information, the application read configuration, created a queue, started listening local endpoint, prepared log files, then created and started task “infoServiceTask” for information retrieval process, and finally started the message queue management loop in the main thread.

WikiSearchServer project, as the main and start-up project, implements the main branch of the functions required for RTHS and will be compiled to “.exe” file for running. This project is designed as a console service application as there is no need for UI access. When the application is running, it maintains two threads: one for message queue management and the other for processing

search from internet and sending back the information to client. In Figure 4.6, the left panel shows the complete main codes of this project. From the codes, we can easily see how the application starts the service by creating and running the two threads.

ClientApp project, a windows Form client application, is built to simulate the real client. Since running the real 3D unity client application spends much time and memory during development. This “fake” client will follow the real communication protocol so as to facilitate the debugging and test of RTHS.

WikiAPI project provides access to searching from Wiki for most related Wiki pages based on predefined parameters such as search keywords, language, etc. Functions implemented in this project will be used in the main Project for RTHS to search from Wiki pages.

HtmlAgilityPack (Html Agility pack 2020) is an open source library which provides tools that can be used for HTML parsing, Wiki page segmentation, HTML to plain text, etc. These functionalities are required by the IIRC component in the main project.

WikiSearchServer is set to the start-up project. WikiAPI and HtmlAgilityPack will be compiled to DLLs to support the start-up project. ClientApp is a independent project for simulating clients of RTHS.

4.2.1 WikiSearchServer

Many core tasks will be solved in this project such as the message management, calculation of Levenshtein distance, segmentation and parsing of a HTML page, and generating the final plain text, etc.

A ServiceInfo struct was implemented with which a complete (request and response) communication data between RTHS and its client can be temporarily stored in an instance of the struct in memory. According to the communication protocol, a request or a response message is a string object that contains several struct fields separated by a “#” character. ServiceInfo struct provides a function to parse each field values from a request message string, a function to generate a response message string, and a function to write or append a complete communication data to a local “.csv” file as a part of the log component of Hypocrates+.

A *TreeNode* generic class was implemented to manage the multi-level sections of a Html page. By managing an instance of such tree node, a Wiki page can be treated as different sections in hierarchy, which is useful for access of section names, calculating the Levenshtein distances in section recommendation.

A *LevenshteinDistance* class was implemented to calculate the Levenshtein distance between two strings, one is section keyword from request message and the other is a name of a Wiki page section from an instance of tree node explained above.

A *WikiPageExtractor* class was implemented to provide most functions for Html page analysis with the help of an HtmlAgilityPack instance. These functions include extracting a brief description of a Wiki page, a html string of a specified section, turning html string to a plain text string, removing additional reference and edit tags which occur very frequently in Wiki pages, and Ignoring some useless sections like “See also”, “External Links”, etc.

A *WikiSearchAlgorithms* class was implemented to perform intelligent information retrieval function. In this class, techniques like Task, Func, and Action in C# are widely used to provide a flexible and easy access to different search algorithms. A static method called “Search” is declared as follows:

```
public static class WikiSearchAlgorithms
{
	private static string Search(Keywords keywords, Language lang,
		Action<Keywords, string, List<string>> OnePageSearchAlgorithm,
		Func<List<string>, string> SelectBestAlgorithm,
		CancellationTokenSource cts){}
}
```

The “Search” method receives five parameters: keywords, language code, search algorithms on one Wiki page, algorithm for final best selection, and a cancellation token source. Among the five parameters, **OnePageSearchAlgorithm** and **SelectBestAlgorithm** are two delegates which need to be implemented elsewhere. The whole method will return a string as the final help information. Currently, the implementation of “Search” method and its two delegates followed the mechanism that is previously explained by Figure 4.5.

In this project, some settings are also provided to allow user to dynamically modify some values of the settings. These settings defined the IP and listening port of the RTHS server, port of a client, maximum waiting time for a help information, minimum and maximum length of a help information, and the file path where all communication data should be saved. These settings are stored in a Xml file with default values as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2"/>
  </startup>
  <appSettings>
    <add key="LocalIP" value="127.0.0.1"/>
    <add key="ListeningPort" value="8052"/>
    <add key="ClientPort" value ="8053"/>
    <add key="MaxTimeOut" value ="10000"/>
    <add key="MaxMsgLength" value="600"/>
    <add key="MinMsgLength" value="80"/>
    <add key="CsvFileName" value ="abc\data.csv"/>
  </appSettings>
</configuration>
```

4.2.2 ClientApp

The purpose of building a ClientApp project is to provide a client that simulates the real client which was developing by another research member in the group. It will be compiled to a Windows Form application with a main “ClientForm” form displayed in left part of Figure 4.7.

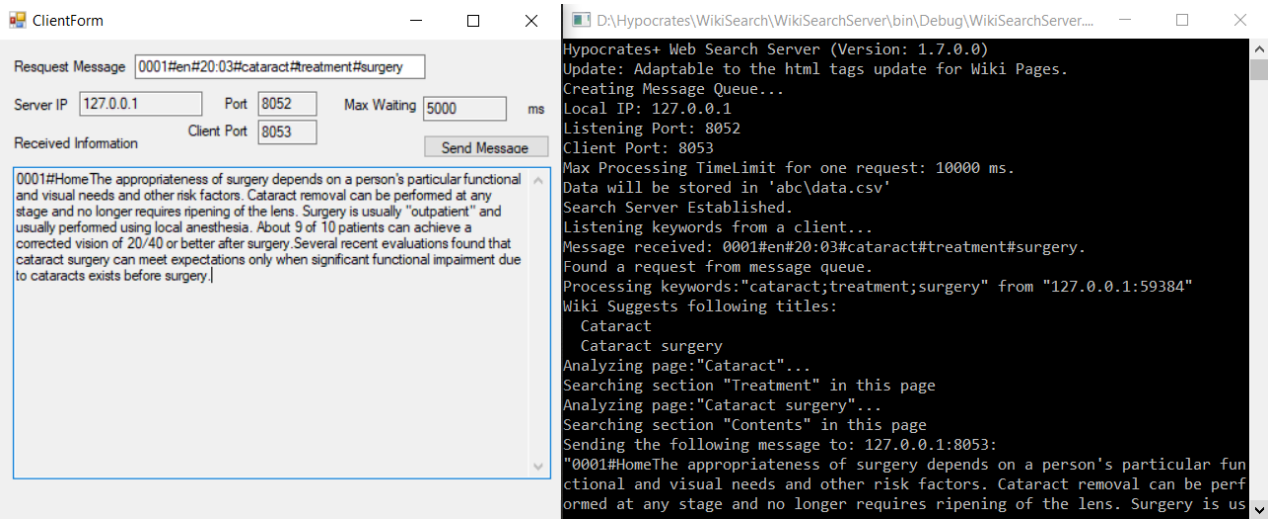

Figure 4.7 A Windows Form UI client and RTHS console server.

The left part of this figure shows the UI of the "fake" client application. Developers and testers can use this client application to test and debug the RTHS, which is a console application displayed in right part. The figure shows that a request message("0001#en#20:03#cataract#treatment#surgery") was typed in the textbox at the top of left panel. When the "Send Message" button was pressed, the right panel received the message, parsed the keywords, got 2 Wiki suggested pages: "Cataract" and "Cataract surgery". The server first analyzed the page "Cataract" in its "Treatment" section, then continued to analyze the page "Cataract surgery" in the section "Contents". Finally, the server generated a response message and sent it back to the client. The client will show the message in its "Received Information" textbox after successfully received it.

The "ClientForm" form allows the developer or a test user to input the request message and send the message to the server. When the server sends back a response message to the client, it will display the received information in the "Received Information" textbox. The right part of Figure4.7 shows the console output when it is activated and processing the request from the client. In "ClientForm" class, an instance of "UdpClient" class was implemented to perform the message sending and receiving via UDP.

4.2.3 WikiAPI

WikiAPI provides a DLL tool to search from Wiki pages by keywords and to obtain the search result. It also provides an access to modify the search settings, including the search language, maximum pages returned etc. The development of the DLL is based on the interface provided by Wikipedia (MediaWiki 2020). Two most important functions provided by this project are: 1) obtaining some suggested Wiki pages according to the keywords provided and 2) obtaining the whole Html content of a complete Wiki page for analysis. These two functions are provided by method "Search", and "GetWebData" in "Wikipedia" class, respectively.

```
public class Wikipedia
{
    // some other codes
    public string GetWebData(string title) {    }
    public QueryResult Search(string query){    }
}
```

`QueryResult`, as the return type of the method "Search", has a `List<Search>` property where a list of search result is stored:

```
public class QueryResult
{
    public SearchInfo SearchInfo {get; set;}
    public List<Search> Search {get; set;}
    public string ServedBy {get; set;}
```

```
        public Error Error {get; set;}
    }
```

Search, as the type of the element in the list returned by the method “Search”, has an important property called “title”, whose value can be used as the value of the parameter of the method “GetWebData” to get the full html content corresponding to the title.

4.2.4 HtmlAgilityPack

HtmlAgilityPack (Html Agility pack 2020), an open source project, is an Html parser written in C# to read and write DOM and supports plain XPATH or XSLT. It provides a `HtmlDocument()` class, from which most of the functions like Html parsing, section segmentation, and plain text transformation for RTHS could be obtained easily.

In general, several different algorithms were used to implement different components of the real-time help system. First, a message queue was used to handle requests from clients; then Wiki search APIs were applied to retrieve suggested Wiki Pages based on keywords extracted from a request; for each Wiki page, section recommendation was applied to provide suggested section so that further search is restricted within a high-related section of a Wiki page. Finally, techniques such as key words matching and regular expression, were used to produce a short message that can be sent back to the client. The following accuracy test and a response time test results proved the effectiveness of the algorithms we chose for the system.

4.3 Tests

Before RTHS was integrated to the Hypocrates+ platform for formal experiment, two tests were performed to check the accuracy of the response information and the response time of RTHS. 10 clinical cases in French language were selected, which bring us 387 help requests and corresponding responses.

4.3.1 Accuracy Test

RTHS uses message mechanism to communicate with its clients. Keywords, including the case keyword, section keyword, and other keywords, play crucial roles during the communication. A mechanism for auto keywords generation was proposed as follows:

1. Use the disease name of a topic scenario as the case keyword;
2. Use “symptoms”, “analysis”, “diagnosis”, “treatment” as the value for corresponding section keyword;
3. Use the name of a symptom, examination technique, disease, or a treatment method as the other keywords for a specific item.

For example, there is a case that discusses “hypertension” in the case database; in the “Analysis” section, a student will be asked whether a “ECG” exam is necessary. In this situation, the case keyword will be “hypertension”; the section keyword will be “Analysis”, and other keywords will be “ECG”.

To check the accuracy of the generated keywords based on the above mechanism, an experiment was designed and performed. 10 clinical cases in French language were selected from the case database; keywords were auto generated by the above mechanism, packaged into a message and sent to RTHS. RTHS finally produced a response information for each request. A medical professional was invited to evaluate the correlation between the keywords and the response information. Result shows that among the total 387 messages in 10 cases, response information for 373 messages are highly related and with high quality to the keywords; 12 are with lower or no relation to the keywords; 2 with a NM response code, meaning that there is no good matched response by IIRC for them. The total accuracy ratio is 96.4%, which is acceptable.

The common reason of those 12 lower quality response information was analyzed and shown in Table 4.1.

These lower quality responses suggest that, by using keywords as the main communication approach, it’s better to use complete, clear, and formal medical words or phrases to introduce

clinical cases; abbreviations, redundant expressions, and informal use of phrases should be avoided as much as possible.

Table 4.1 The common reason of lower quality response information

reason	# cases	example	
		keyword	suggested keyword
use of abbreviation	4	HTA essentielle	hypertension essentielle
unnecessary extra word	3	état de fatigue	fatigue
duplicated name in Wiki	3	repos	repos physique
Informal use	2	blouse blanche	Effet blouse blanche

4.3.2 Response Time Test

In this test, we evaluate how quickly our real-time help system responds to a request. we define that a response time starts from the time when IIRC dequeues a message from the message queue and ends with the time when the response generated by intelligent information retrieval component. The time when a message stays in the message queue or the time when the response message is being sent back to client will not be counted since these two intervals are affected by the load of requests and the network speed between the client and RTHS. Table 4.2 shows the basic statistics of all 385 response times.

Table 4.2 Basic Statistics of Response Time (s)

#	Min	Max	Mean	Variance
385	0.7712	2.2937	1.5120	0.0783

In general, the response time is quite short and acceptable for a real time help system.

The above tests are about the RTHS itself; how the RTHS will perform when integrated to the Hypocrates+ platform is the subject that will be discussed in the next chapter.

Chapitre 5 – Experimental Setup and Results

An experimental study was conducted by our research team to study the emotional change of medical students when interacting with Hypocrates+ and to further investigate the performance of RTHS as well. Certificate of ethical approval for the study was initially issued by the "Comité d'éthique de la recherche en sciences et en santé (CERSES)" on April 1, 2019 and renewed on February 19, 2020.

The subjects of the study will be the first or second year of medical students. This is because current target users of Hypocrates+ are also such population, who have some knowledge of medical theory and need to practice the knowledge by clinical cases. Users with too less medical knowledge will be unable to understand the cases provided by the platform, while advanced medical students or professionals may not be satisfied with the functionalities provided by the platform.

5.1 Objective of the experiment

The whole study is supposed to recruit 20 participants. The analysis will be focused on two main aspects: 1). whether the help information by RTHS is accurate and useful from the perspective of the platform users: medical students. 2). whether it is possible to help medical students reduce their negative emotions by Hypocrates+ with a real-time help system and an improved intelligent agent.

Two questionnaires before and after an experiment is used to help analyze the accuracy and usefulness of the help content that RTHS provides. During each experiment, the EEG signal is collected at the frequency of 128Hz; a software called Emotive SDK analyzes the EEG signals at the frequency of 36Hz and produces five values representing the status of five kind of emotions: mediation, frustration, engagement, excitement, and valence. Our previous studies show that frustration values can be used to describe the status of negative emotion, and reduced negative emotion can help produce a better performance (Ben Abdessalem et al. 2020), we will focus on analyzing the change of the frustration values before and after a help content is sent to the

participant, trying to find whether a real time help content helps to reduce negative emotion represented by frustration value.

5.2 Criteria for inclusion and exclusion

Inclusion Criteria

A participant(subject) will be eligible for inclusion in this study only if all the following criteria apply:

1. Male and female medical students with ≥ 18 years of age.
2. Currently in first or second study year.
3. The subject is able and willing to comply with the requirements of the study.
4. Voluntarily signed informed consent obtained before any study procedures are performed.

Exclusion Criteria

A subject who is not comfortable with experiencing 3D virtual reality environment will not be eligible for inclusion in this study.

The research group first published recruitment poster at the campus (see Annex2 Recruitment poster). Students who are qualified and interested in our research can make an appointment with us by telephone call or online registration. Our research member will verify if the participant meets the study conditions based on the inclusion and exclusion criteria listed above. Qualified participants will be invited to our experiment room for a 45 – 60 minutes experiment.

5.3 Scenario of the experiment

RTHS has the capability to provide service to more than one client at a time, restricted to the purpose of the experiment, only one client will be served by RTHS at a time, and both RTHS and the other part of Hypocrates+ will be running on a single workstation.

When a participant enters our experiment room, he/she needs to sign an informed consent form explaining the purpose and the procedure of the study, the risks and benefits, the right to refuse

or withdraw at any time, and other issues such as confidentiality, collecting the data, and sharing the results, etc.

Having signed the informed consent, the participant will be asked to fill some pre-session forms so that the researcher have access to know the participant's medical background and personal characteristics. Then, our research member will help the participant to put on an Emotiv EPOS headset to collect the EEG signals and a FOVE VR goggle to experience the immersive virtual medical scenarios. The connectivity of the Emotiv EPOS headset sensor will be checked and calibrated so that the quality of collected EEG signals is good enough for real-time emotional analysis. After the calibration, a research member launches all components of Hypocrates+: virtual environment, real-time help system service, and improved intelligent agent. The participant will be able to see the virtual environment through the VR goggle and interact with the environment with a wireless gamepad. EEG Data generated throughout the experiment will be recorded.

The participant is asked to solve 10 clinical cases selected from the clinical case database. For each case, general information (age and sex) and symptoms of a simulated patient will first be presented in a panel in the virtual environment. The participant reads the information and can move to the analysis and diagnosis panels. In these two panels, the participant is asked to answer some choice questions to complete the clinical reasoning of the case. Figure 1.2 in Chapter1 (Page28) shows the virtual scenario from the view of a participant. During the whole interaction, intelligent agent will analysis the EEG signals in real time and judge the emotional status of the participant. If the agent believes that the participant needs a help to maintain the peaceful and positive emotion, it will send a request message with keywords included to RTHS and get back the response message normally with help information included. Then it will present help information to the participant immediately. If RTHS's response doesn't include help information, which may due to failure of internet access or other problems, the agent will react to the participant as it didn't make any request.

When completing the interaction, the participant is asked to fill a post-session form, telling the researcher his/her general feeling about the experience, making his/her own judgement on how

accurate and useful the help information is, and some other questions about the Hypocrates+ platform but beyond the scope of RTHS. When the participant completes this form, the experiment is complete. The research group thanks the participant, and the participant leaves the experiment room.

5.4 Results and discussion

5.4.1 Participant profile

The first qualified participant was recruited on January 6, 2020. Due to the worldwide pandemic of the COVID-19, the study has to be paused. 5 experiments have been completed by March 13 when Québec government declared health emergency and published warning to cancel internal meetings.

All the 5 participants are students from faculty of medicine, University of Montreal. They are all in their 2nd academic year of the medical doctor (MD) program. According to the faculty's program structure, the MD program lasts four or five years, including an optional preparatory year, 2 pre-externship years, and 2 externship years. Medical students in 2nd academic year already have some knowledge of biological sciences, ethics, psychology, and sociology. Besides, they are in the second-half process of learning sixteen clinical courses in the form of problem-based learning, meaning they also have some clinical knowledge for clinical reasoning but don't have opportunities to practice their clinical reasoning skills.

All the 5 participants were able and willing to comply with the requirements of the study, signed informed consent, and didn't develop any uncomfortable experience during the experiment. Measured by the experiment's inclusion and exclusion criteria, the 5 participants were eligible. Table 5.1 shows the basic information of 5 participants.

Table 5.1 Basic information of 5 participants

Participant ID	Gender	Age	Study year
1	Female	28	2nd
2	Female	29	2nd
3	Male	25	2nd
4	Female	21	2nd
5	Female	22	2nd

5.4.2 Accuracy and usefulness of help information

Among the 5 experiments, 3 believe that the help information provided by RTHS is very accurate, 2 accurate; 2 believe the help information is useful, 2 neutral, and 1 less useful (see Table5.2).

The accuracy of the help information evaluated by the participants is consistent with the internal test result by a medical professional. As to the usefulness, however, 1 over 5 participants believe that the help information is less useful. Although more experiments need to be done before we can come to a more convincing result, for RTHS, it's always better to fully discuss the reasons that could cause the less accurate and less useful information by RTHS.

Table 5.2 The accuracy and usefulness of the help information

Category	not accurate at all	less accurate	neutral	accurate	very accurate
# case	0	0	0	2	3

Category	not useful at all	less useful	neutral	useful	very useful
# case	0	1	2	2	0

Generally, the less useful help information may occur when the information was not what participants want due to a specific information source like Wikipedia in this research. As described on its website (Wikipedia 2020), Wikipedia is a free online encyclopedia, created and edited by volunteers around the world and hosted by the Wikimedia Foundation, Wikipedia produces each web page to introduce a topic, which is useful for public to get a basic idea of that topic. Wikipedia

also has many pages focused on medicine, but for medical students, this information is deep enough for medical education. Better medical information sources for Hypocrates+ could be textbooks of different subjects, which are mentioned in first chapter, and diagnosis manuals officially issued by different sorts of medical association. These sources provided more accurate and deep medical information and knowledge very close to medical students. The difficulty of using these sources to Hypocrates+ is that many preliminary works, such as converting the contents of different topics into a unified structure, tagging sections, need to be complete before they can be quickly query.

One possible way to improve the accuracy and usefulness of help information is to study further on the real reason behind a medical student's stuck or confusion on a single question within a clinical case. In other words, when a student is confused by an option provided in a choice question, it doesn't always mean that the student doesn't understand what this option are saying about; sometimes, it is because of the unawareness of the significance of the information in the option to the question that a student gets confused. To refine and classify the real reasons may help the real-time help system to provide more accurate and useful help information. The challenge is that more medical knowledge and field experts are required for the research, which may be covered in the future work of Hypocrates+.

Another reason for the less accurate or useful help information could be the mechanism of the agent, which determines the time and frequency to request and present the help information in Hypocrates+.

5.4.3 Negative emotion analysis

Table 5.3 and Figure 5.1 show the average frustration values before and after help of 5 participants. 4 participants' average frustration values were reduced by help content.

Table 5.3 Average frustration level of the participants

Participant ID	Total average frustration value		Total times of help	Times of decreasing of frustration level
	Before help	After help		
1	0.7328	0.6972	12	9
2	0.4093	0.4212	10	4
3	0.3958	0.3681	7	3
4	0.6233	0.5605	17	14
5	0.4876	0.4574	10	6

Frustration value ranges from 0 to 1. Higher frustration value means a higher negative emotion. The value in the column "total times of help" stands for the times that the help is triggered during the whole experiment (10 clinical cases) for a participant. The value in the column "times of decreasing of frustration level" describes the times that negative emotion was reduced by a help. For example, during a whole experiment for participant 1, 12 helps were triggered, 9 reduced the negative emotion. the average frustration value before and after these helps are 0.7328 and 0.6972, respectively. Detailed data can be found in Annex4.

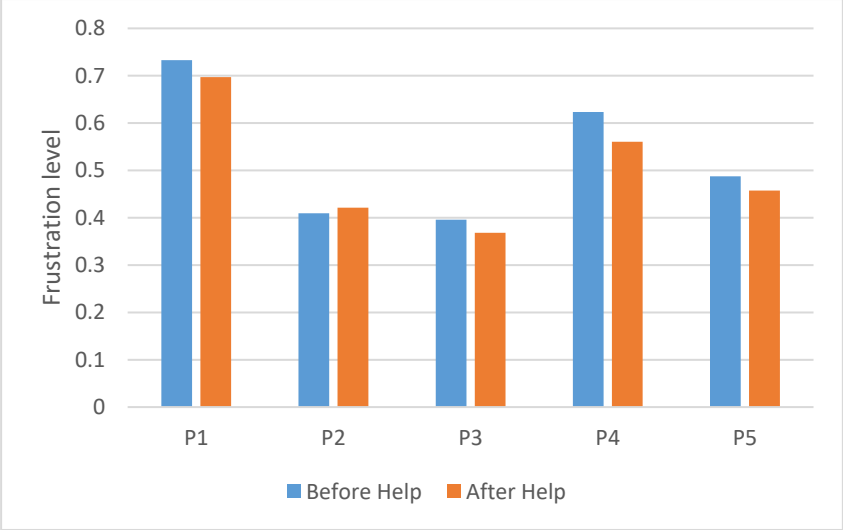

Figure 5.1 Histogram of mean frustration level before and after the help of 5 participants

Let F_b and F_a denote the frustration value before and after help respectively,

set the hypothesis as follows,

$$H_0: F_b = F_a \ , \ H_1: F_b \neq F_a$$

, and select $\alpha = 0.05$,

then, two-sided paired *t* test result (statistic value = 2.4143, *p* = 0.073) shows that we can not reject H_0, which means there is no significant statistical difference ($\alpha = 0.05$) that supports real-time help content can reduce negative emotion.

If we assume that help contents will never increase the frustration value, then we can use one-sided paired t test, set the hypothesis as follows,

$$H_0: F_b \leq F_a \ , \ H_1: \mathrm{F_b} > \mathrm{F_a}$$

then $p = 0.037 \leq \alpha = 0.05$. we reject H_0, which means there is a significant statistical difference ($\alpha = 0.05$) that supports real-time help contents can reduce negative emotion.

Since we currently have no solid scientific evidence which can prove a help content will never increase frustration value and we only have five participants, we can not conclude that the real-time help can reduce the negative emotion. More experiments are needed before we can make a more convincing conclusion.

Chapitre 6 – Conclusion

In this work, we designed and developed a real-time help system as a component of a virtual clinical reasoning platform for medical education: Hypocrates+, which can also be used as a tool for research on brain function assessment in learning. The real-time help system is dedicated to generating useful and effective short piece of medical knowledge as responses to requests from an improved intelligent agent which is another component of Hypocrates+. Based on this work, we submitted a paper (See Annex3) to 2nd international conference of Brain Function Assessment in Learning (BFAL), and the paper is accepted.

We started the work from reviewing the evolution of modern medical education, the most important skills in clinical practice: clinical reasoning, and the use of information techniques in helping medical students developing their clinical reasoning skills. In modern medical education, theoretical and practice are two main aspects. While the practice often takes places in clinics or hospitals, the theoretical aspect usually occurs in classrooms or laboratories. Because of the speciality of medical education, it is most from textbooks and lectures that junior medical students learn medical knowledge, which is not a very efficient way to develop medical students' clinical reasoning skills. Therefore, medical educators developed several new methods, such as problem-based learning (PBL) and case-based learning (CBL) to address this issue in theoretical aspect. In these methods, multi-media resources rather than just plain textbooks were widely used to let students have more image and video input and intuitive experience about a piece of medical knowledge. Recently, virtual reality (VR) technology has a tremendous development and is being quickly applied to many industrial fields, including medical education. Through VR technology, virtual yet very real-like clinical scenarios or cases can be placed in front of a medical student, which deeply impact the mode of theoretical medical education. The effectiveness of VR technology in medical education has been proved. But there are still some problems or questions to be solved or answered, such as how we can provide a real-time help during the virtual learning process, what the detailed mechanism is for the learning of students in such a virtual environment, and so on. By having designed and developed this real-time help system for

Hypocrates+, we provided a more powerful tool for our virtual medical education platform and made it more convenient for studying the impact of real-time help during virtual medical learning process.

6.1 The development of real-time help system

By this work, we verified it is possible to develop an online, accurate and quick response help information based on a specified medical related request. A classical medical case for training clinical reasoning skill has its own intrinsic structure, and this structure can be used to significantly speed up the content searching process online. In our real-time help system, we defined a set of keywords for a clinical case to fully use this structure so that our system has a very short response time. To explain this easily, let's take a brief review of a medical case again. It is composed of four parts: the first part is a brief introduction exposing the age, gender, main complain of the patient in the case, then several true clinical findings (partially like the term "analysis" used in Hypocrates+) are listed to build the second part; these clinical findings could be the symptoms or signs of the patient, findings from a physical examination by a doctor, part of laboratory test results, or even a former diagnosis and treatment by other doctors. The case manager, usually a medical educator, will decide and design some of all this information will be exposed to a medical student, some won't. Given all exposed information, a medical student will be asked what the most valuable information that he or she would like to know in first time so as to make a correct and precise diagnosis he or she believes. The case manager will expose the true information to the student if available. This step could last for several rounds until to a pre-set limit or the student believes he or she has enough information for providing a diagnosis and treatment plan. Then the student will choose a best main diagnosis from a diagnosis list in third part and choose one or several treatments from a treatment list in the fourth and also the last part. Sometimes based on the design of a clinical case, a correct diagnosis will be shown to the student if a wrong diagnosis is chosen. In Hypocrates and Hypocrates+, we call these four parts as "patient information", "analysis", "diagnosis" and "action". All the information in a clinical case points to a very important information: the name of the diagnosis of the patient in the case. This is so important that we use it as our case keywords. Besides, information from different sections usually has little relationship with each other except the case keyword, this could extremely reduce the range of

search in a suggested Wiki page. For example, if the agent believes there should be a help about an information in “analysis” section, the real-time help system will focus on some specific sections in Wiki pages, it will probably ignore the diagnosis and treatment sections. The idea came from the “category and indexing” techniques in information retrieval, which is quite reasonable and efficient. And the results from both internal test and public experiment support this.

6.2 Benefits and disadvantages of real-time help system for Hypocrates+

Current experiment results show that frustration value, which measures the negative emotion of a platform user, is likely to be reduced after a help content is presented to the user. Statistical analysis shows that we cannot conclude that help content provided by real-time help system helps reduce negative emotion based on current experiment results, which came from only 5 participants. To make the conclusion clearer, more participants are required.

There are also some disadvantages of the real-time help system. Current experiment result shows that even if most of the experiment participants think the help information is very accurate, some think information is less useful, which we have discussed this issue in previous chapter. Possible solutions for this could be: improving our agent so that it provides help information when participants really need it; finding a better resource database other than Wiki pages; developing an algorithm to deeply explore the real reason that makes the participant stuck in a point and generate more decent “other keywords”. For example, if a medical student chooses a wrong diagnosis A instead of the correct diagnosis B, it is better to set our “other keywords” be “the difference between A and B” than just set it be “A”. In medicine, this is an extra section called “differential diagnosis” in a clinical case; unfortunately, most Wikipedia pages don’t have this section as they are created and maintained by publics and for publics as well. To successfully apply more decent “other keywords”, we also need a pre-trained natural language model focused on Medicine, large amount of clinical case databases with labeled data for fine tuning the model.

Future work

Although the real-time help system is a component of Hypocrates+, it is indeed a small independent question and answer (QA) system specialized in medical education attached to Wikipedia. One of the future works on this system could be building an independent and specific medical database on a certain level of medical education in a certain topic of medicine, then optimizing the real-time help system on this database. Textbooks are surely very good choices to be a source of medical knowledge for junior medical students; General Internal Medicine and General Surgery could be two best topics to be started with. Some specialized topics, such as Obstetrics and Gynecology, Pediatrics and Ophthalmology could also be a good choice.

The transition from using Wiki pages to using other resources doesn't change the architecture of real-time help system very much; it only needs the resource to have similar structure as a Wiki page. If the resources are organized by different topics and each of which is composed by several sections, the real-time help system can easily adapt this change.

As to Hypocrates+ itself, future work could be focused on: optimizing clinical cases especially in the analysis part; importing image and video resources other than just texts to provide more vivid information in an virtual environment; enriching the interacting approaches in virtual environments, such as gesture; or even developing new medical education method other than clinical cases. For example, we could build a 3D virtual organ with basic or advanced functionalities for medical students to interact. The case manager may ask a student point out where a specific named blood vessel is or use a virtual ophthalmoscope to observe and describe what a given virtual fundus looks like, and so on.

Références bibliographiques

Ben Abdessalem, Hamdi, Alexie Byrns, Marc Cuesta, Valeria Manera, Philippe Robert, Marie-Andrée Bruneau, Sylvie Belleville, and Claude Frasson. 2020. “Application of Virtual Travel for Alzheimer’s Disease.” *9th International Conference on Sensor Networks* 1 (February): 52–60. https://doi.org/10.5220/0008976700520060.

Ben Abdessalem, Hamdi, and Claude Frasson. 2017. “Real-Time Brain Assessment for Adaptive Virtual Reality Game: A Neurofeedback Approach.” *Lecture Notes in Computer Science.* 10512 (September): 133–43. https://doi.org/10.1007/978-3-319-67615-9_12.

Bohil, Corey, Bradly Alicea, and Frank Biocca. 2011. “Virtual Reality in Neuroscience Research and Therapy.” *Nature Reviews. Neuroscience* 12 (November): 752–62. https://doi.org/10.1038/nrn3122.

Bordes, Antoine, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. “A Semantic Matching Energy Function for Learning with Multi-Relational Data.” *Machine Learning* 94 (2): 233–59. https://doi.org/10.1007/s10994-013-5363-6.

Braun, Leah Theresa, Benedikt Lenzer, Jan Kiesewetter, Martin R. Fischer, and Ralf Schmidmaier. 2018. “How Case Representations of Medical Students Change during Case Processing – Results of a Qualitative Study.” *GMS Journal for Medical Education* 35 (3): Doc41. https://doi.org/10.3205/zma001187.

Brin, Sergey, and Lawrence Page. 1998. “The Anatomy of a Large-Scale Hypertextual Web Search Engine.” *Computer Networks and ISDN Systems* 30 (1–7): 107–17. https://doi.org/10.1016/S0169-7552(98)00110-X.

Castillo, Carlos. 2005. “Effective Web Crawling.” *ACM SIGIR Forum* 39 (1): 55–56. https://doi.org/10.1145/1067268.1067287.

Cho, Junghoo. 2003. “Effective Page Refresh Policies for Web Crawlers.” *ACM Transactions on Database Systems* 28 (4): 390–426. https://doi.org/10.1145/958942.958945.

Cho, Junghoo, and Hector Garcia-Molina. 2002. “Parallel Crawlers.” In *Proceedings of the 11th International Conference on World Wide Web*, 124–135. WWW ’02. Honolulu, Hawaii, USA: Association for Computing Machinery. https://doi.org/10.1145/511446.511464.

Clarke, Theresa B., and Irvine Clarke. 2014. “A Competitive and Experiential Assignment in Search Engine Optimization Strategy.” *Marketing Education Review* 24 (1): 25–30. https://doi.org/10.2753/MER1052-8008240104.

Crowe, Sarah, Kathrin Cresswell, Ann Robertson, Guro Huby, Anthony Avery, and Aziz Sheikh. 2011. “The Case Study Approach.” *BMC Medical Research Methodology* 11 (June): 100. https://doi.org/10.1186/1471-2288-11-100.

DeYoe, Edgar A., and Ryan V. Raut. 2014. “Visual Mapping Using BOLD FMRI.” *Neuroimaging Clinics of North America* 24 (4): 573–84. https://doi.org/10.1016/j.nic.2014.08.001.

Dill, Stephen, Ravi Kumar, Kevin S. Mccurley, Sridhar Rajagopalan, D. Sivakumar, and Andrew Tomkins. 2002. “Self-Similarity in the Web.” *ACM Transactions on Internet Technology (TOIT)* 2 (3): 205–23. https://doi.org/10.1145/572326.572328.

Dong, Yang, Sammy Xiaoying Wu, Weisha Wang, and Shuna Peng. 2019. “Is the Student-Centered Learning Style More Effective Than the Teacher-Student Double-Centered Learning Style in Improving Reading Performance?” *Frontiers in Psychology* 10. https://doi.org/10.3389/fpsyg.2019.02630.

Dyer, Elizabeth, Barbara J. Swartzlander, and Marilyn R. Gugliucci. 2018. “Using Virtual Reality in Medical Education to Teach Empathy.” *Journal of the Medical Library Association: JMLA* 106 (4): 498–500. https://doi.org/10.5195/jmla.2018.518.

Fuller, A. W. 1906. “MODERN MEDICAL EDUCATION.” *The Lancet* 167 (4300): 254–254. https://doi.org/10.1016/S0140-6736(01)80627-9.

Fulton, John F. 1953. “History of Medical Education.” *British Medical Journal* 2 (4834): 457–61.

Grady, Cheryl L., Anthony R. McIntosh, M. Natasha Rajah, and Fergus I. M. Craik. 1998. “Neural Correlates of the Episodic Encoding of Pictures and Words.” *Proceedings of the National Academy of Sciences* 95 (5): 2703–8. https://doi.org/10.1073/pnas.95.5.2703.

Gruppen, Larry D. 2017. “Clinical Reasoning: Defining It, Teaching It, Assessing It, Studying It.” *Western Journal of Emergency Medicine* 18 (1): 4–7. https://doi.org/10.5811/westjem.2016.11.33191.

Harris, Zellig S. 1954. “Distributional Structure.” *WORD* 10 (2–3): 146–62. https://doi.org/10.1080/00437956.1954.11659520.

Heydon, Allan, and Marc Najork. 1999. “Mercator: A Scalable, Extensible Web Crawler.” *World Wide Web* 2 (4): 219–29.

Html Agility pack. 2020. “Html Agility Pack | Html Agility Pack.” 2020. https://html-agility-pack.net/.

Khin-Htun, Swe, and Anisa Kushairi. 2019. “Twelve Tips for Developing Clinical Reasoning Skills in the Pre-Clinical and Clinical Stages of Medical School.” *Medical Teacher* 41 (9): 1007–11. https://doi.org/10.1080/0142159X.2018.1502418.

Korfhage, Robert R. 1997. *Information Storage and Retrieval*. USA: John Wiley & Sons, Inc.

Lawrence, Steve, and C. Lee Giles. 1999. “Accessibility of Information on the Web.” *Nature* 400 (6740): 107–107. https://doi.org/10.1038/21987.

Majerus, Steve. 2013. “Language Repetition and Short-Term Memory: An Integrative Framework.” *Frontiers in Human Neuroscience* 7: 357–72. https://doi.org/10.3389/fnhum.2013.00357.

McLean, Susan F. 2016. “Case-Based Learning and Its Application in Medical and Health-Care Fields: A Review of Worldwide Literature.” *Journal of Medical Education and Curricular Development* 3 (April): 39–49. https://doi.org/10.4137/JMECD.S20377.

MediaWiki. 2020. “API:Opensearch - MediaWiki.” 2020. https://www.mediawiki.org/wiki/API:Opensearch.

Queen’s University. 2018. “Queen’s University Launches Canada’s First VR Medical Training Centre.” Faculty of Health Sciences | Queen’s University. October 9, 2018. https://healthsci.queensu.ca/administration/announcements/queens-university-launches-canadas-first-vr-medical-training-centre.

Salton, Gerard, and Christopher Buckley. 1988. “Term-Weighting Approaches in Automatic Text Retrieval.” *Information Processing & Management* 24 (5): 513–23. https://doi.org/10.1016/0306-4573(88)90021-0.

SPARCK JONES, KAREN. 1972. “A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL.” *Journal of Documentation* 28 (1): 11–21. https://doi.org/10.1108/eb026526.

Stahnisch, Frank W., and Marja Verhoef. 2012. “The Flexner Report of 1910 and Its Impact on Complementary and Alternative Medicine and Psychiatry in North America in the 20th Century.” *Evidence-Based Complementary and Alternative Medicine: ECAM* 2012: 647896. https://doi.org/10.1155/2012/647896.

Tabibian, Behzad, Utkarsh Upadhyay, Abir De, Ali Zarezade, Bernhard Schölkopf, and Manuel Gomez-Rodriguez. 2019. “Enhancing Human Learning via Spaced Repetition Optimization.” *Proceedings of the National Academy of Sciences* 116 (10): 3988–93. https://doi.org/10.1073/pnas.1815156116.

Thompson, Richard F., and Jeansok J. Kim. 1996. “Memory Systems in the Brain and Localization of a Memory.” *Proceedings of the National Academy of Sciences* 93 (24): 13438–44. https://doi.org/10.1073/pnas.93.24.13438.

Wikipedia. 2020. “Wikipedia.” 2020. https://www.wikipedia.org/.

Yew, Elaine H. J., and Karen Goh. 2016. “Problem-Based Learning: An Overview of Its Process and Impact on Learning.” *Health Professions Education* 2 (2): 75–79. https://doi.org/10.1016/j.hpe.2016.01.004.

Zhang, Jin, and Alexandra Dimitroff. 2005. “The Impact of Webpage Content Characteristics on Webpage Visibility in Search Engine Results (Part I).” *Information Processing & Management* 41 (3): 665–90. https://doi.org/10.1016/j.ipm.2003.12.001.

Annexes

Annex1 An XML segment of a clinical case

```
<Scenario id_scenario="0">
      <name_Patient>M.A.</name_Patient>
      <age_Patient>50</age_Patient>
      <gender_Scenario>Homme</gender_Scenario>
      <desc_Scenario>N/A</desc_Scenario>
      <keywords_Scenario>hypertension</keywords_Scenario>
      <type_Scenario>0</type_Scenario>
      <TAB_SYM>
        <Symptome id_symptome="1" nom_symptome="Tension" desc_symptome="N/A"
val_symptome="165/95 mm Hg" type_symptome="N/A" keywords_symptome="Tension artérielle" />
        <Symptome id_symptome="2" nom_symptome="Céphalées matinales" desc_symptome="N/A"
val_symptome="N/A" type_symptome="N/A" keywords_symptome="Céphalées; matinales" />
        <Symptome id_symptome="3" nom_symptome="Bourdonnement d'oreille"
desc_symptome="N/A" val_symptome="N/A" type_symptome="N/A" keywords_symptome="Acouphène"
/>
        <Symptome id_symptome="4" nom_symptome="Vision de mouches volantes"
desc_symptome="N/A" val_symptome="N/A" type_symptome="N/A" keywords_symptome="mouches
volantes" />
      </TAB_SYM>
      <TAB_AN>
        <Analyse id_analyse="1" nom_analyse="Interrogation" type_analyse="N/A"
desc_analyse="N/A" vrai_fausse_analyse="1" keywords_analyse="interrogation">
          <RES_AN>
            <ResultatAnalyse id_res_analyse="1" nom_res_analyse="Tabagisme 15
paquets/année" val_res_analyse="N/A" desc_analyse="N/A" />
            <ResultatAnalyse id_res_analyse="2" nom_res_analyse="Mère hypertendue et
diabétique" val_res_analyse="N/A" desc_analyse="N/A" />
            <ResultatAnalyse id_res_analyse="3" nom_res_analyse="Frère 55 ans hypertendue"
val_res_analyse="N/A" desc_analyse="N/A" />
          </RES_AN>
        </Analyse>
        <Analyse id_analyse="2" nom_analyse="Bandelette urinaire" type_analyse="N/A"
desc_analyse="N/A" vrai_fausse_analyse="0" keywords_analyse="Bandelette urinaire">
          <RES_AN>
            <ResultatAnalyse id_res_analyse="1" nom_res_analyse="Cette analyse n'est pas
nécessaire" val_res_analyse="N/A" desc_analyse="N/A" />
          </RES_AN>
        </Analyse>
        <Analyse id_analyse="3" nom_analyse="Pression arterielle" type_analyse="N/A"
desc_analyse="N/A" vrai_fausse_analyse="1" keywords_analyse="pression arterielle">
          <RES_AN>
            <ResultatAnalyse id_res_analyse="1" nom_res_analyse="155/95 mm Hg"
val_res_analyse="N/A" desc_analyse="N/A" />
          </RES_AN>
        </Analyse>
        <Analyse id_analyse="4" nom_analyse="Examen ORL " type_analyse="N/A"
desc_analyse="N/A" vrai_fausse_analyse="0" keywords_analyse="ORL">
          <RES_AN>
            <ResultatAnalyse id_res_analyse="1" nom_res_analyse="Cette analyse n'est pas
nécessaire" val_res_analyse="N/A" desc_analyse="N/A" />
```

```
          </RES_AN>
        </Analyse>
        <Analyse id_analyse="5" nom_analyse="ECG" type_analyse="N/A" desc_analyse="N/A"
vrai_fausse_analyse="0" keywords_analyse="ECG">
          <RES_AN>
            <ResultatAnalyse id_res_analyse="1" nom_res_analyse="Cette analyse n'est pas
nécessaire" val_res_analyse="130/min" desc_analyse="N/A" />
          </RES_AN>
        </Analyse>
        <Analyse id_analyse="6" nom_analyse="Pouls" type_analyse="N/A" desc_analyse="N/A"
vrai_fausse_analyse="1" keywords_analyse="Pouls">
          <RES_AN>
            <ResultatAnalyse id_res_analyse="1" nom_res_analyse="Pouls périphériques sont
tous perçus" val_res_analyse="N/A" desc_analyse="N/A" />
          </RES_AN>
        </Analyse>
        <Analyse id_analyse="7" nom_analyse="Souffle abdominal" type_analyse="N/A"
desc_analyse="N/A" vrai_fausse_analyse="1" keywords_analyse="Souffle abdominal">
          <RES_AN>
            <ResultatAnalyse id_res_analyse="1" nom_res_analyse="Pas de souffle
abdominnal" val_res_analyse="N/A" desc_analyse="N/A" />
          </RES_AN>
        </Analyse>
      </TAB_AN>
      <diag_choisi id_diagnostic="1" nom_diagnostic="HTA essentielle"
desc_diagnostic="N/A" vrai_faux_diagnostic="1"
keywords_diagnostic="hypertension;essentielle" />
      <TAB_DIAG>
        <Diagnostic id_diagnostic="1" nom_diagnostic="HTA essentielle"
desc_diagnostic="N/A" vrai_faux_diagnostic="1"
keywords_diagnostic="hypertension;essentielle" />
        <Diagnostic id_diagnostic="2" nom_diagnostic="HTA blouse blanche"
desc_diagnostic="N/A" vrai_faux_diagnostic="0"
keywords_diagnostic="effet_«_blouse_blanche_»" />
        <Diagnostic id_diagnostic="3" nom_diagnostic="Angine" desc_diagnostic="N/A"
vrai_faux_diagnostic="0" keywords_diagnostic="angine de poitrine" />
        <Diagnostic id_diagnostic="4" nom_diagnostic="HTA maligne " desc_diagnostic="N/A"
vrai_faux_diagnostic="0" keywords_diagnostic="hypertension maligne" />
      </TAB_DIAG>
      <TAB_ACT>
        <Action id_action="1" num_etape_action="N/A" nom_action="Bilan biologique"
desc_action="N/A" vrai_fausse_action="1" keywords_action="bilan biologique" />
        <Action id_action="2" num_etape_action="N/A" nom_action="ECG" desc_action="N/A"
vrai_fausse_action="1" keywords_action="ECG" />
        <Action id_action="3" num_etape_action="N/A" nom_action="Conseils hygiéno-
diététiques" desc_action="N/A" vrai_fausse_action="1" keywords_action="régime alimentaire
et mode de vie" />
        <Action id_action="4" num_etape_action="N/A" nom_action="Demande de contrôle"
desc_action="N/A" vrai_fausse_action="1" keywords_action="demande de contrôl" />
        <Action id_action="5" num_etape_action="N/A" nom_action="Traitement Antibiotique"
desc_action="N/A" vrai_fausse_action="0" keywords_action="antibiotique" />
        <Action id_action="6" num_etape_action="N/A" nom_action="Psychothérapie"
desc_action="N/A" vrai_fausse_action="0" keywords_action="psychothérapie" />
        <Action id_action="7" num_etape_action="N/A" nom_action="Repos" desc_action="N/A"
vrai_fausse_action="0" keywords_action="Repos_(physique)" />
        <Action id_action="8" num_etape_action="N/A" nom_action="Taitement symptomatique"
desc_action="N/A" vrai_fausse_action="0" keywords_action="symptomatique" />
```

```
        <Action id_action="9" num_etape_action="N/A" nom_action="Traitement antalgique"
desc_action="N/A" vrai_fausse_action="0" keywords_action="antalgique" />
        <Action id_action="10" num_etape_action="N/A" nom_action="Anticoagulats"
desc_action="N/A" vrai_fausse_action="0" keywords_action="Anticoagulant" />
        <Action id_action="11" num_etape_action="N/A" nom_action="Hospitalisation"
desc_action="N/A" vrai_fausse_action="0" keywords_action="Hospitalisation" />
        <Action id_action="12" num_etape_action="N/A" nom_action="Amoxiciline"
desc_action="N/A" vrai_fausse_action="0" keywords_action="Amoxiciline" />
      </TAB_ACT>
    </Scenario>
```

Annex2 Recruitment poster for experiments

Département d'informatique et de recherche opérationnelle

Participants recherchés

Compensation de 20$

Interaction avec un environnement en réalité virtuelle

En bref :

- Durée : 45-60mn
- Lieu : Laboratoire HERON
- Adresse : Local 3332 - Pavillon André-Aisenstadt (3ème étage) - Université de Montréal
- Participants recherchés : Étudiant(e)s en médecine (1er / 2e année)

Vous interagissez avec un environnement de réalité virtuelle dans lequel vous allez résoudre des cas médicaux. L'expérience se déroule dans un casque FOVE VR et pendant ce temps, un casque sans fil va enregistrer vos données cérébrales.

Pour plus d'informations ou prendre un rendez-vous, contactez Qiang Ye.

Vous pouvez aussi prendre un rendez-vous sur notre site Web.

Annex3 Paper accepted by 2nd BFAL international conference

The BFAL2020 (2nd BFAL) Conference proceedings are published in the LNCS series of Springer: Springer's Lecture Notes in Computer Science series (LNCS) No. 12462

Author Contribution:

Qiang Ye developed the real-time help system of Hypocrates+, modified the clinical cases for the experiments, performed the experiments, participated in the discussion, and wrote related sections of the paper.

Hamdi Ben Abdessalem developed the neural agent of Hypocrates+, directed the experiments, collected and analyzed the data, directed the discussion, and wrote related sections of the paper.

Marwa Boukadida developed the virtual environment of Hypocrates+, prepared the clinical cases for the experiments, participated in the experiments and the discussion, and wrote related sections of the paper.

Hypocrates+: A Virtual Reality Medical Education Platform with Intelligent Real-time Help System

Qiang Ye, Hamdi Ben Abdessalem, Marwa Boukadida

Abstract. Virtual reality, an immersing and interactive information communication technology, is changing the fundamental of medical education by creating more astonishing vivid realistic clinical environment and cases. Here, we proposed a virtual reality medical education platform with intelligent real-time emotion evaluation and help system, named Hypocrates+. By this platform, medical students can gain their medical knowledge and experience through well designed virtual clinical cases and environments with less frustration. Experiments shows that the overall mean frustration before getting the help was 0.53 and the mean frustration after was 0.50; better performance is related to lower frustration. We concluded that Hypocrates+ is a good start for developing a popular virtual medical education platform.
Keywords: Virtual Reality, Medical Education, EEG, Real-time Help System, Intelligent Agent

1. Introduction

Virtual reality (VR) technology is changing the style of medical education [1] with its powerful capability of simulating the clinical environments and clinical cases. By using a virtual medical education platform, medical students have an opportunity to do clinical reasoning on virtual cases instead of on real patient. It is much safer to make mistakes on virtual patients during the long-time medical learning process.

In our previous work [2], we proposed a virtual reality platform with several well-designed virtual clinical environments and clinical cases, Hypocrates, where a medical student can experience virtual clinical cases and learn medical knowledge while a real-time emotion analysis of the student can also be performed by the electroencephalographic (EEG) signal [3] collected from the users' interaction as well. Our further research shows that, during interaction between a medical student and the platform, positive emotion has a strong correlation to better learning result, and vice versa. Can we improve our platform to reduce students' negative emotion so that they can interact with the platform and learn medical knowledge in a more peaceful and positive mood? Is it possible to help students in order to reduce their negative emotions?

In this article, we imported a real-time emotion analysis and a real-time help system to our existed Hypocrates, and we call it Hypocrates+. When interacting with Hypocrates+, medical students' EEG signals are collected simultaneously and sent to the real-time emotion analysis system, where a decision of whether a help is needed for the student is making. If such a help is needed, the help system will provide the student related useful medical information intelligently acquired from internet in real time.

2. Related Works

2.1 Real-time Help Systems

Learning new things usually is a long-term process. Repetition is required, help is often useful, and correction is essential when mistakes are made. In which situation and how frequently a repetition, help, or a correction is required is a subject of research. A real-time help system provides timely information to learners when they get stuck during a learning process, which often saves time for

learners. Moreover, a real-time help system may also provide targeted and personalized help information if an intelligent analysis component is equipped.

Due to the development of information technology, medical education is changing its style: from traditional paper books and real clinical practice to electronical resources and virtual standardized patient model [4]. By interacting with well-designed electronical and virtual clinical cases, students are more easily to master medical knowledge and apply them to real clinical cases correctly [5]. Students are also aware of medical errors; however, remaining tensions may limit learning [6]. In such situation, a real-time help system may help maintain students' positive emotion.

2.2 Virtual Reality

Virtual reality is more and more used in different domains and it proved its efficiency due to its advantages. Its main advantage is the possibility to isolate the user from any external visual distraction and thus making them believe they are in a real world [7]. Some researchers generated sensorimotor illusions giving the user a sense of presence in the environment by manipulating multimodal stimulus inputs thanks to virtual reality [8]. This technology has been used to treat various psychological disorders including brain damage [9], anxiety disorders [10] and alleviation of fear [11]. Moreover, some researchers designed real-time editable virtual reality environment in order to change its parameters according to the user's brain activity [12].

3. "Hypocrates+": Methodology

3.1 Environment: "Hypocrates+"

We created an interactive virtual reality environment. The choice of virtual reality is based on the fact that it ensures a total isolation from external distraction, as seen in a previous section, and thus increase concentration on the task. This environment is a dynamic system able to present various medical cases in real-time. The student initially goes through an introductory scene in which we explain the tasks to be done. Subsequently, they are immersed in a virtual operating

room or a doctor's office, depending on the type of the medical case. The participant interacts with the virtual environment through a virtual reality headset and a gamepad.

For each medical case, we start by exposing a panel in which we display to the student information about the patient and their symptoms, a reliability score gauge, an "Analysis" and a "Diagnosis" buttons. The score of reliability gauge decreases every time the student selects a wrong choice.

The student must read the displayed symptoms, then they can click on one of the buttons ("Analysis" or "Diagnosis" buttons) depending on whether they want to ask for analysis or directly choose a diagnosis. If the student asks for analysis, a panel containing a checklist of analysis will appear where they can check one or more analysis and then can see a panel displaying the results by clicking on the "Results" button. If the student clicks on the "Diagnosis" button, a panel displaying four medical diagnosis appears and they must choose the correct one by clicking on it. Once the diagnosis is selected a series of panels appear one at a time. Each one of these panels contains 3 possible actions (one correct, and two wrong). The number of actions' panels depends on the number of actions to perform in the medical case being resolved. For example, if the number of actions needed to be performed is three, three panels will appear, one by one, containing each one three actions. Every time the student chooses an action the next actions' panel will appear.

3.2 Real-time help system

Within Hypocrates+, the real-time help service functions as a server, providing help content to its client: the other parts of the whole platform. Fig.1 briefly shows the architecture of the real-time help system and how it communicates with its client. It includes three main components: a communication component, an intelligent content generating component, and a log component. When a client asks for a help, it sends a message to the real-time help server. When receiving this message, the communication component enqueues this message to a queue; meanwhile, the content generating component continuously checks the message queue, dequeues a request, and intelligently produces response content. Once a response content is prepared, communication component will send the content back to the client who requested it. All communication history is

recorded by the log component for future analysis. The whole system is capable to provide the service stably and robustly to multiple clients.

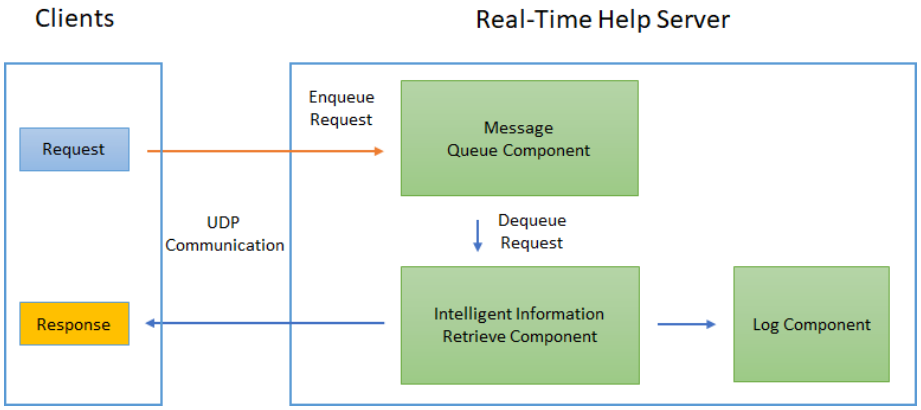

Fig. 1. Architecture of the Real-Time Help System for Hypocrates+

Intelligent content generating component is another major module of the real-time help server. It parses the request, invokes Wiki search API to acquire most related Wiki pages, analyzes the pages and selects most meaningful, high-related text within a certain length as a response and sends it back to the endpoint who requests it. Fig. 2 briefly shows how this component works. Having Considered the characteristics of the description of a clinical disease in a Wiki Page, we designed and implemented section recommendation and best response decision intelligent algorithms to produce best response accurately and quickly.

3.3 Neural Agent

The neural agent in Hypocrates+ is responsible for communicating with the real-time help system and the virtual environment. By receiving the EEG signal of a Hypocrates+ user who is interacting with the virtual environment, the neural agent calculates the emotional state of the user and makes decisions on whether a real-time help request is needed according to an inner intelligent algorithm. If this request is needed, the agent will send a request to the real-time help server, then receive a help information and display it on the virtual environment to the user.

4. Experiments

We conducted experiments involving 5 medical students (4 females) in order to test our approach. The experimental protocol is the following. In the first step of the experiment, the

participant signs an ethic form which explains the study and fills a pre-session form. In the second step, we install an Emotiv EPOC headset. In the third step, the participant is equipped with the FOVE VR headset and we give him a wireless gamepad to interact with the environment. After these steps, we start the "Hypocrates" environment, the Real-time help service and the Neural Agent.

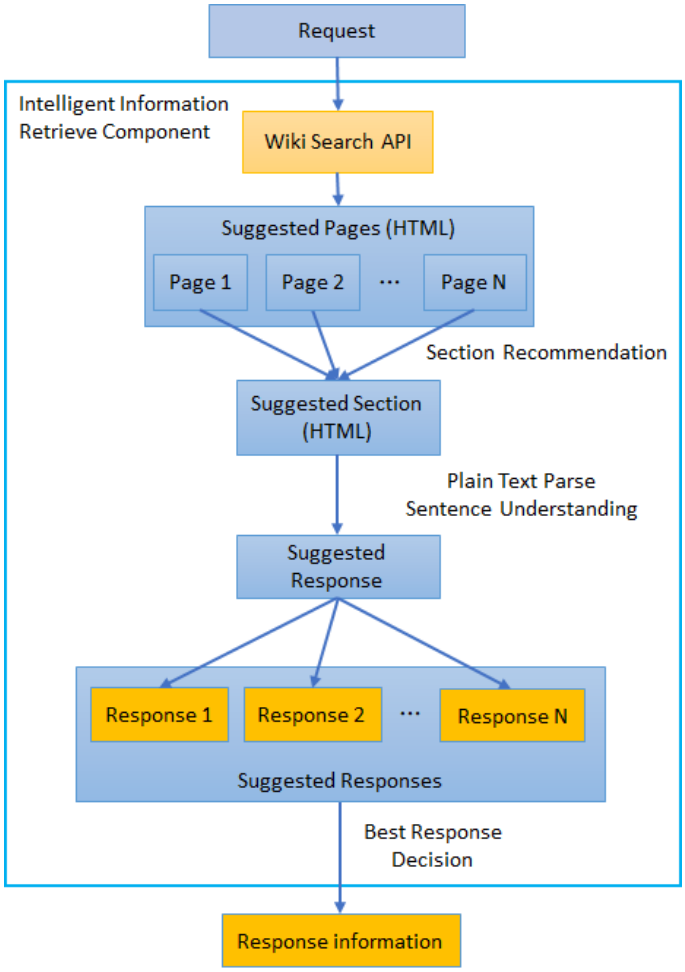

Fig. 2. Architecture of the Intelligent Content Generating Component

5. Results and Discussion

The objective of this study was to discover whether it is possible to help student in order to reduce their negative emotions. To this end, we analyzed the mean frustration level of the participants

before and after they received a help information. The results show that the overall mean frustration before getting the help was 0.53 and the mean frustration after was 0.50.

More detailed results are shown in Fig. 3 where we note that the frustration decreased for all the participants expect P2.

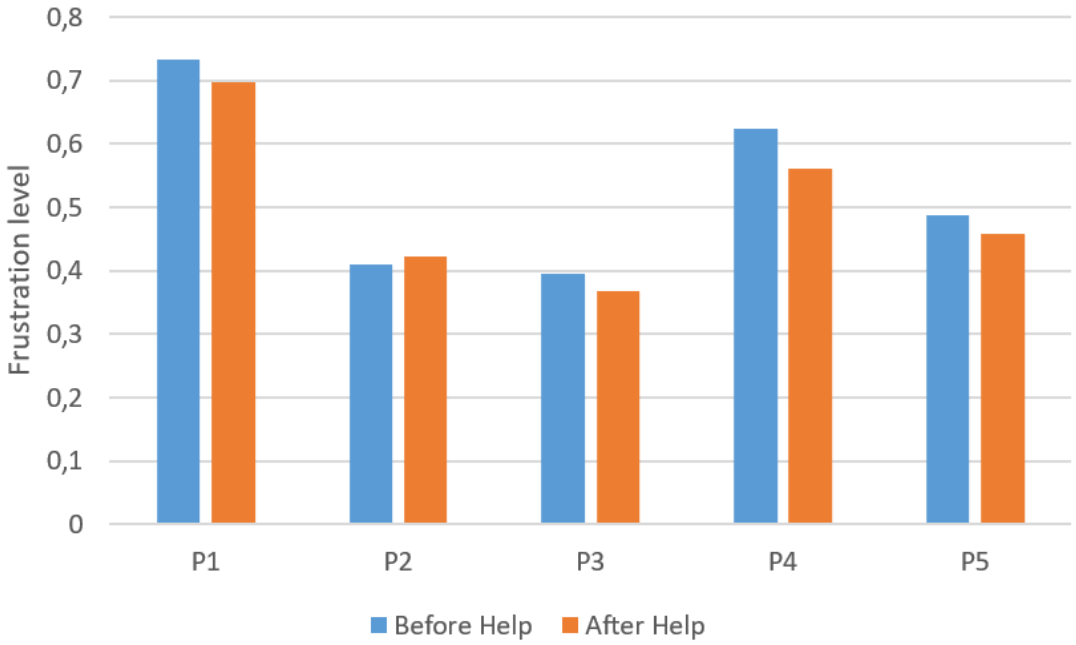

Fig. 3. Histogram of mean frustration level before and after the help

This effect obtained in our first analysis lead us to analyze the performance of each participants. The highest performance was made by the participant 1 with a mean score of 72.77% and the worst performance was made by participant 2 with a mean score of 55%. This result may be explained by the frustration that was not decreased by the help.

6. Conclusion

In this paper, we presented a novel approach to help medical students resolving medical cases by importing a real-time help service in our platform: Hypocrates+. We conducted experiments, and results show that the emotional state of a medical student is easy to keep stable and positive, which provides more peaceful and positive mood for learning. Besides, it is also possible to help reduce mistakes when the student makes clinical reasoning on virtual clinical cases presented by 3D

virtual environment. Hypocrates+ is a good start for developing a popular virtual medical education platform.

Acknowledgment

We acknowledge NSERC-CRD (National Science and Engineering Research Council-Cooperative Research Development) and BMU for funding this work.

References

1. Rishad Khan, Joanne Plahouras, Bradley C Johnston, Michael A Scaffidi, Samir C Grover, Catharine M Walsh. Virtual reality simulation training for health professions trainees in trainees in gastrointestinal endoscopy. Cochrane Database Syst Rev.2018 Aug;2018(8)

2. Ben Abdessalem, Hamdi, and Claude Frasson. 2017. "Real-Time Brain Assessment for Adaptive Virtual Reality Game: A Neurofeedback Approach." In, 133–43. https://doi.org/10.1007/978-3-319-67615-9_12.

3. McFarland, D.J., Sarnacki, W.A., Wolpaw, J.R.: Electroencephalographic (EEG) control of three-dimensional movement. J. Neural Eng. 7, 036007 (2010).

4. Joshua Moran, Gregory Briscoe, Stephanie Peglow. Current Technology in Advancing Medical Education: Perspectives for Learning and Providing Care. Academic Psychiatry. 42:796-799 2018.

5. Elizabeth Dyer, Barbara J. Swartzlander, Marilyn R. Gugliucci. Using Virtual Reality in Medical Education to Teach empathy. Virtual Projects. 518 498-500 2018.

6. Melissa A Fischer, MD, MEd, Kathleen M Mazor, EdD, Joann Baril, BS, Eric Alper, MD, Deborah DeMarco, MD, and Michele Pugnaire, MD. Learning from Mistakes - Factors that Influence How Students and Residents Learn from Medical Errors. J Gen Intern Med. 21(5), 419-423 2006.

7. Biocca, F.: The Cyborg's Dilemma: Progressive Embodiment in Virtual Environments [1]. J. Comput.-Mediat. Commun. 3, 0–0 (2006).

8. Bohil, C.J., Alicea, B., Biocca, F.A.: Virtual reality in neuroscience research and therapy. Nat. Rev. Neurosci. (2011).

9. Rose, F.D., Brooks, B.M., Rizzo, A.A.: Virtual Reality in Brain Damage Rehabilitation: Review. Cyberpsychol. Behav. 8, 241–262 (2005).

10. Gorini, A., Riva, G.: Virtual reality in anxiety disorders: the past and the future. Expert Rev. Neurother. 8, 215–233 (2008).

11. Alvarez, R.P., Johnson, L., Grillon, C.: Contextual-specificity of short-delay extinction in humans: Renewal of fear-potentiated startle in a virtual environment. Learn. Mem. 14, 247–253 (2007).

12. Ben Abdessalem, H., and Frasson, C. 2017. Real-time Brain Assessment for Adaptive Virtual Reality Game: A neurofeedback Approach. The First International Conference on Brain Function Assessment in Learning, 10512: 133-134. Patras, Greece: Springer International Publishing

Annex4 Detailed data for negative emotion analysis

Participant ID	Frustration Value		Decreased after Help
	Before Help	After Help	
P1	0.78092846	0.68505951	Yes
	0.68505951	0.64908450	Yes
	0.92109913	0.82390833	Yes
	0.82390833	0.81431957	Yes
	0.82390833	0.81431957	Yes
	0.59781625	0.38614740	Yes
	0.59781625	0.54225825	Yes
	0.57285094	0.57754803	No
	0.58435780	0.83503778	No
	0.63462958	0.70341894	No
	0.84685360	0.67438342	Yes
	0.92445331	0.86080479	Yes
P2	0.51067011	0.58476936	No
	0.58476936	0.53425806	Yes
	0.50921989	0.70765167	No
	0.42480848	0.46194929	No
	0.35891333	0.31114754	Yes
	0.37356228	0.45995032	No
	0.32314712	0.41473569	No
	0.25823610	0.12426512	Yes
	0.47777062	0.26280681	Yes
	0.27219448	0.35046885	No
P3	0.64455423	0.40918064	Yes
	0.35575333	0.44319585	No
	0.34740806	0.45558966	No
	0.40525725	0.21985588	Yes
	0.38635593	0.23327034	Yes
	0.37901750	0.40123240	No
	0.25227827	0.41462541	No
P4	0.77870453	0.66375803	Yes
	0.64158880	0.55633033	Yes
	0.61327151	0.54398305	Yes
	0.77988712	0.62397670	Yes
	0.75060358	0.44827024	Yes
	0.52510909	0.84870393	No
	0.62830218	0.61677762	Yes
	0.74890654	0.62715721	Yes

	0.54943784	0.49811336	Yes
	0.40845813	0.41908744	No
	0.50237581	0.59371134	No
	0.55233646	0.54912839	Yes
	0.73376647	0.55014083	Yes
	0.75779275	0.64816935	Yes
	0.60113809	0.46262994	Yes
	0.57807808	0.50533360	Yes
	0.44707859	0.37305585	Yes
P5	0.58480332	0.36802663	Yes
	0.51924192	0.84644959	No
	0.84644959	0.52149152	Yes
	0.42251693	0.35879639	Yes
	0.37005139	0.52357929	No
	0.41359862	0.47965828	No
	0.31524358	0.39610881	No
	0.53511878	0.35993015	Yes
	0.42093264	0.33696629	Yes
	0.44778478	0.38277903	Yes

Each frustration values in this table is the mean of 36 discrete instant frustration values in one second before or after help. Each instant frustration value, which is not shown in this table, is generated by auto analyzing the EEG signals through a software tool: Emotiv SDK.