A SPEECH RECOGNITION ENABLED AMBULANCE CALL REPORT SYSTEM FOR PARAMEDICS

By
Rafal Zydowicz

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

Your file Votre référence
ISBN: 978-0-494-79593-4
Our file Notre référence
ISBN: 978-0-494-79593-4

Abstract

We have designed an Ambulance Call Report (ACR) system that uses speech recognition. Our system uses a client connected wirelessly to a computer which performs speech recognition using the CMU Sphinx4 speech recognizer. Each field in the ACR is represented by either a grammar or a language model. Because much of the ACR is codified and the language used is specific to paramedics, speech recognition can achieve sufficiently high accuracy to potentially be used in a deployed system. This thesis describes the current state of Paramedic Services in Ottawa to identify areas that may be improved. It also describes past and present speech recognition systems to understand the capabilities of speech recognition. This is followed by a detailed description of our system design and implementation. Finally, a limited study was conducted with four paramedics to determine the success of our user-interface design, ACR accuracy, Word Error Rates, and user satisfaction.

Acknowledgements

I would like to acknowledge the following people:

- Dr. Jorg-Rudiger Sack for the patience and guidance he has provided throughout my studies.
- Dr. Avi Parush and Leslie MacDonald-Hicks for helping me with my application for ethics approval.
- Kaveh Shahbaz and Masoud T. Omran for letting me vent my frustrations.
- Adrian Batos-Parac for helping me design requirements for my system.
- Eric Hopkins for helping me whenever I had questions about paramedics and for organizing volunteers for my study.
- Raj Wasson for helping me with the study.
- My mum, dad, and brother.

Thank you all.

Contents

Chapter 1

Introduction

One of the most popular buzz words in healthcare of the last decade is e-Health. Although definitions vary broadly [56], Eysenbach's [27] definition is perhaps the most inclusive. He defines it as

> an emerging field in the intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies. In a broader sense, the term characterizes not only a technical development, but also a state-of-mind, a way of thinking, an attitude, and a commitment for networked, global thinking, to improve health care locally, regionally, and worldwide by using information and communication technology.

By employing advances in technology, patient care can in theory be improved by streamlining inefficient processes while simultaneously reducing costs. Unfortunately, in Canada, it is also viewed as money pit which has failed to produce any tangible results [58]. Part of the reason for this view as argued in [58] is that expensive proprietary solutions offer negligible functionality benefits over free open-source solutions. While it is true that many open-source solutions rival and even outperform proprietary solutions, they rarely come with a similar level of support, if any. Except for a small subset of well managed, often corporate sponsored open-source projects, many open-source projects have unpredictable development cycles and can be abandoned at any time. Few institutions have the resources available to maintain development in such events, or wish to invest in a solution that may need to be replaced in the near future. A more likely reason for the negative view of e-Health is that many institutions view it as a "silver bullet"

where it clearly is not. If an institution has work flow problems, simply sticking an e-Health solution in the middle is unlikely to bring any benefits regardless if the solution is proprietary or open-source. If an institution has multiple legacy systems that cannot inter-operate, an e-Health solution will not be used to its full potential. It would be wiser to first prepare the legacy systems or replace them so that they could work with an e-Health solution when it is introduced. For e-Health solutions to provide the expected benefits, they must be introduced where appropriate.

1.1 Thesis Rationale

Budget constraints limit the number of paramedics that Paramedic Services can hire and are forcing them to seek innovative solutions to improve efficiency. The recent recession is likely to lead into even tighter budgets over the next few years. In a 2008 audit [55], it is revealed that the Ottawa Paramedic Service has maintained consistently high turnaround times even after the migration to electronic ACRs. One factor contributing to these turnaround times is the time required to complete Ambulance Call Reports (ACR). Paramedics must fill out an ACR for every call they respond to. It is a time consuming task which is often completed after the patient has been transferred to the care of the hospital. As a result, vital patient data is often communicated verbally causing a potential increase in the chance of errors and a lower standard of care for the patient. Augmenting ACRs with speech recognition offers the possibility to reduce the time required to complete ACRs.

Using speech recognition, our system requires less searching and selecting than the Rampart CST system currently used in Ottawa and substantially less typing than the iMedic system used by many services in Ontario. These benefits allow paramedics using our system to complete ACRs faster than with current systems. As a result, the likelihood of ACRs being completed before transfer of care also increases thus reducing verbal communication errors and allowing physicians to make more informed decisions. Finally, having ACRs completed quicker reduces the turnaround time of ambulances, increasing their availability.

While speech recognition enabled Electronic Health Records (EHR) exist, they are designed to be used in a different manner and by different users than ACRs. EHRs are often completed after a patient has received care. Immediate patient care is not affected by the speed with which EHRs are completed. Because of this lack of immediacy, EHRs can be completed at the convenience of the physician on a stationary device with a large display and abundant processing power. On the

other hand, ACRs contain information that can severely affect patient care upon transfer of care. For ACRs to be completed as quickly as possible, they must be portable and work in a diverse range of environments. They are therefore limited by a small display and processing power. To our knowledge, our system is the only ACR system specifically designed to be speech recognition enabled.

1.2 Related Work

Speech recognition systems have been studied in the medical field for many years. Much of this research is focused disproportionately in radiology [19, 40, 72, 59, 61], but also includes dentistry [50], pediatrics [36], and various other specialties [8]. A literature review of speech recognition systems in healthcare found many studies to be vague on details such as the types of users and the environments they were used in [23]. As a result, it is often difficult to determine why some implementations were successful and others were not.

As mentioned, much of the literature on speech recognition in the medical fields pertains specifically to radiologists with wide ranging opinions on its benefits. Various studies show that shifting the transcription work from transcriptionists to radiologists results in cost increases, significant increases of up to 50% in the time required fill out reports, and longer turnaround times [61, 59]. In [59], only 3% of reports completed by transcriptionists contained errors after being signed out, where as 35% of reports completed by the speech recognition system contained errors after being signed out. Speech recognition accuracy was 96% [59]. The authors [59] do note that while the errors are typically minor, "malpractice attorneys will often note typos and grammatical errors in reports as examples of a radiologists carelessness or disinterest." In another study on radiologists, speech recognition systems were shown to dramatically reduce turn around times and high levels of satisfaction [40].

Outside of radiology, Mohr et al. [48] studied the effects of speech recognition on the productivity of secretaries and transcriptionists working in endocrinology and psychiatry. Reports were transcribed manually start to finish or with the help of a speech recognition system. The results revealed that productivity actually worsened when using the speech recognition system. Such a result is somewhat expected as the transcriptions generated by the speech recognizer required proof reading, a potentially time consuming task even if error rates are low.

The authors of [76] showed that speech recognizers in the medical domain can achieve recognition rates of 97% in 1999. The authors compared three medical vocabularies containing word pronunciations and found that the least expensive

required over 400 additions to achieve similar a recognition rate as the two more expensive vocabularies. Further, they state that results were only achieved if corrections were made as they occurred and the users spoke in a clear voice.

Detailed comparisons of competing speech recognition systems for use in the medical field are frustratingly rare. A comparison of three leading speech recognition systems in 2000 [22] compared IBM ViaVoice with General Medicine Vocabulary, Dragon NaturallySpeaking Medical Suite, and L&H Voice Xpress for Medicine. For the study, clinicians were asked to fill out a medical discharge summary and a medical progress note, which combined, were approximately 1,000 words. The WERs ranged from 7.0% with IBM's system to 15.2% with Dragon's system. These results are consistent with WERs of other large vocabulary speech recognition systems [71, 75]. In 2010, of these three systems only Dragon NaturallySpeaking remains on the market.

The University of Pittsburgh Medical Center has been using speech recognition enabled Electronic Health Record (EHR) systems since 2001. As of 2010, their current EHR system is VoiceOver Enterprise which uses Dragon NaturallySpeaking Medical Edition. It is truly a large vocabulary speech recognition system with a vocabulary of over 260,000 words. A 2010 report of their system shows it to be a success [38]. Turnaround times for reports have decreased 81% since the speech recognition system has been implemented with the number of reports signed out within a day increasing by 89% [38]. Even more surprising is the reduction in transcription errors. They report an almost 50% reduction in transcription errors as a result of the speech recognition system [38]. One likely reason for the success of the speech recognition at UPMC is the use of templates guiding the user through the process of filling out a report. This instantaneous feedback can reduce errors by clearly displaying what needs to be input.

1.3 Contributions

We have shown that using current technology, it is possible to design and build a speech recognition enabled ACR system. Our system consists of a Client, Server, and Viewer. The Client is a small lightweight touch screen device worn by the paramedic. The paramedic uses the Client to fill out the ACR using speech recognition, but can also fill it out manually using dialogs and an on-screen keyboard. The Client only records audio and sends it wirelessly to the Server for recognition. The speech recognizer used in our system is CMU Sphinx4. The results are sent back to the Client and the displayed ACR is updated. The Viewer is used by physicians and nurses to view the completed ACRs and runs on a separate

computer.

Because of the processing and memory requirements of speech recognition, to maximize speech recognition accuracy, a speech enabled ACR system should use a Client-Server design. We have shown that using a low latency WLAN connection, the delay for transmission is insignificant compared to the time required for speech recognition processing. The size and resolution of the display of the device used by the paramedic greatly affect the design of the user interface (UI). With the 3.3 inch screen device we used, only 3 fields of the ACR can be displayed at any one time without making selection error prone and the text too small. Fields in our system used either a grammar or a language model to constrain the acceptable utterances. We have shown that 134 of the fields in the 5 major sections of an ACR can be constrained by a grammar and only 16 require a language model. Because the strict syntax of grammars can result in higher accuracies, we expected high overall ACR accuracies.

Using our system, we achieved average ACR accuracies of 90%. Accuracies were calculated by dividing the number of correctly filled out fields of the ACR by the total number of fields to be filled out. When only fields using a grammar were taken into account, average ACR accuracies of 95% were achieved. When only fields using a language model with acceptable Word Error Rates (WER) of 0% were taken into account, average ACR accuracies of 63% were achieved. When the acceptable WER was increased to 15%, average ACR accuracies increased to 69%. While accuracies are often measured in WERs. measuring the WER of fields that use grammars, as is the case for a majority of the fields in our system, provides little useful information as they are most often either correct or incorrect. This is a result of the strict syntax expressed by the grammars. WERs were calculated for the fields in our system that use a language model. For these fields, the average WER was 22%. This number is higher than expected as a comparison of three commercial speech recognition systems used in the medical field showed WERs from 7.0% to 15.2% [22]. It is also much higher than the approximately 3% WER for vocabularies of less than 1,000 words described in the Sphinx4 white paper [71]. Our study revealed a flaw in our correction functionality which adversely affected WERs in fields that use language models. A direct comparison to other systems is difficult as studies do not always specify how accuracy is calculated. nor do they always provide it. For example, in [59], it is not explained how the report accuracy figure of 96% is calculated. In [38], Hyunseok Peter Kang et al. only show an approximately 50% decrease in errors using the speech recognition system compared to dictation without specifying how errors were calculated.

Our study revealed that fields that used a language model had lower than

expected accuracies. Two contributors were identified. First, user pronunciations differed from those in the vocabulary. As was shown in [76], missing or incorrect pronunciations can have drastic effects on speech recognition accuracy. Second, correction functionality which allowed the user to utter "clear" and "undo" allowing a paramedic to clear a field and undo the most recent addition to a field respectively, did not perform as expected. Both commands were often mis-recognized leading to compounded errors. Our study also revealed that users often did not follow syntax as described by hints in every field.

Our study revealed that fields that used a grammar achieved the expected high accuracies. Since we discovered that approximately 90% of the fields of an ACR can be constrained using a grammar, this implied that highly accurate speech recognition enabled ACRs are possible. One of the reasons why we attempted a system that uses speech recognition for filling out ACRs is because large proportions are already codified and thus expressible by grammars. Paper ACRs were designed this way to reduce errors and simplify data analysis. By taking advantage of this, speech recognition should feel natural for the paramedic. Finally, the negatives observed in the study can mostly be resolved without major changes to our design.

While our system is not the only possible design, even according to our requirements list, it is a functional system that allows paramedics to fill out ACRs in their entirety. Because our design decisions, difficulties, and limitations are all discussed in detail and supported by a study using paramedics, future research in this area can refer to our work.

1.4 Methodologies

Developing a system such as ours required knowledge of diverse fields. Knowledge of Paramedic Services and the paramedics themselves was one of the first steps taken and critical to its success. The most recent audit [55] of the Paramedic Services in Ottawa was analyzed along with related literature. Two current ACR systems were studied: Rampart CST which is used in Ottawa, and iMedic which is used by various Paramedic Services in Ontario. Over the course of development of our system, we met numerous paramedics on several occasions. These meetings allowed us to better understand the paramedics job, their frustrations with current systems, and what they expect from a speech recognition enabled ACR system. These meetings were invaluable as there is often a disconnect between how tasks should be performed, and how they are actually performed.

Knowledge of speech recognition was also critical to the success of our system.

Understanding the strengths and weaknesses of speech recognizers allowed us to design realistic requirements for our system. Strengths such as higher achievable accuracies using grammars and easier processing of grammars using tags, could be exploited, while weaknesses such as lower achievable accuracies using language models and the high processor and memory requirements, could be mitigated.

Finally, a major aspect of development was the actual programming. While high-level design decisions simplified this aspect, it was still a challenging task that required many months of work to complete. Making sure the Client-Server connection functioned as expected and implementing the multithreaded aspects of our system were major difficulties encountered.

After development, we conducted a small usability study based on the Thinking-Aloud protocol as described by Boren and Ramey [16]. After being granted ethics approval by the Carleton University Ethics Board, we found four paramedics willing to participate in our study. While the study was designed to reveal system usability and design errors, it was flexible enough to also reveal ACR accuracies.

1.5 Outline

Following this introductory chapter, the structure of this thesis is as follows. In Chapter 2, we discuss the current state of Paramedic Services in Ottawa with comparisons to other services in Ontario. In Chapter 3, we discuss speech recognition by briefly discussing its history, the breakthroughs, and the modern Sphinx4 speech recognizer we use for our system. In Chapter 4, we discuss our system in detail with explanations for design decisions including a discussion of difficulties and limitations. In Chapter 5, we discuss a study that was conducted with four paramedics using our system while thinking-aloud. The objective of the study is to determine the success of our user-interface design, the ACR accuracy and word error rates achievable using our system, and user satisfaction. Finally, in Chapter 6, we discuss what we have learned and our thoughts on using speech recognition to fill out ACRs.

Chapter 2

Paramedic Services: A Description of Services in Ottawa and Ontario

It is 8:00 pm and you are having dinner with your wife at your favorite restaurant. A patron nearby suddenly starts grabbing his chest; he is having a heart attack. The lady he is with screams for someone to call 911. At 8:08 pm the paramedics arrive and immediately begin stabilizing the patient. The patient is loaded onto the ambulance and is sped off to the Ottawa Heart Institute for further care. Upon arriving at the hospital, the paramedics communicate to the receiving physician pertinent details about the patient. The patient recently had surgery and was on blood clotting medication. Knowing this information, the physician can decide what dose of blood thinning medication should still be administered. The paramedics having completed the ACR, ready the ambulance and prepare for the next call.

The fact that paramedics arrived within 8 minutes of the call is not a coincidence. Paramedic Services have response time targets that they constantly strive to attain. For an urban area, as is the case in this scenario, the target is 8:59. Sending the patient directly to the Ottawa Heart Institute instead of the nearest hospital is a conscious decision that requires communication between the dispatch center, the paramedics, and the hospital itself. The pertinent details of the patient are communicated to the physician because paramedics are required to fill out ACRs for every call they handle.

Paramedic Services provide the quality of care they do because of a well defined process. We must study this process if we are to design a successful speech recognition enabled ACR system.

2.1 Determining Call Priority

Paramedic Services have a finite set of resources. One of their most important resources is naturally the ambulances themselves. Ambulance resources can be limited in two ways. There can either be not enough ambulances in service, or they can be preoccupied with existing emergencies. In either case, they are a limited resource and must be carefully administered. If two emergencies occur at the same time where one involves a broken limb and the other a cardiac arrest, there should be a way to prioritize who is responded to first.

2.1.1 DPCI

In Ontario, excluding Toronto and the Niagara Region, the Dispatch Priority Card Index (DPCI) is used to determine call priority. In DPCI, the call is given a priority from 1 to 4. They are [3]:

Code 4 Life threatening call. Life and /or limb threatening in nature.

Code 3 Urgent call. Non-life threatening, however, call is of serious enough in nature.

Code 2 Scheduled call. Generally non-urgent/non-life threatening.

Code 1 Deferrable call. Non-urgent/non life threatening calls.

A code 4, the most serious, also necessitates a "lights and sirens" response, while the remaining levels do not. While a "lights and sirens" response will potentially get the paramedics to the patient quicker, it puts them and the public at greater risk to accidents. The system must also accurately determine priority, otherwise resources may be mis-allocated. Returning to the previous example, if the broken limb emergency and cardiac arrest emergency are both classified as code 4, the system has not done a good enough job. Therefore, accuracy of such a system is crucial to be of any benefit.

Accuracy is the key to a good call priority system. DPCI has four broadly defined levels of priority. Unfortunately, the current criteria for classification determine the vast amount of calls to be of a code 4 nature. In 2007, 76% of calls to Ottawa Paramedic Services were rated as a code 4 [55]. A 2008 audit of the Ottawa Paramedic Services [55] states that

> the implications of this trend are critical and extremely problematic from a public safety perspective – there is no effective triaged response taking place.

Although this number has been steadily rising from 64% in 2001, in 2006 the amount of code 3 calls was almost halved while code 4 calls noticed a large upwards spike [55]. It is possible that a DPCI update, or a change in interpretation policies, is the reason for this anomaly. Compared to other Paramedic Services that use alternatives to DPCI, even the 2001 numbers are too high. This endangers the lives of paramedics, bystanders, and most critically, even patients since ambulances may be mis-allocated. The 2008 audit [55] recommends DPCI be replaced with another call priority system such as AMPDS.

2.1.2 AMPDS

Toronto and the Niagara region do not use DPCI as the rest of the Ontario does. Instead they use the Advanced Medical Priority Dispatch System (AMPDS), which is widely deployed in Canada, the United States and various other countries. A major advantage of AMPDS over DPCI is that triage is much more accurate. A caller is placed into one of five priority levels. In order of severity, from least to most, they are: Alpha, Bravo, Charlie, Delta, and Echo. The placement of an individual is automatic based on a decision tree that the dispatcher follows, although the option does exist for override. Carefully worded questions such as "is the person completely awake," are simple enough for callers to comprehend, especially under duress, while guaranteeing identical results regardless of dispatcher [57]. The current version 11 has approximately 300 different questions which the dispatcher may ask. The combination of extra levels and a complex decision system allows AMPDS to drastically reduce the amount of calls which would require a "lights and sirens" response. This allows better allocation of resources, and most importantly, reduces the chances of available ambulances reaching critically low levels or even zero availability. AMPDS is not a perfect dispatch priority solution. The National Academy of Emergency Medical Dispatch which maintains it releases regular updates to improve the system.

2.2 Triage: CTAS

Triage is not definitive and is in fact performed on a patient at least twice before arrival at the hospital; once by the dispatchers, and again by the paramedics who

arrive at the scene. The Canadian Triage and Acuity Scale (CTAS) is the method used by paramedics to determine triage. Unlike DPCI, it has a scale from 1 to 5, where 1 is the most severe and 5 the least severe. A CTAS 1 dictates a "lights and sirens" return to the nearest hospital, unless there is a reason to bypass for reasons such as hospital specialization. Both DPCI and AMPDS provide triage, but their effectiveness is limited by the fact that it is determined by the information the caller provides to the dispatcher. The duress the caller is often in, plus the fact that they are rarely qualified to make a proper prognosis means that such systems will naturally err on the side of caution. This contributes to the difference in triage between dispatch priority systems and on scene triage performed by the paramedics using CTAS. Specifically, many code 4 emergencies are evaluated by the paramedics to not be CTAS 1 or CTAS 2 which are the most severe [55]. Besides the mis-allocation of ambulance resources, another mis-allocated resource are the Advanced Care Paramedics (ACP). Because of the poor triage performance of DPCI, ACPs are being dispatched to scenes which do not require them, making them less available for emergencies where they are actually needed. In Ottawa, ACPs were available for approximately 70% of all code 4 calls, a number which is steadily falling [55]. There is an ongoing debate whether ACPs actually improve chances of patient survival. In cases of trauma, recent research shows patient mortality either remaining the same, and in some cases actually increasing when provided Advanced Life Support [66]. On the other hand, a program undertaken in Ottawa in 2005 where paramedics trained in Advanced Cardiac Life Support (ACLS) take STEMI patients directly to the heart institute where percutaneous coronary intervention (PCI) is performed showed a 50% decrease in mortality in its first year [42].

2.3 Paramedics in Ontario

In Ontario, there are three types of paramedics: Primary Care Paramedics (PCP), Advanced Care Paramedics (ACP), and Critical Care Paramedics (CCP). Primary Care Paramedics provide Basic Life Support (BLS) while Advanced Care Paramedics can provide also Advanced Life Support (ALS). PCPs and ACPs are the paramedics which are dispatched to 911 calls. Critical Care Paramedics are most often used for patient transfers between hospitals and have additional skills on top of what a PCP and ACP would have. The type of care a PCP, ACP, or CCP can provide is dictated by the Ambulance Act of Ontario [53]. A PCP can perform the following [54]:

- conduct patient assessments.
- provide basic airway management.
- administer oxygen by demand, by bag-valve-mask or basic mechanical ventilation.
- perform cardio pulmonary resuscitation (CPR).
- provide basic trauma care (e.g. spinal and wound care, limb immobilization/traction).
- administer symptom relief medications and perform semi-automated external defibrillation (SAED) when certified by a base hospital director.

An ACP can perform the tasks of a PCP plus the following [54]:

- provide advanced airway management, including oral and nasotracheal intubation.
- perform laryngoscopy and removal of foreign body obstruction using forceps.
- provide basic field mechanical ventilation.
- conduct 12 lead ECG interpretation.
- administer symptom relief medications and perform manual defibrillation when certified by a base hospital director.

A CCP can perform the tasks of a PCP and ACP plus the following [54]:

- administer any drug listed in Schedule 1 and 2, when certified by a base hospital medical director.
- perform additional advanced airway procedures, such as needle thoracostomy, cricothyroidotomy.
- interpret x-rays and lab blood valves.

To become a PCP requires the completion of a two-year program at a college. An ACP requires a post-diploma program which takes an extra year. There are currently 390 paramedics serving the Ottawa area [4]. A worrying trend is that from 2001 to 2007, the number of calls has increased at twice the rate of new paramedics [55]. In fact, paramedics in Ottawa handle significantly more calls than they do paramedics in Toronto, Montreal, Edmonton, and Calgary [55].

2.4 Time Targets

In an emergency, few will disagree that prompt care reduces morbidity and mortality. As a result, targets have been set for the various events in the sequence of events which begin with the call to 911. The events are [55]:

T0 Call received by CACC

T1 Call taker does call triage, dispatcher selects ambulance

T2 Paramedics notified given priority and pick up location

T3 Travel time to call

T4 Arrived scene time

T5 Depart scene travel to hospital

T6 Arrive at hospital

T7 Depart hospital

T Max Arrive at station

Two targets which receive the majority of attention are: the time it takes for an ambulance to reach the scene, and the time it takes for the patient to receive definitive care.

2.4.1 Response Times

Ottawa Paramedic Services have two targets for the time to scene, T2-T4 inclusive, depending on area density. In high-density urban areas, defined as "areas with greater than or equal to 24 calls per sq km per year in groups of 6 contiguous sq km," [5] the target is 8:59. In low-density rural areas, which cover everything else, the target is 15:59. The targets are becoming increasingly difficult to achieve because of the limitations of DPCI, a lack of paramedics, and other inefficiencies. In 2007, only 67% of urban calls and 77% of rural calls met their targets [55]. In the first quarter of 2009, the 90th percentile response time for life threatening calls was 13:08, far short of 8:59 [6]. A study [13] of 746 patients with life-threatening emergencies in the United States showed that adherence to a response time of below 10:59 had no statistically significant effect on mortality rates and the frequency of critical procedural interventions. This is further supported by [29],

which argues that these strict response times should be made flexible as they are based on mortality data on cardiac arrest patients in the 1970s and were simply extended to all emergencies. More flexible reponse times based on call severity may result in better management of ambulances.

2.4.2 Time to Definitive Care

Another major target is the time to definitive care, which is T0-T6 inclusive plus offload time. The target is 60 minutes and is often referred to as the "golden hour". The benefits of adhering to this target are inconclusive. A literary review by [44] revealed little support for the target for trauma patients, while [15] revealed that performing thrombolytic treatment in acute myocardial infarction patients within two hours reduced mortality by 44% compared to patients who received the treatment between two and six hours. A somewhat surprising fact is that a major contributor to this time is not within the T0-T6, but rather the offload time. From Jan - Jun 2007 in Ottawa, the 90th percentile time to offload a patient was 1:50:24 [55]. While this number is certainly too high, a closer analysis of what offloading entails shows that this number can also be misleading. According to [55],

> when measuring hospital offload times, the Paramedic Service currently calculates the duration from unit arrival at hospital until the unit reports itself as clear for the next call. The Paramedic Service does not record the actual time of patient transfer to the hospital – typically understood as when the emergency department triage nurse first examines the patient.

The offload time captures the time it takes to fill out the ACR and prepare the ambulance for the next call on top of the time it takes for transfer of care. Filling out the ACR and preparing the ambulance can easily account for half this time. An initiative taken in Ottawa hospitals is to hire dedicated nurses which the paramedics can offload the patients to. This does not improve the "time to definitive care", but does at least improve turnaround time for the paramedics.

2.4.3 Turnaround Time

The turnaround time is described as the moment an ambulance responds to a call to the moment the ambulance is ready to respond to another call. Thus, this covers events T2-T7. The amount of time T2-T6 takes is limited by the pace

at which paramedics can do their job safely and properly. On the other hand, the amount of time T7 takes can be improved. Being able to transfer care to a dedicated nurse as quickly as possible is one of these improvements. The process of filling out the ambulance call report (ACR) is another target for improvement. The paramedic may begin filling it out whenever he chooses, but often begins once the ambulance leaves the scene and is travelling towards the hospital. The process then often continues after the patient has been transferred to the care of the hospital. This delay can significantly reduce the amount of ambulances available.

2.5 Ambulance Call Reports

When a patient is offloaded at a hospital, the paramedics need to provide the receiving party, who is most often a nurse, patient information such as potential allergies, treatments or drugs administered, and other vital information. This information is passed on verbally and also by providing a copy of the Ambulance Call Report (ACR). The ACR is a comprehensive report divided into five main sections administration, clinical information, physical exam, clinical treatment / procedures and results, and general administration. Appendix A shows an ACR and how its fields can be filled out. Although the paramedics can begin filling out the report the moment they get dispatched, it is often completed once they arrive at the hospital. This limits its immediate usefulness upon arrival as pertinent information will be passed on verbally which can result in omissions of vital information. ACRs were originally, and up to a few years ago in Ottawa, filled out using pen and paper. Four copies of the ACR were made: one for the patient chart, one for the billing office, one for the base hospital, and finally one more for the ambulance service. A paper solution creates many challenges. Hand writing is not always legible, and can become even less legible on a third generation copy. If not enough pressure is used, some information may simply not go through to the underlying copies. Similar incidents may be described in a different language causing difficulties in statistical analysis which is used to optimize performance. Finally, both the paramedic services and the hospitals transfer these paper reports to electronic systems causing a potential for further mistakes and inefficient use of resources. Such limitations are being overcome using electronic ambulance call reports.

New solutions are continuously being developed with varying success at improving the speed and accuracy with which these reports can be completed. Rampart CST guides the user through a decision tree, while iMedic allows the user

more freedom in filling out the report Each method has its strengths and weaknesses These solutions also provide other benefits such as immediate availability of data to hospitals, system interoperability, and safety.

2.5.1 Rampart CST

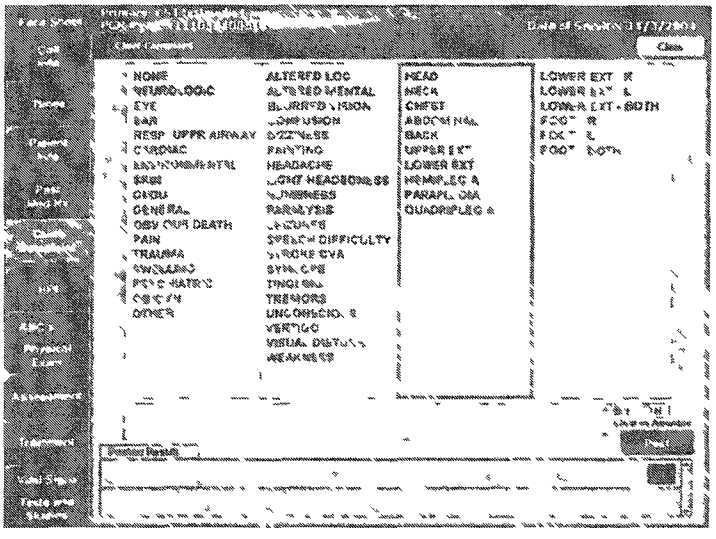


Figure 2.1: A screen capture of the chief complaint section in Rampart CST.

The Ottawa Paramedic Service uses the Rampart Clinical Support Tool in place of a paper Ambulance Call Report Using a laptop or tablet, a paramedic uses Rampart CST to create a Patient Care Record (PCR) There is a fundamental difference in how a PCR is filled out compared to a paper ACR The difference is not that it is electronic compared to pen and paper, but rather free form writing is replaced by a "builder" concept. Much of the functionality of Rampart CST is handled by dedicated builder screens, which allow the paramedics to fill out the PCR using buttons and lists The use of buttons and lists to fill out forms is a fundamental concept in graphical user interfaces. Almost anyone who has ever used the internet will also be familiar with such widgets When a paramedic wants to fill out a chief complaint using the builder, he/she must first begin with a broad list of choices from a list This opens another list of more detailed choices which opens another list of even more detailed options. This process often concludes within 3 to 4 steps For example, selecting chief complaint (see Figure 2 1),

cardiac, and chest pain produces the sentence "Patient is complaining of chest pain." Such a design produces data which is very easy to data mine and can be of great benefit to paramedic services and hospitals. It also can produce frustrations for the actual users. Part of the problem is that many paramedics have been using paper ACRs for many years and simply need to adjust to a new system. Although good software should try to minimize such drastic concept changes, if the benefits far outweigh the drawbacks, a case can be made for such a leap. A patient record, be it an ACR or PCR, needs to be completed both quickly and correctly so that nurses and physicians know vital patient information as soon as possible. Thus quickness and correctness should be two benefits of such an electronic system. Quickness is already hampered by the fact that many paramedics have still not adapted. In replacing unrestricted writing, the time spent writing is replaced by time spent searching. Correctness is potentially even more important than quickness. Medication errors are a common problem in hospitals across Canada and the United States and can lead to adverse drug effects on patients [70]. Further, they can also cause physicians and nurses to have high stress levels impacting their work [65]. A builder concept as used in Rampart CST eliminates spelling errors on the paramedic's part, but introduces target selection errors. It is more likely that a paramedic correctly writes down what they are thinking as opposed to selecting what they are thinking. Selecting from a list is made more difficult by the fact that the paramedic is forced to use the language provided. It is thus entirely possible that forcing the user to use lists to construct sentences can actually introduce more errors than unrestricted writing. Finally, the sentences created by the builders can become convoluted and difficult to comprehend for the receiving nurses and physicians. One major benefit not yet mentioned that improves both quickness and correctness is interoperability. With Rampart CST, an ECG can automatically upload data to the PCR. Unfortunately other readings such as blood pressure are still lacking interoperability. The result is a system that provides questionable benefits in quickness and correctness, which are the two most vital aspects of a patient record.

2.5.2 iMedic

A competing electronic Patient Care Record used in various municipalities in Ontario is iMedic ePCR. The major difference between it and Rampart CST is its interface. In essence, iMedic ePCR simply takes the paper Ambulance Call Report (ACR) and digitizes it. The benefits of such a system are that paramedics accustomed to the original forms should have little difficulty in adjusting to this

new format, while the benefits of interoperability remain available. Such a solution is not without drawbacks. Since the reports are filled out using unrestricted writing, data analysis becomes more difficult than in a more rigid solution as used in Rampart CST. This drawback is mitigated by the fact that analysis can be done over a much longer period of time that does not affect patient care.

Chapter 3

Speech Recognition: A Brief History & Current Technology

The earliest method of recording sound was by using a phonoautograph invented by Édouard-Léon Scott de Martinville in 1857. Unfortunately, it was only capable of recording sound with no ability to play it back. By the late 19th century, the record playing phonograph was invented; a device not far removed from the record players many people still have at home today. It functioned on a similar principle to the cylinder playing phonograph which preceded it, offering similar audio fidelity, but had the benefits of cheaper production and easier storage of the recorded media. As a result, it eventually became the dominant player. Improvements such as the switch from acoustic recording and reproduction to electrical recording and reproduction eventually lead to quality levels that even compact discs, a medium released one hundred years later, cannot match. Once we attained the ability to record and reproduce sound, and thereby speech, it was only natural to attempt to recognize what was being spoken.

Because speech recognition is an integral part of our system, we must understand it in sufficient detail to make good design decisions. We begin by studying early speech recognizers. While modern speech recognizers differ in some cases quite substantially from early speech recognizers, studying them as well allows us to avoid past mistakes while broadening our knowledge of many concepts that are still in use today. These concepts help us understand how modern speech recognizers such as CMU Sphinx4 function. We examine Sphinx4 in detail towards the end of this Chapter.

3.1 Early Speech Recognition History: Early Successes & Deadends

Potential solutions on how to best handle speech recognition have been described since at least the 1950s. In [31], the authors discuss the different linguistic units a speech recognizer can use. They provide four different units: phonemes, morphemes, words, and sentences. Naturally, they each have their advantages and disadvantages mostly in the form of a trade-off between storage requirements and accuracy. Phonemes are the most basic linguistic unit used to differentiate utterances. In General American English, there are 40 phonemes. Because the pronunciation of a phoneme is affected by its preceding and proceeding neighbours, phonemes are usually grouped into sequences of 3 phonemes called tri-phones. Thus, to model every possible combination requires 64,000 (40 * 40 * 40) tri-phones. Likewise, the pronunciation of words is also affected by its neighbours. Therefore, if word units are chosen, then for a vocabulary of 100,000 words, you have 10^{15} (100,000 * 100,000 * 100,000) words. Using phonemes as a linguistic unit requires multiple orders of magnitude less training data for speech recognition than using words or sentences as a linguistic unit. Naturally, this implies that the storage requirements are also that much lower. For these reasons, later research almost unanimously uses phonemes as the basic linguistic unit.

The earliest speech recognition systems concentrated on digit recognition. One such attempt was by K. H. Davis et al. from Bell Laboratories in the early 1950s [20]. The authors showed that they could achieve an accuracy varying from 97% to 99%, but only on speaker trained data. Using training data that was gathered from another speaker lowered accuracy to between 50% and 60%. A system such as this is known as speaker dependent as opposed to speaker independent systems that work without speaker specific training. Another limitation of this system was that the digits had to be spoken as isolated words. There are two main reasons for such a limitation. First, it allowed the recognizer to determine word boundaries. Second, it removed the effects of neighbouring words on pronunciation. In 1967, D.R. Reddy researched speech recognition using speaker dependent connected speech [62]. The vocabulary was still very small at fewer than thirty words, but did allow the formation of short sentences. Similar to most research in this era, segmentation of the input signal attempted to map segments to phonemes. Approximately 275 phonemes existed in the training data which were classified into one of four groups: stop sounds, fricative like sounds, nasal-liquid sounds, and vowel sounds. Although the research achieved an accuracy rate of 81% and most phoneme errors were at least in the proper group, it was

still heavily susceptible to any imperfections in the recording. Later, in 1969, R. De Mori et al. designed a small vocabulary speech recognizer capable of partial speaker independent recognition [21]. Using male speakers and computing the average characteristics of the pronunciations for the training data, they achieved accuracies from 85% to 99% depending on the speaker. In this case though, words still needed to be spoken in isolation and the vocabulary of the system was only 15 words.

In the same year, J. R. Pierce, another researcher from Bell Laboratories, published a letter that claimed to "examine both motivations and progress in the area of speech recognition," but was little more than an attack on the speech recognition community [60]. The following is often quoted:

> The attraction is perhaps similar to the attraction of schemes for turning water into gasoline, extracting gold from the sea, curing cancer, or going to the moon. One doesn't attract thoughtlessly given dollars by means of schemes for cutting the cost of soap by 10%. To sell suckers, one uses deceit and offers glamor. [60]

Pierce may have failed to comprehend that not all funding must go for research which has a high chance of success. He also did not have the hindsight of forty years to see what some unlikely areas of research have led to. While we may not be able to turn water into gasoline, the hydrogen extracted from water will likely power our cars in the very near future. The knowledge accumulated from cancer research has undoubtedly contributed to our understanding of the human body while maintaining steady progress in the search for a cure for cancer. Going to the moon may have been funded for political reasons, but did open the space age which the modern world is very dependent upon. While an argument can be made that all this knowledge could have been acquired in a more cost effective manner, there is also no guarantee that we would have had the motivation to acquire it in the first place. Regardless of the funding issues, Pierce's other arguments did highlight some real issues with current research in speech recognition. Twenty years of research produced limited results and some of the research during that era was of questionable quality by not taking into account related research. An explanation for this may be that many researchers simply believed that speech recognition was a simpler process than it actually was. In response to the letter several rebuttals were soon published [64, 28].

3.2 Modern Speech Recognition: A Way Forward

In 1971, the Advanced Research Projects Agency (ARPA), later known as DARPA, began a "five-year research and development program with the objective of obtaining a breakthrough in speech understanding capability that could then be used toward the development of practical man-machine communication systems." [39] Specifically, the objectives were to

> develop several speech understanding systems that accept continuous speech from many cooperative speakers of a General American dialect. Recordings were to be made in a quiet room using a good-quality microphone. Slight tuning of the system would be allowed to handle new speakers, but the users could be required to make only natural adaptations to the system. The language definition should include a slightly selected vocabulary of at least 1000 words and an artificial syntax appropriate to the limited task situation (e.g., a data management task). Less than 10% semantic error would be tolerated and the system would have to run in a few times real time using the next generation of computers... [39]

Four different groups delivered a system at the conclusion of the program. A system from Systems Development Corporation (SDC) [12], a system from Bolt Beranek and Newman Inc. named HWIM [73], a system from Carnegie Mellon University named Hearsay-II [26], and finally another system from Carnegie Mellon University named Harpy [46].

SDC's system was most similar to systems from the 1960s. The input signal was first parameterized and then segmented based on the relationships between the parameters and manners of articulation in humans. Each segment would be placed in a syllable class which represents similar sounding phonemes. A mapper would then attempt to match parts of the phonetic transcription (i.e., the syllable classes) to potential words in a left to right fashion. In an effort to increase the chance of recognition, probabilities were added to the potential phonetic construction of words, such as replacing similar sounding phonemes, so that direct matches were not required. No effort was made to understand the context of the words. As such, illogical sentences would be just as likely to be recognized as logical sentences. The system was the most poorly performing of the group with an utterance accuracy of only 24%. The ARPA study uses a branching factor to measure task difficulty. It defines it as the "number of words that [...] have to be considered at each point along the correct left-to-right path through

the syntactic production rules during the processing of a typical utterance." [39] For SDC's system, the average branching factor was 105.

From a design perspective, BBN's HWIM system had some potential advantages over SDC's system. While SDC's system contained lexical representations of words, which allowed for multiple pronunciations, the representations depended on a phonetic transcription that did not contain any added or removed phonemes. Any such errors in the phonetic transcription would result in a complete failure of recognition, especially if they occurred early in the utterance. Instead of working in a left to right fashion attempting to map sequential syllable classes to potential words, the HWIM would attempt to map any sequential syllable classes to potential words and then spread outwards until it found the most likely sentence. This process is also known as phrase-island generation. Further, a much larger set of rules are applied to potential pronunciations such that phonemes can be added or removed, especially at word boundaries. In other regards such as parameterization, segmentation, phoneme generation, HWIM was very similar to SDC's system. HWIM achieved a much higher utterance accuracy of 44, but came at a cost of the highest branching factor of 195.

The concepts behind CMU's Hearsay-II had many similarities to BBN's HWIM. This fact was supported by the researchers themselves [26]. The input signal would be parameterized and segmented into pieces of similar acoustic properties. These segments would then be mapped to potential syllable classes. An area where Hearsay-II clearly outperforms HWIM is in phrase-island generation, which selects sequential syllable classes and hypothesizes potential sentences by spreading in both directions. Unlike HWIM which uses single word islands, Hearsay-II uses multi word islands. The main difference between Hearsay-II and HWIM lies in the implementation. Hearsay -II uses a "blackboard" design with interconnected, but modular, "Knowledge Sources" (KS). These Knowledge Sources are responsible for all the stages of speech recognition. Having a modular design allows KSs to be easily interchanged or modified without affecting the functionality of the rest of the system. The benefits of such a design were realized in development when parts of CMU's Harpy were used in Hearsay-II [26]. CMU's Hearsay-II was the second highest performing system with utterance accuracy of 74 and a branching factor of 46.

The committee overseeing the program recommended two approaches for implementing a speech recognition system. The recommended approach was to "simplify the general speech recognition problem by finding ways to apply syntactic and semantic constraints," while the other approach was to "improve upon previous speech recognition capabilities." [39] SDC's system, BBN's HWIM, and

CMU's Hearsay-II were all based on the latter approach. Within the speech recognition community, there was a growing consensus that a novel approach may be necessary [28]. While performance varied between the three systems, most differences could be attributed to minor differences in implementation.

CMU's Harpy was the only system in the program that attempted a novel approach to speech recognition. Instead of attempting to generate phonetic transcriptions directly from the input signal, Harpy used Hidden Markov Models (HMM) to determine the probability that a given sequence of segments mapped to a phoneme. Each phoneme would be described by a HMM, while multiple HMMs could be combined to form words. This greatly simplified the acoustic processing stage of speech recognition. Each segment of the input signal would be parameterized into 14 linear prediction coefficients. Starting at the beginning of the utterance and working forward, every possible sentence would be hypothesized according to a specified limit defined as the beam length. Each word would be given a score based on how well it matched the phonetic transcription and the sentence with the highest combined word score would be returned as the final result. A strict grammar limited the number of potential sentence hypothesizes. The difficulty in such a design is that it requires vast amounts of training data to account for as much variability in pronunciation as possible. Regardless, Harpy was the only system to achieve ARPA's program objectives. Its utterance accuracy was 95% with a branching factor of 33. Figure 3.1 highlights how Harpy met the objectives. Because of its significance in speech recognition, it should be noted that the Dragon system described in 1974 was a precursor to Harpy [10].

GOAL (Nov., 1971)	Harpy (Nov., 1976)
ACCEPT CONNECTED SPEECH	YES
FROM MANY	5 (3 MALE, 2 FEMALE)
COOPERATIVE SPEAKERS	YES
IN A QUIET ROOM	COMPUTER TERMINAL ROOM
USING A GOOD MICROPHONE	CLOSE-TALKING MICROPHONE
WITH SLIGHT TUNING/SPEAKER	20 TRAINING SENTENCES/TALKER
ACCEPTING 1000 WORDS	1011
USING AN ARTIFICIAL SYNTAX	AVG. BRANCHING FACTOR ~33
IN A CONSTRAINING TASK	DOCUMENT RETRIEVAL
YIELDING <10% SEMANTIC ERROR	5%
IN A FEW TIMES REAL TIME	80 TIMES REAL TIME
ON A 100 MIPS MACHINE	ON A .4 MIPS PDP-KA10
	USING 256K OF 36-BIT WORDS AND
	COSTING $5 PER SENTENCE PROCESSED

Figure 3.1: CMU Harpy meeting the objectives of the ARPA Speech Understanding project [39].

Early speech recognition solutions used grammars that specified syntax to limit the potential utterances that could be recognized in an effort to maximize accuracy and reduce computational time. Such solutions severely limited the flexibility of speech recognition. Utterances needed to be said according to rules; otherwise they would be rejected as out of grammar or an incorrect match would be forced. Such solutions worked for recognizing commands, but were not flexible enough for any larger vocabulary applications. One of the reasons why CMU's Harpy achieved such high accuracy is because of such strict grammars. A solution to grammars called language models was described by F. Jelinek in 1976 [37]. Instead of specific rules for each sentence, language models stored the transitional probabilities from one word to another. Such a design naturally gives properly formed sentences the highest probabilities, while maintaining enough flexibility to allow rarely occurring word combinations as well.

3.3 Current Speech Recognition: Sphinx4

Carnegie Mellon University did not stop work on speech recognition after the success of Harpy. Sphinx was released in the early 1990s [43]. It demonstrated the recent advances in speaker independent continuous speech large vocabulary speech recognition. Sphinx was followed by Sphinx2 in 2000 [34], which was followed by Sphinx3 in 2002 and finally Sphinx4 in 2004. Each version of Sphinx usually marked a change in the type of HMMs employed [71]. Sphinx employed discrete HMMs, Sphinx2 employed semi-continuous HMMs, and Sphinx3 employed continuous HMMs. Sphinx4, which also employed continuous HMMs, was written in Java to be a highly modular speech recognition system.

Sphinx4 is CMU's current state of the art speech recognizer. It is capable of speaker independent continuous large vocabulary speech recognition. Figure 3.2 shows the modular Sphinx4 architecture.

3.3.1 FrontEnd

The Sphinx4 FrontEnd module takes an input and outputs a sequence of features that the Decoder can use. The FrontEnd performs this task through DataProcessors. The input must go through many DataProcessors before a sequence of features are produced. Some of the more used DataProcessors are the Preemphasizer, Dither, RaisedCosineWindower, DiscreteFourierTransform, MelFrequencyFilterBank, DiscreteCosineTransform, BatchCMN, LiveCMN, and DeltasFeatureExtractor.

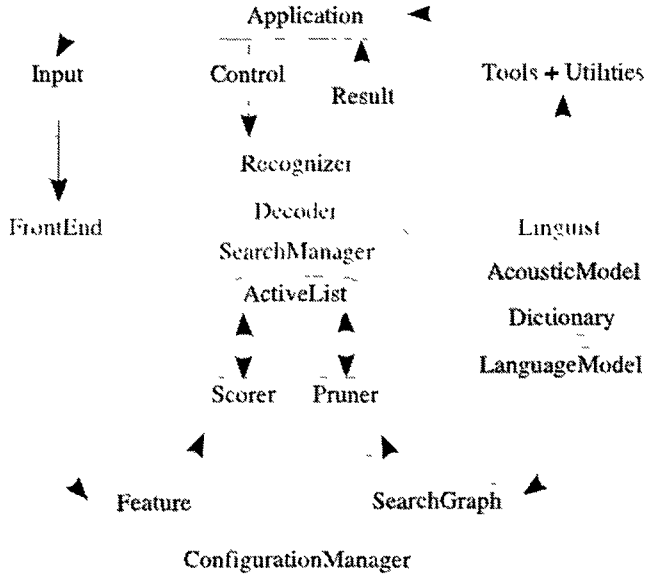


Figure 3.2: CMU Sphinx4 architecture [71].

Preemphasizer A high-pass filter that weakens the lower frequencies while letting the higher frequencies pass through untouched.

Dither Adds noise to the input to remove areas of zero energy which can cause difficulties for the Decoder.

RaisedCosineWindower Takes the input and converts it into overlapping frames, or windows in Sphinx4 parlance. These are usually Hamming windows, but other sizes can also be specified.

DiscreteFourierTransform Performs a Fast Fourier Transform on the individual windows of the input.

MelFrequencyFilterBank The input is put through a series of filters with a maximum frequency and minimum frequency and outputs a mel-spectrum. The number of filters and frequencies are affected by the sample rate of the input. At the recommended 16,000 Hz, by default, Sphinx4 uses 40 filters with a maximum frequency of 6800 Hz and a minimum frequency of 130 Hz.

DiscreteCosineTransform Given a mel-spectrum, outputs a 13 dimension Mel-Frequency Cepstral Coefficient (MFCC) vector.

LiveCMN Given a MFCC, performs Cepstral Mean Normalization by subtracting the mean calculated from all preceding MFCCs.

BatchCMN Given a MFCC, performs Cepstral Mean Normalization by subtracting the mean calculated from all the MFCCs from the complete input.

DeltasFeatureExtractor Given a MFCC, outputs a 39 dimension MFCC by computing the first and second delta of the original MFCC and adding it to the original MFCC.

3.3.2 Linguist

The Sphinx4 Linguist module, through the use of an AcousticModel, Dictionary, and LanguageModel, outputs a search graph that the Decoder can use to search for the most likely path resulting from the input features. The search graph contains states which can represent HMMs, phonemes, and words. States are connected by arcs that have transition probabilities. A basic search graph containing five HMMs is shown below in Figure 3.3.

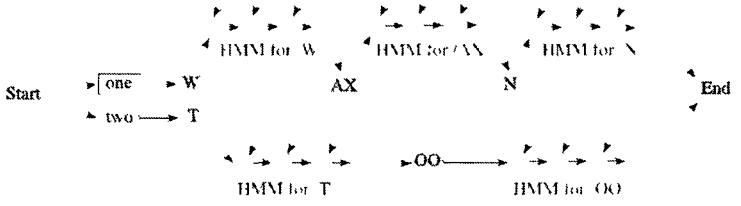


Figure 3.3: CMU Sphinx4 search graph [71].

Acoustic Model

Contains a Hidden Markov Model for every linguistic unit. If phonemes are selected as the linguistic unit, the acoustic model contains a HMM for every phoneme. Similarly, if the linguistic unit are words, then there would be a HMM for every word. Generating acoustic models is a very time consuming process.

Even if the speech corpus and corresponding recordings are provided, the generation of an acoustic model from this data is still a computational complex process. Larger linguistic units such as words require orders of magnitude more training data for adequate models as every word needs to be included at least once. Therefore, words are mostly used as units in applications with small vocabularies. Phonemes allow a much reduced amount of training data and are more flexible. While an acoustic model trained on data including every word will provide higher accuracies, missing words can still be recognized. Because the pronunciation of a phoneme is affected by its immediate neighbours, Sphinx4 uses tri-phones, which are sequence of three phonemes. This causes an explosion in the number of HMMs from approximately 40 to approximately 64,000. To reduce redundancy, the individual states of the HMMs can be shared. These states are defined as Senones and were first introduced in Sphinx2 [34]. Accuracy is increased by storing the placement of the tri-phone in the individual words.

Dictionary

Contains all the words in the language model and their pronunciation. The pronunciation is formed using the linguistic units chosen for training.

LanguageModel

The naming of this component is slightly misleading. Sphinx4 supports both n-gram language models described below and various grammars such as the Java Speech Grammar Format (JSGF). The purpose of language models and grammars is to constrain the recognizer to certain word combinations. The size of search graphs is greatly affected by language models and grammars.

Language models are best suited for situations where flexibility and large vocabularies are required. Search graphs produced from language models must allow for any combination of words regardless of probability. This makes them typically much larger than search graphs produced from grammars. A typical application of language models is for dictation. The language model should contain as many possible combinations of sentences that could be spoken. In an n-gram language model, the probability of a word w appearing at location i can be defined as

$$P(w_i|w_{i-1}, w_{i-2}, \ldots, w_{i-n+1}), \forall n \geq 0$$

This simply means that the probability of a word appearing depends on the previous n-1 words. Language models constructed from sentences in a narrow domain

will outperform language models constructed from a broad domain in their own domain. The trade-off is that a language model tailored to a narrow domain will likely produce very low accuracy for utterances outside the domain. When constructing language models, it is likely that some n-grams will be much more represented than others while others may appear very few times or not at all. Thus, n-gram language models often discount and back-off. Discounting redistributes probability mass from more represented n-grams to less represented n-grams. The discount can be either absolute or proportional. The default language modeler used for Sphinx, *lmtool*, uses an absolute discount of 0.5. Backing-off allows the language model to use smaller n-grams that may be better represented such as using bigrams when trigrams are underrepresented.

Grammars are best suited for situations where syntax is predictable and vocabularies are small. They will produce smaller search graphs than language models given the same vocabulary, unless so complex as to mimic language models. This greatly reduces unnecessary computations while improving accuracy. A typical application of grammars is for commands. Grammars can be very complex giving the speaker the illusion of flexibility. Adding or omitting a single word that is not explicitly handled by the grammar will cause failed or incorrect recognition. The strict syntax does have at least two advantages over language models. While sentences generated using language models have no meaning to the computer without complex analysis, meaning can be easily attached to grammars through Tags. With grammars, multiple similar utterances can be tagged as having the same meaning. "Shutdown", "Turn off the computer", and "Close computer" can all be mapped to the command which turns off the computer. Another advantage is increased accuracy. Using language models, the utterance "one two three" can easily be recognized as "one to three". With the strict syntax of grammars, such errors can be avoided.

3.3.3 Decoder

The Sphinx4 Decoder takes as input the features from the FrontEnd and the search graph from the LanguageModel and outputs a result. It begins by creating a trellis which is a directed acyclic graph of the cross-multiplication of the incoming features and the search graph. The trellis is searched using the Viterbi [30] algorithm which finds the shortest path through the trellis. The SearchManager uses a token passing algorithm [74]. Starting with an initial state and one token, every connecting state is given a copy of the token. This process repeats until desired or a terminating state is reached. Each token represents a state in

the trellis and contains associated feature data, acoustic score, language score, frame number, etc. Each token also contains a reference to a predecessor; thus allowing the complete path the token has taken to that point to be reconstructed. The active set of tokens in the trellis is often maintained in one or more ActiveLists depending on implementation. If ActiveLists are used, a Pruner is used to reduce the number of potential paths. The task of applying scores to tokens is handled by the Scorer.

ActiveList Maintains a list of active tokens.

Pruner Because the number of potential paths may be large, the tokens in the ActiveList are pruned according to absolute and relative beam widths. The absolute beam width sets the maximum number of total paths. The lowest scoring paths below the absolute beam width are discarded. The relative beam width sets a threshold for the maximum difference between the best scoring path and the current path. Paths outside the threshold are discarded. Beam widths must be selected carefully as they can greatly affect accuracy.

Scorer Calculates the scores for every token in the trellis.

Chapter 4

A Speech Recognition Enabled ACR System for Paramedics

While some trivial applications can be developed in an ad-hoc manner, most applications require significant planning. Design decisions must be made early to avoid costly changes later in development. The design decisions can range from simple functionality additions to decisions that affect the entire design of the system and are often presented as requirements.

Firm requirements are essential to the development of any software. They provide clear goals that need to be met upon completion of the system. Without requirements, feature creep often results where resources are spent on unnecessary features not related to any goals. While these features may be useful, they should not be considered until the requirements are met. We have therefore determined a list of functional requirements and non-functional requirements. Functional requirements specify the architecture of our system. They define how our system should function. Non-functional requirements specify constraints on the architecture of our system. They define what constraints should be applied to the functions. Using these requirements, we have designed our speech recognition enabled ACR system.

We begin by describing both the functional and non-functional requirements of our system. These requirements are accompanied by rationale for their inclusion. This is followed by a brief overview of the system we developed and its three components: the Client, the Server, and the Viewer. We then describe the three components in detail and include explanations for design decisions. Because two of the components, the Client and Server, communicate continuously, we also describe the Client-Server communication in detail as well. Finally, we conclude

with the difficulties and limitations encountered during development.

4.1 Functional Requirements & Rationale

All sections of an Ambulance Call Report must be handled.
Ambulance Call Reports are divided into nine major sections: Administration, Clinical Information, Physical Exam, Clinical Treatment / Procedures & Results, Intake, Output, General Administration, Aid to Capacity Evaluation, Refusal of Service. Although some digital solutions such as Rampart CRT have divided these sections differently, it is not apparent that they provide any benefit over the original nine. To better distinguish the benefits of speech recognition in an ACR, we have decided to use the same nine sections in our system.

Unwanted utterances must be correctable.
Mistakes occur when recording speech. A user may continue to record when he/she does not wish to, such as when interrupted by a bystander. A feature therefore, must be in place to correct such unwanted utterances.

Mis-recognized utterances must be correctable.
Even in a perfect environment, speech recognition is not 100% reliable. It is thus expected that some words will be mis-recognized. Words can be mis-recognized because of environmental noise, incorrect pronunciation, pauses, and a variety of other factors.

All data can be input using speech recognition.
Speech recognition is a much faster method of recording language than typing. People can speak and listen comfortably at over 150 wpm, a figure which surpasses hand writing and even typing [67]. To achieve as much benefit as possible from speech recognition, as much functionality as possible must be provided by this means. Some functionality is excluded because it would vastly increase the dictionary size and increase error rate.

All data can be input using on-screen keyboard or dialogs.
If for whatever reason, the user wishes not to use speech recognition, an alternate form of input via an on-screen keyboard must be available. Reasons include environmental noise, microphone malfunction, or even withholding information from patients or bystanders.

There must be a way for completed reports to be transferred.
When a physician, or nurse, receives a patient, the completed report needs to be transferred.

There must be a way for completed reports to be signed.
For legal reasons, completed reports need to be signed when being handed over to the receiving care givers.

Server must handle at least 2 concurrent client connections.
Paramedics typically work in teams of 2. Therefore, to eliminate the need for a separate server for every client, each server must handle at least 2 clients.

Time logging must be automated.
Manual time logging is often error-prone and susceptible to manipulation. An automated system will not only save time, but will also provide more reliable data for data analysis.

4.2 Non Functional Requirements & Rationale

User interface must be intuitive and simple to use.
Any system that a paramedic uses must be as unobtrusive as possible to ensure care is not affected.

Language model must be specific to paramedics.
Because we are dealing with a specific domain (paramedics), a language modeled specifically for this group will increase speech recognition accuracy. Therefore, we must include colloquialisms, slang, short form, and abbreviations. It is expected that paramedics use vocabulary which is not necessarily represented in official forms and manuals, and these words must also be modeled.

The confidence level of recorded speech must be conveyed.
Because accuracy is critical, having a method to immediately see the confidence level of speech allows users to make changes on the spot and not at a later time from memory.

Report must be cached on lightweight client until it is signed off.
Because of the wireless nature of the design, if a connection cannot be made,

the client must be able to accept input and store it for future use. An added benefit is redundancy.

Server must have sufficient performance for speech recognition.
Speech recognition is a computationally expensive operation. Maximizing correctness while retaining acceptable speed requires a high-end computer.

Connection between lightweight client and server must be secure.
Because the paramedics are dealing with sensitive information, the connection between client and server must use some form of wireless encryption or encryption which allows secure transfer over insecure networks.

Connection status between client and server must be conveyed.
A connection must exist between the client and server for the system to have full functionality. Such a feedback will provide reassurance to the user that the system is fully functional.

4.3 Design Overview

Our system consists of a Client, Server, and Viewer. Each of these components run on separate hardware. The Client uses a small lightweight low-power device that is mounted on the paramedic's arm. The Server uses a powerful computer and is located in the ambulance. The Viewer also uses a computer, but is located in the hospital. The system was designed to allow flexibility with hardware selection. For instance, the Client will run on any device that uses the Android operating system, while both the Server and Viewer will run on any computer that has a Java Virtual Machine.

The Client and the Server are used by paramedics to fill out the ACRs. The Client allows the paramedic to fill out the ACR manually using an on-screen keyboard and dialogs, and through speech recognition. Because of the low power nature of the Client device, the Client only records speech as a WAV and transfers it wirelessly to the Server which performs the speech recognition. It then downloads the result from the Server as a XML. The process of uploading a recording and downloading the result is often completed within one second for fields using a grammar and within 5 seconds for fields using a language model. The speech recognition on the Server is handled by CMU Sphinx4, an advanced open source speech recognizer. Both grammars and a language model are used by Sphinx4 depending on the requirements of the field in the ACR. Grammars are more restrictive than language models, but provide higher accuracy, faster decoding and

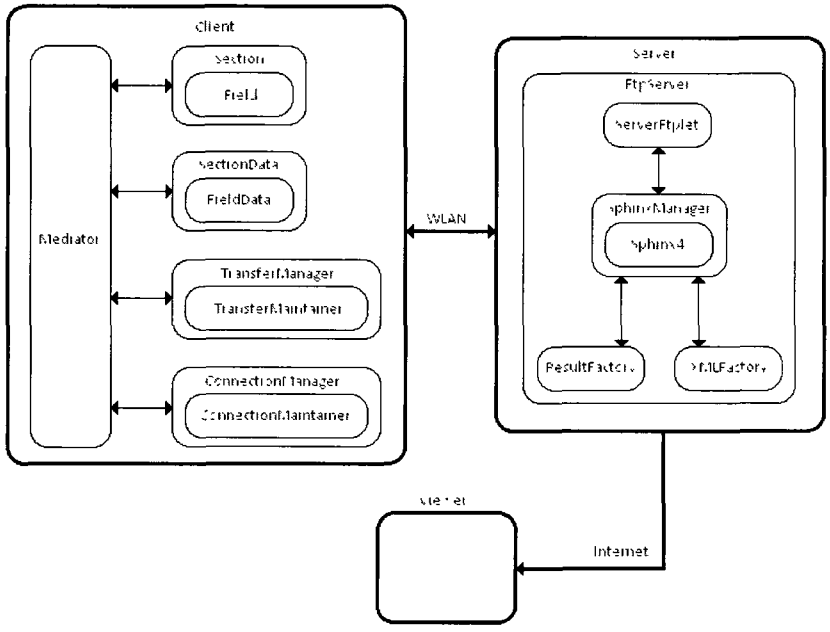


Figure 4.1: An overview of our system.

allow easier processing of the result into application useful data. Language models are the exact opposite and sacrifice these benefits for much greater flexibility in accepted utterances. The wireless connection is secured using WPA2. The Viewer is used by physicians and nurses to view the ACRs. It allows the user to easily view the data contained in an ACR. Fields that are labeled urgent are highly visible in the Viewer. Further, each field that was inputted using speech recognition has a button to play back the original recording.

An overview of our system is shown in Figure 4.1. For the Client, the Mediator, Section, Field, SectionData, FieldData, TransferManager, TransferMaintainer, ConnectionManager, and ConnectionMaintainer components were implemented by us. For the Server, SphinxManager, ResultFactory, and XMLFactory components were implemented by us. The FtpServer and ServerFtplet components are part of Apache FtpServer used by our system. The Sphinx4 component is part of the CMU Sphinx4 speech recognizer used by our system. The Client and Server are discussed in detail with explanations for decisions in sections "Client" and "Server" respectively, while the section "Client - Server Communication" discusses how they communicate. The Viewer is discussed in detail with explanations

for decisions in the section "Viewer".

4.4 Client

While the Server does most of the heavy processing, it is the Client that the user will interact with. The Client is to be worn by the user with a either a wireless Bluetooth or wired microphone (see Figure 4.2). The user interface is designed for one-handed use. It is expected that not having to hold the Client will allow the paramedics to simultaneously provide patient care while filling out the ACR.

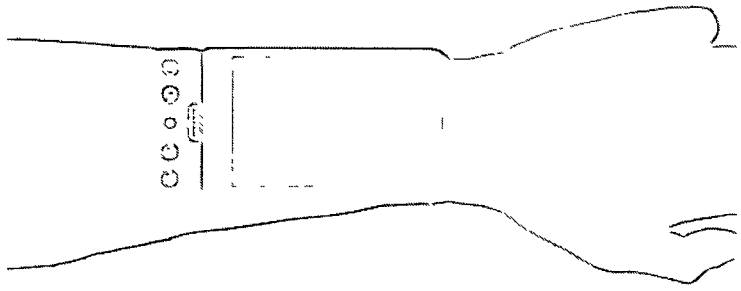


Figure 4.2: The location and relative size of the arm mounted Client.

A common source of frustration with many applications is their usability does not match the quality of their feature set. A feature may be available, but be buried deep in a menu, or only accessible using the command line. In other cases, the feature may be easily accessible, but not perform the expected actions. Many of the most popular user interfaces in the world have usability problems. Facebook suffers from general clutter from an attempt to provide as many features as possible. Google Chrome suffers the exact opposite. In an effort to reduce clutter, expected features in the browser were removed or poorly conveyed. When features are added to new releases, they are all placed under one ever-expanding control button.

Designing a powerful and easy interface is an incredibly challenging task. It can involve taking into account the set of tasks it will be used for, the hardware it runs on, the environment it runs in, the users who will be using it, and more. There are numerous generally accepted UI standards we can follow that will improve the usability of our application. Further improvements will come by using accepted testing methods such as Thinking-Aloud [25] and Heuristic Analysis [51].

4.4.1 Hardware

The Client of our system was originally developed for an HTC G1 smart phone. The G1 was the first device to use the Google Android platform. It had a 3.2-inch display with a 320 x 480 resolution, poor battery life, and only supported 802.11b/g Wi-Fi. Since development began, several Android based smart phones and tablets have come to market that offer significant improvements over the G1. As a result, we have switched development to the Samsung Galaxy S. The Galaxy S has 4-inch display with a 800 x 480, larger battery, and support for 802.11b/g/n Wi-Fi. While we have used these devices, our Client will function adequately on any Android based device.

4.4.2 Software

Developing for devices based on the Android platform has several advantages over competing platforms such as Apple's iOS and Microsoft Windows Mobile. Because of Android's rapidly expanding market share which currently sits at 34% of the US market [18], there is a very large selection of devices to choose from. Specifications vary with processor speeds from 500 MHz to 1,000 MHz, memory from 128 MB to 512 MB, and screen sizes from 2.6 inches to 10 inches. Price points range from $250 to $800. Some devices may be better suited than others for a given environment or user. Developing on the Android platform also allows us to use one common programming language for the entire system as Android runs a Java Virtual Machine.

4.4.3 User Interface

While designing the Client's user interface, several factors were taken into account:

The User The Client is to be used by paramedics. While paramedics are often comfortable around computers, we cannot assume this to be true for all paramedics. Many paramedics have spent most of their careers filling out

conventional paper ACRs making them comfortable with the section design and code description pairings.

The Environment The Client will be worn by the paramedic. The ACR may be filled out in a variety of environments such as at the scene, en-route to the scene, en-route to the hospital, or after arrival at the hospital. These environments are likely to be bumpy and noisy. With current systems, ACRs are most often filled out en-route to the hospital and after arrival at the hospital.

The Task The paramedic's primary job is to provide primary care and transport of the patient to definitive care. Filling out the ACR, while important, is secondary.

The Hardware The device must be small and light enough to be comfortably worn by a paramedic. As a result, the size of the screen is limited to approximately four inches. Current resolutions of 800 x 480 for 4-inch screens approach the resolving power of the human eye at 12 inches. Therefore we cannot expect increases in resolution to solve our problems.

These factors give us a framework to work in. Together with generally accepted UI standards, we designed our UI. We will describe our high-level UI decisions first, and progressively describe lower-level decisions.

While most modern Paramedic Services have switched to digital ACRs, their designs introduced some significant changes to how and where data is gathered. Because we could not find any data that either supports or opposes the changes, we decided to take advantage of user experience with paper ACRs and design our system around them. We therefore have the following 9 sections:

1. Administration
2. Clinical Information
3. Physical Exam
4. Clinical Treatment & Results
5. Intake
6. Output
7. General Administration

8. Aid to Capacity Evaluation

9. Refusal of Service

Selecting sections 1, 2, 3, 7, 8, 9 will bring up a list of every field in that section. Figure 4.3 shows the Administration section. Sections 5, 6, 7 were implemented as tables in the paper ACRs. Selecting sections 5, 6, 7 will instead bring up a section overview of the table. The section overview allows the user to create new rows in the table and access previous rows. Figure 4.4 shows the Clinical Treatment & Results section overview. Creating or accessing a row will bring up a list of every field in that section.

Figure 4.3: Administration section.

Each section contains a non-scrolling menu on the left hand side of the screen. It contains a "Record Speech" button, a "Switch Input" button, and a "Go Back" button. To the right of this menu is a scrollable list of all the fields contained in the section. The sizing of each field allows three fields containing one line of data each to visible at one time. Smaller text would increase the number of visible fields at a cost of reducing readability. To further improve readability, the color scheme used throughout is white on a black background.

The Client was designed to accept both manual entry using an on-screen keyboard and dialogs (see Figures 4.5 and 4.6) and speech recognition as sources of data. This flexibility is required for various reasons. Some fields such as time and date may be filled out quicker simply using manual entry. Speech recognition is not perfect and errors may be easier to correct manually. Finally, if the wireless connection with the Server is lost, manual entry will still allow data to be inputted.

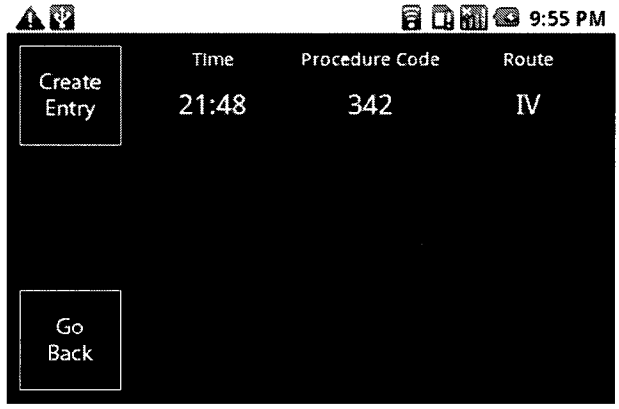


Figure 4.4: Clinical Treatment & Results section overview.

While speech recording will still be allowed, no results will be returned until the connection is automatically reestablished.

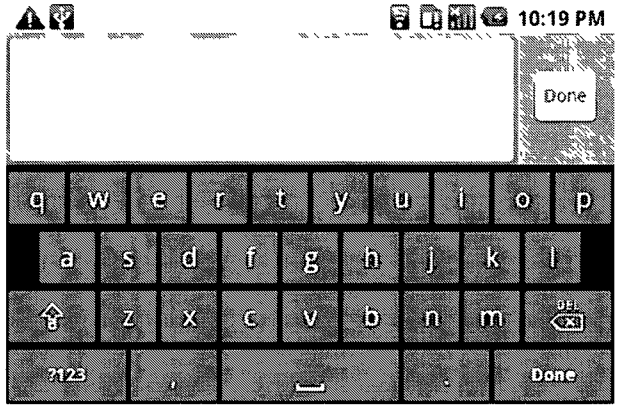


Figure 4.5: Onscreen keyboard used for manual entry.

In an effort to maximize limited screen real estate, we aggressively reduced clutter while trying to maintain functionality. An early design had both a "Manual Entry" and "Record Speech" side by side on each field. This led to selection errors because of their close spacing. It also reduced the already limited space for field data. Therefore, these two buttons were merged into one that could be switched between both actions by pressing the "Switch Input" button in the menu.

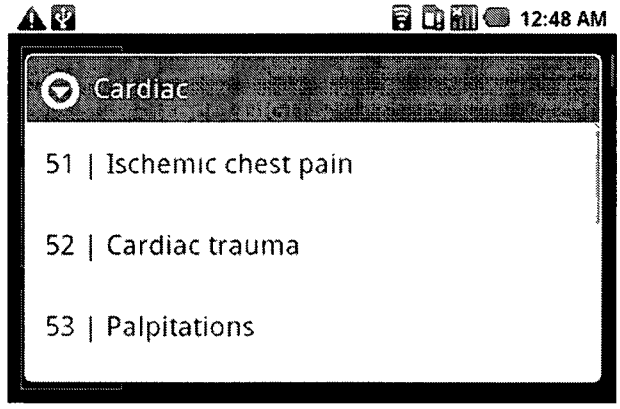


Figure 4.6: Dialog used for manual entry.

Speech can be recorded by either pressing the large "Speech Record" button on the menu or pressing the "Speech Record" button on the specific field. Naturally pressing the button on the field will record for that field. Pressing the button on the menu will record for the currently highlighted in red field. The highlighted field always moves to the following field once data is inputted. It is accompanied by an automatic scrolling of the list of fields. When recording begins, a large "Recording... Press to Stop" frame appears at the bottom of the screen (see Figure 4.7). An early design made the stop recording button fill the entire screen to maximize ease of selection. A consequence of such a design was that it hid the list of fields potentially allowing the user to forget what field they were recording for. If a user wishes to record successive fields, they may simply press the "Record Speech" button on the menu to automatically stop the previous recording, and begin recording for the following field.

Each field displays a confidence level of the recording on the right edge of the field. The confidence level is calculated by Sphinx4 based on how closely a recording matches the acoustic model. A high-level of confidence (i.e., >75%) is denoted as a green bar, a medium level (i.e., >25% and ≤75%) as a yellow bar, and a low-level (i.e., ≤25%) as a red bar. While a recording may be recognized correctly with a medium-level confidence, the feedback indicates the recognizer had difficulties doing so. This may be caused for reasons such as user error or a noisy environment. The user may use this information to increase confidence levels by speaking more clearly or recording in a quieter environment. Because confidence calculations do not work with the Linguist used for grammars in Sphinx4, fields that use grammars will either have a green confidence if they return a result or

Figure 4.7: The"Recording...Press to Stop" button which appears on the lower half of the screen while recording.

a red confidence if they do not. Confidence is calculated correctly for fields that use language models.

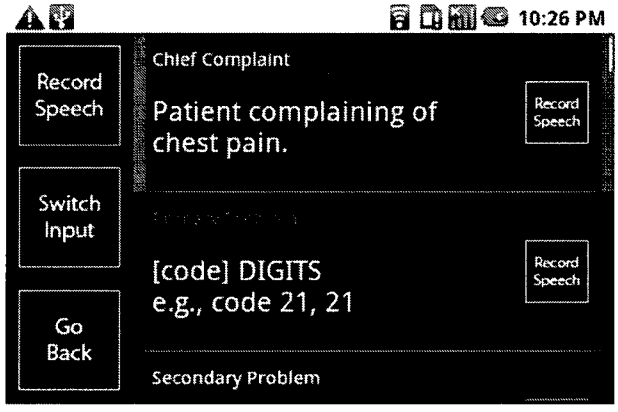


Figure 4.8: A field that is marked urgent and recognized with a high-level of confidence.

Each field can also optionally display a red bar on the left edge of the field. It denotes an urgent field. Both manually-entered and speech-recognized data can be marked "urgent" by long pressing the corresponding button. Figure 4.8 shows a field marked urgent that is recognized with a high level of confidence. Urgent fields represent information that the user deems noteworthy such as rare allergies. Urgent fields are highly visible in the viewer to the physician or nurse.

If a field contains no data, it will display a hint. For grammars, the hint displays the correct syntax for speech recognition and is sometimes accompanied by an example. For language models, the hint displays "FREE_SPEECH". The use of hints became evident with early prototyping, as users were consistently unsure of what they were allowed say for each field.

Finally, recording for fields that use a language model contains extra functionality to account for its flexibility and lower accuracy. Each time a user records for such a field, the resulting data is appending to the previous data. By recording "undo" the most recent data is undone by reloading the previous XML. By recording "clear" all data is removed.

A user interacts with an application through its user interface (UI) and speech recognition is only a small part of it. A poor UI will lead to user dissatisfaction regardless of how good the actual speech recognition component of our system is. We attempted to strike a balance between ease of use and functionality. Constant prototyping allowed us to remove many annoyances while introducing interesting new functionality.

4.4.4 Start, Stop, Save Scenario

Figure 4.9 shows a sequence diagram depicting a scenario where a recording is started, stopped, and then saved. The same scenario applies for all sections of the ACR.

Section Each Section contains an Adapter that populates the fields of a Section and handles all interaction with the list of fields. The Adapter is not displayed in the scenario as it is tightly coupled to the Section and is withheld for brevity. When a user initiates recording for a particular field, the Section first requests the PCMAudioRecorder from the Mediator. Because previously initiated recordings may be ongoing, a stop recording request is sent to the PCMAudioRecorder. The request is asynchronous and the recording will usually not stop before the start recording request is sent. The start recording request initiates a separate thread and simply waits for its turn without blocking the UI thread. In this scenario, it is assumed that no previously initiated recording is ongoing.

Mediator The Mediator is the central component of our application on the Client. It stores the report data of every section which in turn store the report data of every field. Any access to the report data must go through

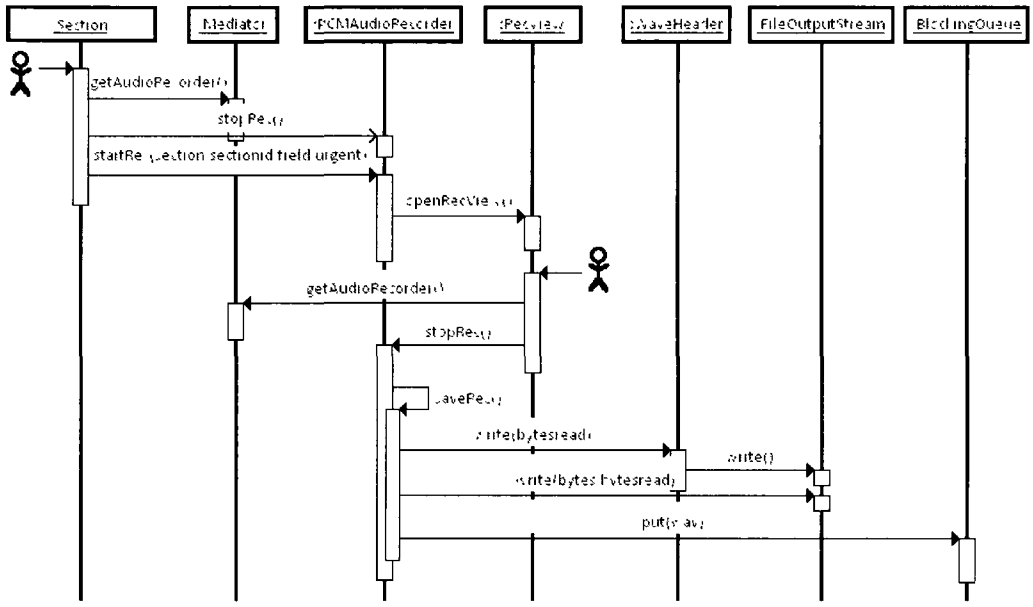


Figure 4.9: A sequence diagram depicting a scenario where a recording is started, stopped, and then saved.

the Mediator. The Mediator also stores the instance of ConnectionManager which maintains a FTP connection through the ConnectionMaintainer, TransferManager which maintains all transfers through the TransferMaintainer, and the BlockingQueue which contains all pending uploads and downloads. As a result of the Android API, each Section is loaded indirectly which makes direct passing of the Mediator impossible. To work around this, the Mediator is built using the Singleton pattern which creates at most one object of a class. Therefore, the Mediator can be accessed from anywhere in the application using a simple "getInstance()" method.

PCMAudioRecorder PCMAudioRecorder will only begin processing a new start recording request if no other recording is taking place. Once this is the case, the current start recording request acquires a lock to block any other requests that may occur. With a lock acquired, PCMAudioRecorder sends a request to RecView and immediately begins reading the recording buffer. The PCMAudioRecorder finishes reading the buffer and immediately saves the recording to device memory. The number of bytes read is

sent to WaveHeader which creates a correct WAV header which is then written to the FileOutputStream. The bytes read and the number of bytes read are then also written to the FileOutputStream. Finally, the complete path of the saved recording is put in the BlockingQueue to be eventually used by the TransferMaintainer. The multi-step process of saving is required because our application creates mono, 16 bit, 16000 Hz PCM WAVs. Media recording using the Android API is typically done using a MediaRecorder, which only saves recordings in the AMR (Narrowband) format. To save in our desired format, we needed to handle the entire recording process at a lower level.

RecView RecView displays a "Recording...Press to Stop" recording button on the bottom half of the screen. When a user presses the button, RecView retrieves the PCMAudioRecorder and requests it to stop recording. This closes the RecView window.

WaveHeader WaveHeader creates a correct WAV header based on bytes read, number of channels, frequency, and bit rate.

FileOutputStream Standard Java FileOutputStream.

BlockingQueue The BlockingQueue maintains all pending transfers, both upload and download. It is capable of blocking the thread if the queue is either empty or full.

TransferMaintainer The TransferMaintainer, which is not shown in this sequence diagram, is responsible for uploading the WAV and downloading the resulting XML. Once downloaded, the XML is parsed using SpeechXMLHandler and the correct field is filled with data. The multithreaded design of our Client application allows multiple recordings to be sent to the Server before the first XML is ready for download. The application handles such a situation by simply processing the results in the background as they are downloaded.

4.5 Server

The processing power of state-of-the-art low power devices such as smart phones and tablets is significantly lower than that of even entry-level desktop computers. These devices have size and battery life constraints that desktop computers do

not. While their per core processing power is increasing rapidly because of market demand [35], there are still many applications, such as speech recognition, that require the processing power and memory of a desktop computer. The per core processing power of desktop computers has stagnated in recent years. Instead, to increase performance, modern CPUs have two or more processing cores. While two cores can theoretically double the processing power of a single core by running together in parallel, many applications do not or cannot take advantage of such a coarse level of multiprocessing.

The processing and memory demands of speech recognition can vary dramatically based on several factors. For instance, specifying a grammar will almost always result in lower demands than if a language is specified. The Sphinx4 Linguist creates an internal grammar with transition probabilities regardless if a grammar or language is specified. The internal grammar created from a typical specified grammar will be a one-to-one conversion and likely contain a limited number of potential paths. The internal grammar created from a specified language using an identical vocabulary will contain every possible word combination. The resulting number of paths can easily reach hundreds of megabytes depending on vocabulary size. Sphinx4 uses a pruner to keep the number of paths at a manageable size, but is still not suitable for low-power devices.

4.5.1 Hardware

Because of speech recognition's high processing and memory requirements, to achieve high accuracy while maintaining quick decoding, a modern mainstream machine is recommended. Currently this implies an Intel i5 processor or equivalent with 2 GB of RAM. These processors contain 4 separate cores allowing multithreaded applications to potentially receive significant speed improvements. Scoring in Sphinx4 is parallelizable and is shown to benefit from multi-core CPUs [71]. Our system would place such a machine inside the ambulance. Not only would the paramedics not have to carry a relatively bulky machine, but the machine would use the ambulance's power alleviating any battery concerns.

4.5.2 Software

Our application on the Server can be broadly defined as having two core components that function together: a FTP server using Apache FtpServer [9] and a speech recognizer using CMU Sphinx4 [7]. A tight coupling is achieved by integrating both components into our code. Both FtpServer and Sphinx4 are written

in Java; thus it was only natural that our application was written in Java. This greatly simplifies integration of these components and provides the numerous benefits of Java such as platform independence and garbage collection.

Sphinx4 is the latest in a line of speech recognizers from CMU. It is primarily intended to be used as a research tool. As such, while its API is well documented and the official homepage provides a few useful tutorials, there is no real customer support aside from the official message board. Sphinx4 is designed to be extremely flexible and modular. Entire modules can be replaced without the need to recompile and often while the application is running. A multitude of parameters can be tuned that dramatically affect speech recognition accuracy and speed. This flexibility allows us to fine tune our application while also providing insight into the inner workings of speech recognition. In our opinion, these benefits clearly outweigh the steep learning curve.

Sphinx4 is released under a BSD license, a permissive free software license. Permissive licenses allow the authors to use the software, both binary and source, however they wish. A common criticism of GPL , the most common license in the open source community, is that it is a copy-left license. This implies that if any part of the software is under the GPL, then all of the software must also fall under the GPL. This restricts where GPL software can be used and has been used to debate the definition of "free" in free software. Using Sphinx4 as a speech recognizer avoids these restrictions. Microsoft's Speech Platform SDK is also released under a flexible license, although it is less flexible than a BSD license. It permits royalty free redistribution in applications, but does not provide source code.

Apache FtpServer is a fully functioning FTP server written in Java. It is released under the Apache license, which like BSD, is also not copyleft. It was chosen specifically because it can be embedded into our Server application. Not only does this allow immediate feedback to what the FTP server is doing, it allows custom commands to be easily added. These features allow the two major components of our Server application (i.e., the speech recognizer and FTP server) to communicate efficiently.

4.5.3 Sphinx4 Performance

Table 4.1 shows Word Error Rates (WER) and Real Time ratios (RT) for four different vocabulary sizes using Sphinx4.

TIDIGITS, which only allows connected digits (i.e. oh, zero, one... nine), has a WER of 1 word for every 200 words spoken. Such high accuracy approaches

	WER	RT 1 CPU	RT 2 CPU
TIDIGITS (11 words)	0.549	0.07	0.05
RM1 (1000 words)	2.739	0.50	0.40
WSJ5K (5000 words)	7.174	1.22	0.96
HUB-4 (64000 words)	18.878	4.40	3.80

Table 4.1: Word Error Rates (WER) and Real Time ratios (RT) for four different vocabulary sizes using Sphinx4 running on Dual Core 1 GHz UltraSPARC with 2 GB of RAM [71].

human error rates [45]. Sphinx4 is not alone in achieving very low WERs for connected digits recognition. Microsoft's Speech Recognizer built into Windows uses connected digits as a fallback option to prevent "spiraling error situations, where the attempt to fix one error led to creating additional errors" [52]. When a user wishes to correct an error, he/she is presented with a dialog with numbered solutions; thus reducing the chances of repeat error. Nuance's Dragon NaturallySpeaking, another popular speech recognizer, provides similar functionality. While such solutions reduce the likelihood of repeat errors, they remind the user harshly that they are in fact speaking to a machine and not a human. Rarely does a server at a fast food restaurant ask a patron to state the combo number if they did not understand an order the first time. The best input methods are those that are invisible to the user. Users don't describe "moving the mouse to the icon and clicking it". Rather they "select the icon". Similarly, they don't describe pressing the individual keys while typing a document. The input method is an extension of their thoughts and actions are typically subconscious. While dictating a document may feel natural, for many users speaking commands is not. Therefore, care must be taken when deciding how and when an application should use speech recognition with some applications being a more natural fit than others.

The results for the RM1 and WSJ5K tasks are also very relevant to our system. While most of the ACR is already codified, there are still vital fields that require larger vocabularies. These fields may be constrained by a grammar or a language. A grammar is used as often as possible as it defines specific utterances that may be allowed. Utterances that are too similar to each other or easily mis-recognized can be reworded to increase accuracy. Such a heavy hand to constraints is not always possible. Some fields that require natural language such as comments or history require flexibility that grammars are not designed to offer. In such

cases, languages must be used. By default, Sphinx4 uses trigram language models which are a type of n-gram language model where n=3. Language models allow any combination of words to be spoken and use probabilities to determine likely utterances. Table 4.1 shows Sphinx4 can achieve WERs of approximately 3 words for every 100 words for a typical vocabulary of 1000 words, and approximately 7 words for ever 100 words for a typical vocabulary of 5000 words. Because of the specific vocabulary used in filling out ACRs, a comprehensive language model will contain well under 5000 words. We therefore expect our WERs to fall somewhere between these two figures. The acceptance of such an error rate is discussed in the study described in Chapter 5.

4.5.4 Grammar Design

Each field that does not require the flexibility of a language, but requires speech recognition, has an associated grammar. Grammars are written by hand to be as precise as possible. Precision does not imply a complete lack of flexibility. Many accepted utterances contain multiple optional words to account for variations between how paramedics speak. For example, while grammars for time of day only accept utterances based on 24 hour time, both "seven hundred hours" and "seven hundred" are accepted. Similarly, both "zero hundred hours" and "twenty four hundred hours" are accepted and taken to mean 0:00 even though 24:00 is technically improper. These variations can significantly improve the usability of speech recognition systems such as ours at only a small cost in accuracy.

Our system is currently set up to force a match between a spoken utterance and potential utterances accepted by the grammar. Detecting out of grammar utterances is unnecessary because our system forces the paramedic to press a button to initiate recording. It is expected that the paramedic will only attempt to speak accepted utterances. While this may sometimes generate incorrect results, it is preferred to potentially rejecting mispronounced or noisy, but otherwise accepted utterances.

Grammars used in our system follow the JSGF format. While they allow us to explicitly control the potential results returned by the decoder, the results are still only Strings. Therefore, the utterance "two hundred and fifty" will generate the String "two hundred and fifty". These Strings often have little to no value to the application in their current form. Transforming these Strings into data that the application can use can be challenging. If the decoder instead generates the String "250", a simple typecast will allow this data to be useful to the application. JSGF grammars have a tagging mechanism to facilitate this transformation. Thus, a

simple grammar that would accept the utterance "two hundred and fifty" can be written as:

```
public <number> = ( two hundred and fifty ) {250};
```

Tags allow the encapsulation of legal utterances and their associated useful application data. This decoupling results in numerous benefits for the application as a whole. Updating grammars no longer necessitates updates to the application code. Complete grammars can be replaced without the application being affected. This greatly reduces the chances of bugs and the amount of coding required.

Unfortunately, more complex grammars quickly exceed the design of tags. In some cases we would like the result generated by the decoder to be placed into an Array. In the utterance "Today's high is thirty degrees Celsius and low is ten degrees Celsius," the application may only be interested in the two temperature values. These values are stored in Arrays of length 2 where the first number is the high in degrees Celsius and the second is the low in degrees Celsius. Providing the application with data in a useful form requires more than simple tags. ECMAScript extends the functionality of tags [47]. It allows tags to contain pieces of code. These pieces are executed in the same order that the tags are reached. Using ECMAScript, a simple grammar that would accept the utterance "Today's high is thirty degrees Celsius and low is ten degrees Celsius" can be written as:

```
<number> = thirty { this.$value = \30" } | ten { this.$value = \10" };

public <temperatures> =
   <NULL> { this.temp = new Array() }
   todays high is
   <number> { this.temp = this.temp = this.temp.concat($.$value) }
   degrees celsius and low is
   <number> { this.temp = this.temp = this.temp.concat($.$value) }
   degrees celsius;
```

Using an ECMAScript parser, which is given the grammar and original String result, the native Array of length 2 called "temp" can easily be extracted from the grammar.

Not every grammar in our system is unique. Many fields are satisfied by identical grammars. The following grammar is used for any field that requires a single code:

```
<number> =
   zero {this.$value = "0"} | oh {this.$value = "0"} |
   one {this.$value = "1"} | two {this.$value = "2"} |
   three {this.$value = "3"} | four {this.$value = "4"} |
   five {this.$value = "5"} | six {this.$value = "6"} |
   seven {this.$value = "7"} | eight {this.$value = "8"} |
   nine {this.$value = "9"};

public <code_rule> =
   <NULL> {this.code = ""} [code]
   ( <number> {this.code = this.code + $number.$value} )+;
```

Any utterance that optionally begins with the word "code" followed by one or more digits is accepted. Naturally the word "code" is of no use to the application and only the spoken digits are returned in the "this.code" variable. The reasoning behind pairing each field with a separate grammar is that often slight variations to the grammar specific to the field may be done to further increase accuracy. A grammar that accepts a number from 0 to 999,999 may be required for an Altitude field, but such a grammar for a Blood Pressure field would generate numbers that are impossible for blood pressure readings. In such a case, the grammar for the Blood Pressure field should be limited to numbers from 0 to 999.

4.5.5 Language Model Design

Fields which require more flexibility than grammars for speech recognition are handled by a language model. While each field that uses a grammar has its own grammar, the language model is shared by all fields that use it. This is possible because the language used in the ACR is applicable to all the fields. The language is also specific enough to keep the Word Error Rate from the resulting language model in the low single digits.

A grammar forces a speaker to only say predefined utterances. While restrictive, a speaker can in time become familiar with the accepted utterances and the resulting high recognition accuracy will instill confidence in the system. A

language model gives the speaker the flexibility to say any combination of words in the vocabulary. If expected words are missing from the vocabulary, or their combinations are poorly modeled, accuracy can decline drastically. In time, the speaker will lose confidence in the system. Language models generated from large domain specific text corpuses have a much greater chance of successfully modeling utterances. To create a sufficiently complete language model for our system, hundreds of ACRs may potentially need to be used depending on the variability between them. Our system uses a combined corpus from three ACRs, numbers from 0 to 300, and some system specific commands to generate the language model.

No attempt is currently made to transform the resulting Strings returned by the decoder to application useful data. Therefore, the utterance "blood pressure is one eighty over ninety" will result in the String "blood pressure is one eighty over ninety". Such a solution is unacceptable in a deployable system and would need to be handled by either post processing the resulting String or modifying it at a lower level (i.e., at the Linguist, or at the Decoder). The modularity of Sphinx4 makes it possible to attempt such low-level modification.

4.5.6 WAV to XML Scenario

Figure 4.10 shows a sequence diagram depicting a scenario where a WAV is uploaded to the Server, decoded, and a XML is generated.

ServerFtplet The ServerFtplet on the Server hooks into FtpServer. It listens for any commands sent to the FTP server and allows code to be executed both prior to a command being executed and after the command has been executed. The ServerFtplet decides what should be done with an uploaded file. For instance, WAVs are put in a BlockingQueue which are then retrieved by the SphinxManager which handles speech recognition in our system, while XMLs require no further action. The ServerFtplet allows the list of commands handled by the FTP server to be easily expanded by handling any unknown commands manually. To be able to check if a XML is ready for download, we have implemented a CHK command.

BlockingQueue The BlockingQueue is used to prevent any bottlenecks. When a WAV is received by the FTP server, its filename is placed in the queue. SphinxManager which is looping in a separate thread retrieves the filename from the queue and sets up the Recognizer. The WAV is decoded only when all the previous WAVs in the queue are handled. This can sometimes lead

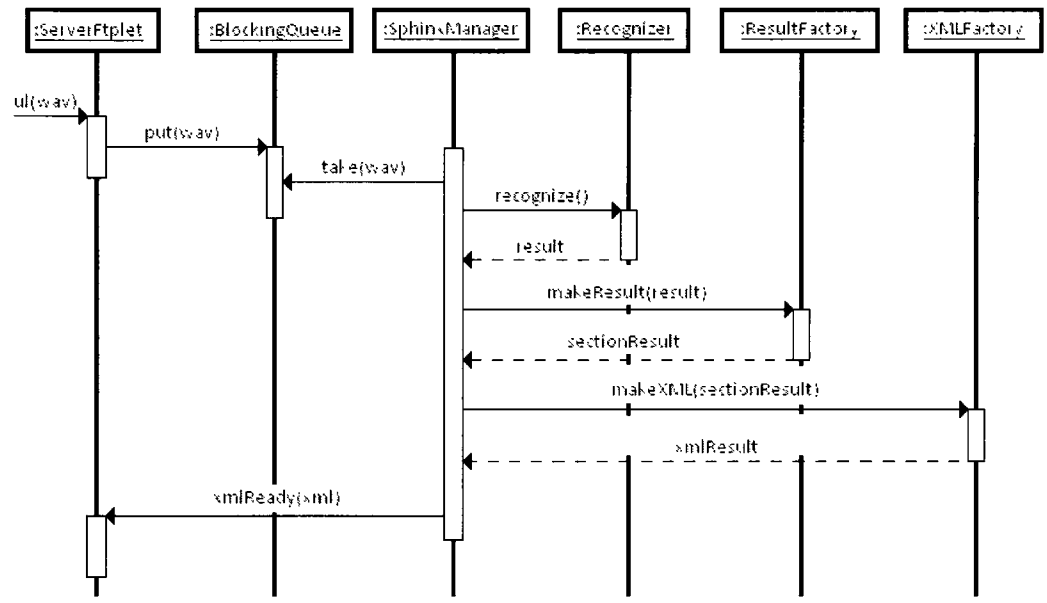


Figure 4.10: A sequence diagram depicting a scenario where a WAV is uploaded to the Server, decoded, and a XML is generated.

to a lengthy delay between when WAV was uploaded to the FTP server, and when the corresponding XML is ready for download. Without the BlockingQueue, some FTP commands such as uploading a WAV would block the FTP server until decoding is complete.

SphinxManager SphinxManager is a crucial component of our Server. It has two Recognizers loaded into memory: one which uses a language model, and another which uses grammars. It decides if the WAV to be decoded should be handled by a language model or a grammar by parsing the filename. If it is a grammar, it loads the appropriate grammar for the field. The WAV is then handled by the correct Recognizer. Once the Recognizer returns a result in the form of a String, SphinxManager sends it to ResultFactory which creates application useful data from the result. SphinxManager then sends this application data to XMLFactory which simply converts it to an XML. Finally, SphinxManager notifies ServerFtplet that the XML is ready for download.

Recognizer The Recognizer is part of Sphinx4. Given an audio input (i.e.,

WAV), it returns a Result object. Most commonly, the Result is used to return a final String representation of the decoded audio input.

ResultFactory The ResultFactory uses the String representation of the decoded audio input and the section and field to generate application useful data. It contains HashMaps for code description pairs in the ACR. Using the section and field with the code description pairs, it can decide which pairs are valid for a given field and only return a description if the code exists in the HashMap. For example, if the audio input resulted in the String "code one two one", the ResultFactory would check the HashMap for "121" and return the description if it exists.

XMLFactory The XMLFactory generates an XML based on the data generated by ResultFactory. The XML is stored locally and then the ServerFtplet is notified that the XML is ready.

4.6 Client - Server Communication

The hardware requirements of speech recognition had a large effect on the design of our solution. To ensure the highest accuracy while maintaining reasonable decoding speed, it was decided that the task of speech recognition should be handled by a high performance computer (i.e., the Server). The mobile computer (i.e., the Client) therefore would be the device that the user would interact with. With a reliable low-latency link, the user should not perceive that speech recognition processing is actually being offloaded to another device.

Two types of data are being exchanged between the Client and Server: audio data and textual data. The audio data is recorded at 16,000 Hz, 16 bits, in mono (single channel). It is saved as an uncompressed WAV using the LPCM encoding format. These values were chosen because of the design of Sphinx4 which is optimized to run at the specified settings. The textual data contains ACR data. It is stored using the common Extensible Markup Language (XML). Each XML can contain data from one or more fields of an ACR. The Server generates an XML to describe the result of audio data. The Client can also generate XML to describe manual changes to the ACR. Every WAV for an ACR is placed in a separate "audio" folder. Every XML for an ACR is placed in a separate "xml" folder. These folders in turn are placed in separate folder for each ACR.

4.6.1 Wireless Connection

The Client and the Server are connected wirelessly. While both ad-hoc networks and networks created using an access point would satisfy our goals, the version of Android that our Client was developed for had only limited ad-hoc functionality. Ad-hoc networks set up in Android 1.6 were limited to WEP encryption that has been shown to be easily compromised [68]. Security is critical for our system since we are handling confidential patient data. Security is simplified because of our decision to use WLAN over WWAN. Since data travels over public networks with WWAN, both the radio communication and the data itself need encryption. With WLAN, since we confine ourselves to a private network, only the radio communication needs encryption. This encryption is provided by WPA2, which is currently known to be secure. Not only is bandwidth much greater, but critically, its latency is much lower. The longest audio recording permitted by our system at 10 seconds has a file size of approximately 300 KB, but the majority of recordings are approximately 50 KB. In the average case, bandwidth is not an issue. A user is much more likely to perceive a delay as a result of latency. Single digit ms latencies are regularly achievable on a WLAN where as the current state of the art WWANs regularly achieve latencies of 50 ms to 100 ms. The range of WLANs is affected by line of site, but implementations using 802.11n can reach 500 m [17]. While this is adequate for a majority of cases that paramedics will encounter, for situations where the patient is on a multi floor building, a mobile server solution would be required. Such a solution could perhaps be attached to a stretcher. If a connection is lost, the user is notified on the device and can continue to fill out the ACR using the on-screen keyboard and dialogs.

4.6.2 Connection Protocol

The client-server design of our system necessitated a protocol for communication. The protocol chosen needed to be able to reliably transmit both binary data (i.e., WAV data) and textual data (i.e., XML data) while being simple to implement. The File Transfer Protocol (FTP) is a very common client-server protocol. Although mainly used for file transfer, it can be augmented to perform other tasks as well.

If a request is made and a response does not arrive within a specific amount of time, a timeout results, and the client may reattempt the request. With our system, the Client should upload a WAV file, have the Server decode it using the Sphinx4 component, and return an XML file. The FTP protocol is not designed to perform such a task in one step. Therefore, this task is broken down into three

sub tasks: an upload task (see Figure 4.11), a check task (see Figure 4.12), and a download task (see Figure 4.13).

A WAV is first uploaded to the Server. Based on the filename of the WAV, the Client adds the corresponding XML filename to the transfer queue. Once the WAV upload is complete, the Server immediately adds the corresponding XML filename to a HashMap of XML filename Boolean pairs called a readyList.

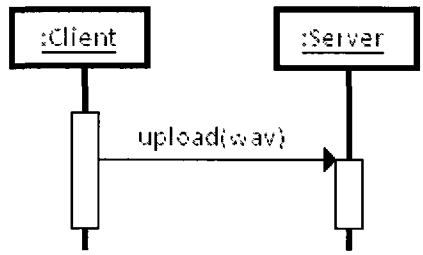


Figure 4.11: A sequence diagram depicting the upload task.

Before the XML can be downloaded, the Client must check the Server if the file is indeed ready for download. A custom command has been added to the FTP protocol called CHK. CHK checks the readyList for the XML filename and returns 200 if readyList returns true, 800 if the readyList returns false, and 900 if the readyList returns NULL. If CHK returns 800, the Client will send the CHK command again every 100 ms. The delay is used to avoid swamping the Server while minimizing the amount of time an XML is ready for download. If CHK returns 900, the XML filename is missing from the readyList and will be removed from the transfer queue on the Client. Such an error can occur if the Server crashes right when a file is finished uploading, but not yet added to the readyList. If CHK returns 200, the Client immediately downloads the XML.

4.6.3 Upload, Check, Download Scenario

A WAV upload, check, XML download scenario in our system is represented by the sequence diagram shown in Figure 4.14.

TransferMaintainer A TransferMaintainer on the Client runs on a separate thread while a connection with the Server is established. It maintains a blocking queue of files that need to be uploaded and downloaded. The

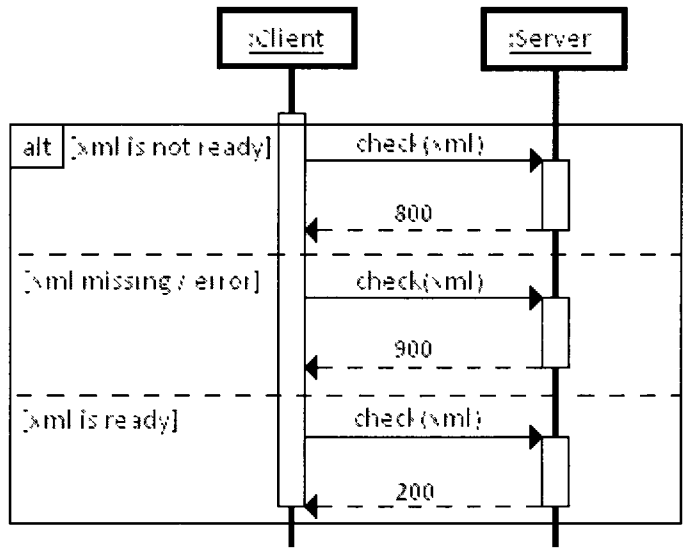


Figure 4.12: A sequence diagram depicting the check task.

blocking calls to retrieve from the queue are given a timeout to facilitate simple and safe termination of the thread.

ServerFtplet The sequence diagram in Figure 4.14 is slightly simplified since TransferMaintainer actually communicates with an FtpServer object and not the ServerFtplet. We show the ServerFtplet because it is able to intercept every command sent to the FtpServer. See the description for Figure 4.10 for more details.

4.6.4 Connection Scenario

A connection scenario in our system is represented by the sequence diagram shown in Figure 4.15.

ConnectionMaintainer A ConnectionMaintainer on the Client runs on a separate thread once a connection is established with the Server. Its purpose is to maintain a connection and to reestablish a connection if it fails. A connection is maintained with the Server by sending a NOOP command every 10 seconds. A NOOP command simply resets the FTP's timeout timer.

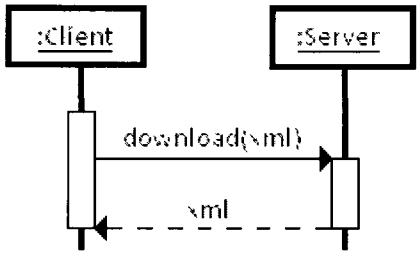


Figure 4.13: A sequence diagram depicting the download task.

When a connection between the Client and Server is lost, the ConnectionMaintainer will attempt to reconnect using the same login and password every 10 seconds.

ServerFtplet The sequence diagram in Figure 4.15 is slightly simplified since ConnectionMaintainer actually communicates with an FtpServer object and not the ServerFtplet. We show the ServerFtplet because it is able to intercept every command sent to the FtpServer. See the description for Figure 4.10 for more details.

4.6.5 Naming Conventions

The file naming convention for WAVs is as follows:

```
rec_(Section)_(SectionId)_(Field)_(Version).wav
rec_(Section)_(SectionId)_(Field)_(Version)_urgent.wav
```

The file naming convention for XMLs is as follows:

```
rec_(Section)_(SectionId)_(Field)_(Version).xml
```

Section Corresponds to a section in the ACR (e.g., Administration, Clinical Information). It is represented as an integer.

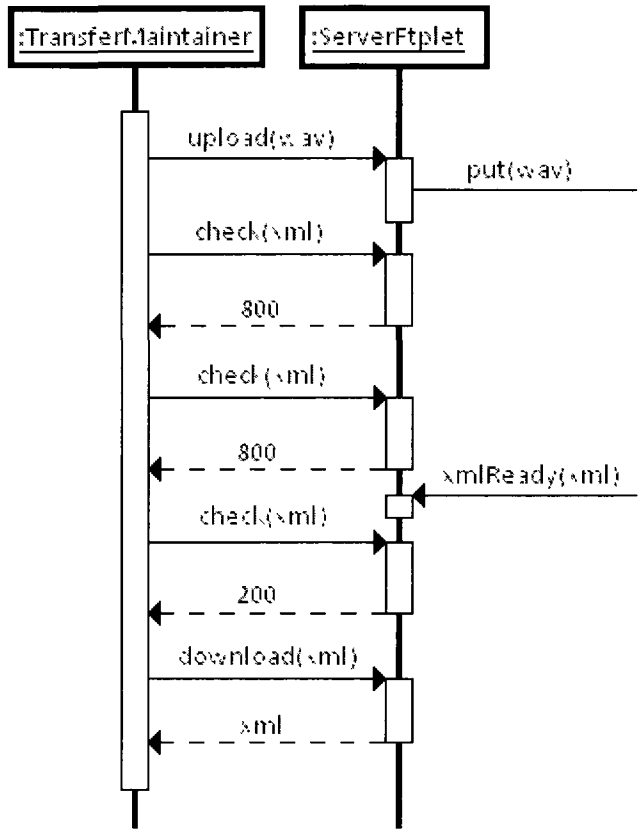


Figure 4.14: A sequence diagram depicting the WAV upload, check, XML download scenario.

SectionId Corresponds to a row from a section in the ACR that is represented by a table. Most sections only have one section id. It is represented as an integer.

Field Corresponds to a field inside a section. Each section contains multiple fields. It is represented as an integer.

Version Corresponds to the version of a recording. Every recording is stored and given a version. The version increases by one for every recording. It is represented as an integer.

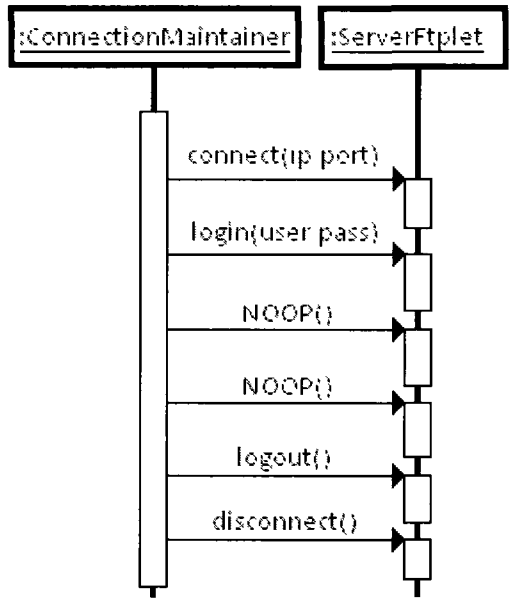


Figure 4.15: A sequence diagram depicting a connection scenario.

Urgent Modifier Corresponds to the urgency of a recording. When a recording is labeled as urgent by the user, "_urgent" is appended to the file name prefix.

4.6.6 XML Structure

The following code snippet shows the structure of a typical XML file.

```
<?xml version="1.0" standalone="yes"?>
<reportdata timestamp='1284736274226'>
   <section code='0'>
      <sectionid id='0'>
         <field position='0'>
            <confidence>green</confidence>
            <urgent>false</urgent>
            <data>Smith</data>
         </field>
```

```
      </sectionid>
   </section>
</reportdata>
```

(reportdata) timestamp Every ACR is given a unique report id calculated taking the current Unix time.

(section) code Corresponds to the Section. See above.

(sectionid) id Corresponds to the SectionId. See above.

(field) position Corresponds to the Field. See above.

confidence The confidence level of the decoding.

urgent States if a recording was labeled as urgent or not. See above.

data The ACR data. This can be generated by the Server as a result of speech recognition or by the Client as a result of manual input.

4.7 Viewer

Physicians need to be able to view the report data captured by our system. While the simplest way to achieve this would be to reuse our Client code and simply run it on a larger device, such a solution would not be optimal. While the current notebooks used by paramedics cannot be used for the Client side of our system because of their bulk, the computers available to physicians at most hospitals are more than adequate for the even simpler task of viewing the Ambulance Call Reports. Therefore, a separate application needed to be designed that could run on the various operating systems in use by physicians. Once again, the Java programming language was chosen because of its write once, run everywhere compatibility.

4.7.1 User Interface

The report viewer presents the data in a nested tab format. The top-level tabs correspond to individual sections of the report. If a section contains a table, the inner level tabs correspond to the various rows of the table. They are labeled after

the first three or four fields of the row data depending on section and availability (see Figure 4.16). Otherwise, there is only a single inner level tab that represents the section which is simply labeled after the section name (see Figure 4.17).

Each filled-out field is color-coded. A green field is non-urgent, where as a red field is urgent. If applicable, each filled out field can also contain a play-back button which will play back the recording which generated the data. If the data was filled out manually, the play-back button is omitted. Empty fields are simply left non-color coded.

The question of displaying the confidence level of data in the viewer is a complex problem. If we display the confidence level, the physician or nurse may begin to question the validity of the data. After all, there is no guarantee that even a confidence approaching 100% is correct. A simple substitution of the word "to" with the word "two" is indistinguishable at an acoustic level and not always catchable at the language model level. If we do not display the confidence level, the physician or nurse may believe that the data is always correct. They may not play back urgent fields. Further, we lose redundancy because the confidence level is only checked by the paramedic. A study is required to determine which solution may be better, if any. Our viewer is currently designed to not show the confidence level.

4.8 Discussion

Developing our system was a large undertaking. During the course of development, we encountered numerous difficulties. These ranged from high-level affecting the entire system to low-level affecting a very specific function. While most difficulties have been overcome, certain limitations still remain with our system. Some of these limitations can be solved or mitigated by reevaluating our design decisions, while others are inherent to the task the system is expected to perform.

4.8.1 Difficulties

Two Input Methods

From the very beginning, it was decided that the Client be capable of filling out an ACR using both speech recognition and manually using dialog boxes and an onscreen keyboard. This allows the user to use the appropriate input option depending on conditions. Crucially, it allows data to still be inputted manually if the Client-Server connection is lost. Implementing this functionality increased the

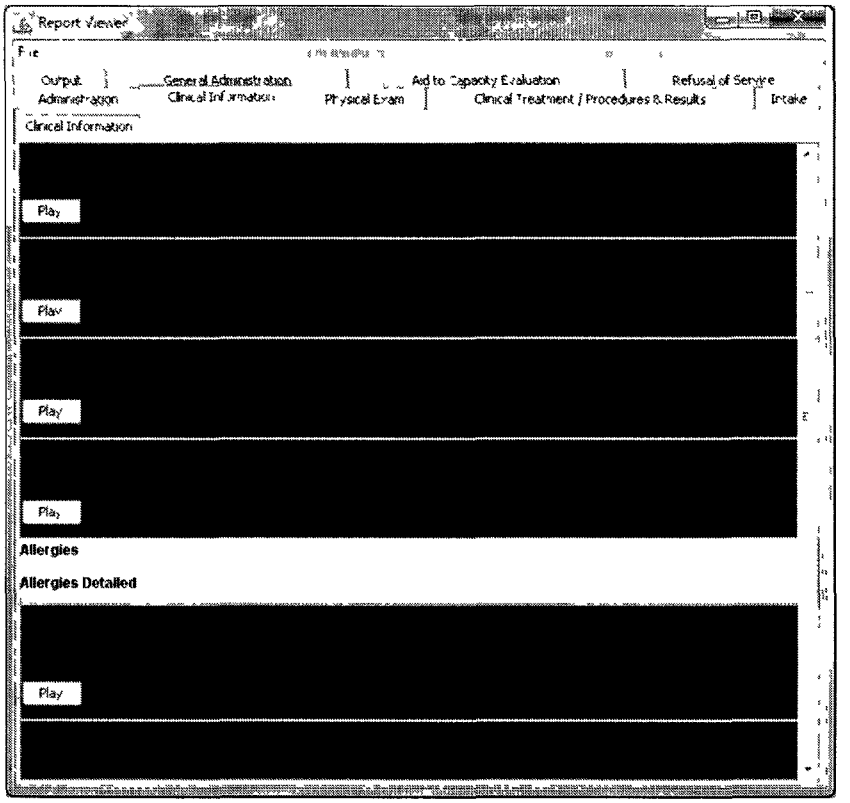


Figure 4.16: The Viewer is displaying the Clinical Information section. The Medications Detailed field is labeled urgent. Every field that contains data and was inputted using speech recognition contains a button to play back the recording. Allergies and Allergies Detailed do not contain data.

complexity of the Client as it would need to contain knowledge about every field. If incorrect data is entered in a field, the Client must inform the user of the error. If a field is to contain a code, the Client must display a list of code-description pairs appropriate for that field. Further, because all data must be written to a file regardless of how it was entered, the Client must be able to generate XML files.

Multithreading

Speech recognition is computationally intensive. Even if our Server is a high-end computer, there is likely to be a perceivable delay between the time a user finishes

recording and a result is displayed. To inform the user of the current state of the system, the field is updated with a "transferring..." message and later a "processing..." message. During this time, it is expected some users may wish to begin filling out other fields. To accomplish this, the Client needs to be multithreaded. The speech is recorded and saved in one thread, and a transfer manager uploads and downloads it in another thread. To facilitate thread communication, each of these threads has a reference to a queue. When a recording is saved, its location is put into the queue. The transfer manager takes this location and uses it to upload the recording to the Server. When the transfer manager downloads the result, it parses the XML and sends a message to a message handler in the UI thread to update the field. Multithreaded applications are notoriously difficult to debug. Threads may become starved, deadlocked, or simply be in an incorrect state for reasons not immediately apparent to the developer. Allowing users to start another recording without explicitly stopping the current recording required much effort and testing to function correctly. The feature was finally implemented using two separate overlapping locks. With Android, certain objects can also only be accessed by specific threads as is the case with View objects and the UI thread. This required us to use the message handler to actually update the field.

WAV Recording

Android 1.6 does not allow recordings in an uncompressed format which we required for our speech recognizer. Luckily, the API does allow access to the actual buffer that stores the recordings in uncompressed format. This required us to manually read the buffer and add a proper LPCM WAV header to the data. One bug which was surprising occurred when a user would stop recording immediately after starting. If stopped quickly enough, the buffer would not be filled with any data. The resulting WAV file would cause the speech recognizer to throw an exception while attempting to open the file. This was resolved by forcing each recording to contain at least some data before stopping.

Client - Server Communication

Client-Server communication is a crucial aspect of our system. Without it, speech recognition cannot take place. Our system utilizes a client-server model where the client (i.e., our Client) initiates all communications and polls the server (i.e., our Server). Because wireless connectivity can be unreliable, we had to design the Client to handle situations where the connection was lost. Further the Client needed to handle the possibility of a Server crash. When a connection is lost or

the Server is down, the Client will allow the user to continue to make recordings which are added to the queue for upload. The Client keeps track of uploaded recordings for which results have not yet been downloaded. Once the connection is reestablished, the Client resumes processing downloads and uploads in its queue. The original design of the communication utilized a peer-to-peer model where both sides could initiate communication. This allowed the Server to immediately inform the Client that a result was ready for download. Unfortunately, having such a model work reliably proved more demanding than expected and it was thus replaced with the current client-server model.

Grammar Design

The majority of the ACR fields in our system use grammars to constrain what could be recognized. A well designed grammar can greatly increase the chance a field will be filled out correctly. On the other hand, a poorly designed grammar may lead to nothing but frustration for the user. We cannot assume that the users of our system will make the same assumptions we made about how a field should be filled. Therefore, while we added hints to each field that describe the required syntax, we have also made the grammars somewhat flexible by adding optional words and sometimes even two separate syntaxes. At the same time, we had to test to make sure that expected utterances were supported by the designed syntax. A simple visual inspection of all possible utterances was enough to determine this when a syntax had only a handful of word combinations. Testing if every number from 0 to 999,999 was supported by the grammars syntax required more effort. A script needed to be designed for this task.

4.8.2 Limitations

Language Model

The current state-of-the-art speech recognition systems cannot match the accuracy of a human listener and will likely not until computers become vastly more powerful [45]. The accuracy of speech recognition systems drops steadily with increasing vocabulary size and drops markedly with non domain specific utterances. Therefore, most systems are tailored to a specific domain and our system is no exception. The language model used in our system is specific to the paramedic domain. Currently it is modeled after three ACRs which must be greatly increased if our system is to be deployed. With such a limited language model, many trigrams and even words are missing. An adequate language model would

require a very large and varied sampling of ACRs to account for the variability in how they are filled out.

Speaker Adaptation

Many speech recognition systems are speaker independent. Dragon NaturallySpeaking, Microsoft Speech, and CMU Sphinx4 are all speaker independent. The acoustic models of these systems are trained using a broad variety of speakers allowing them to work "out of the box". Because voices and pronunciation varies from person to person, adapting these models to specific speakers can improve speech recognition of these systems. Our system does not currently have built in adaptation functionality limiting the maximum speech recognition rate our system can achieve.

Noise

One of the biggest challenges facing the speech recognition field is managing noise. While speech recognition systems often work well in quiet laboratory environments, their accuracy is reduced drastically in noisy environments. Real world applications of speech recognition systems are often employed in environments that contain some background noise. This noise can originate from an endless amount of sources: wind, rain, machinery, chatter, etc... A paramedic's work environment is likely to contain all of these sources. Tian Ye et al. [69] describe three techniques to handle noise: "filtering of the noisy speech prior to classification, adaption of the speech models to include the effects of noise, [and] use [of] features that are robust to noise." Often these techniques are combined. Except for a non-speech filter, our system does not use any additional techniques to compensate for noisy environments. While this should pose no issues in our research in Chapter 5, noise will have to be considered in a deployed system.

Distance

Because of our use of WLAN, paramedics must remain in close proximity to the Server or the connection will be lost. One potential solution is to use wireless internet. This would require additional encryption as we would be sending data over an unsecured network. The ultimate solution is to merge the Server and the Client and have all processing done on the Client. If the current rate of innovation continues, mobile devices should be powerful enough within a few years to perform this task.

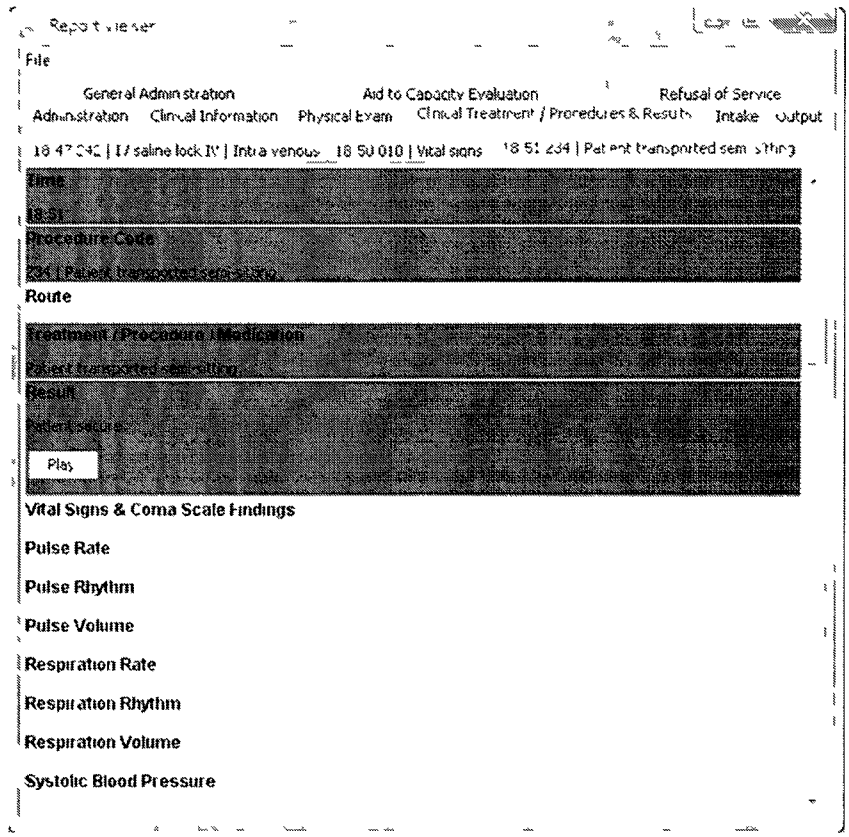


Figure 4.17: The Viewer is displaying the Clinical Treatment / Procedure & Results section with multiple inner-level tabs The inner tabs are labeled after the first three fields of the row data time, procedure code, and route The Result field has a play-back button because it was inputted using speech recognition while the other fields were inputted manually

Chapter 5

Study

Every legitimate software development model contains a testing phase. The often criticized [63] waterfall model [14] specifies testing near the very end of development. The iterative model breaks the one large waterfall development cycle up into multiple smaller cycles, while the extreme programming model takes this idea even further by writing test cases prior to writing the functions [11]. Software development models, regardless of type, provide a structured development process that is required in any larger scale project. In theory, this results in more accurate deadline planning, better use of resources, and a product that better satisfies the customer's requirements.

The testing phase in software development models often consists of unit tests, regression tests, acceptance tests and others. While these tests are effective at uncovering bugs in the software and making sure that requirements have been satisfied, they disregard usability which can lead to unsatisfied customers whose requirements have otherwise been met. Broadly speaking, usability is the study of how a user interacts with an application. Usability testing involves testing for five qualities in a user interface: learnability, efficiency, memorability, errors, and satisfaction [51]. Satisfying all five qualities can be a major challenge as they are not necessarily independent. For example, a UI that is easy to learn may sacrifice efficiency by reducing or avoiding the use of difficult to learn keyboard shortcuts. Attempting to catch all errors and making recovery easy may again reduce efficiency by inundating the user with warning dialogs. Balancing these qualities often arises when designing a UI for both novice users and power users who have different needs. When such issues arise, often designers will create a default interface for novice users, and a separate optional interface for power users. While this introduces issues of migration from one interface to the other,

it is seen as a good compromise.

5.1 Usability Testing Methods

There are various accepted methods to perform usability testing which are divided into two groups: inspection methods and test methods [33]. Inspection methods include heuristic evaluation, cognitive walk-through, and action analysis, while test methods include thinking-aloud, field observation, and questionnaires. These two groups differ in the number of users they require to be performed. Inspection methods only require a small number of evaluators in the range of two to three and no users to perform the actual tests [33]. Because their success at revealing errors rests solely on the skills of the evaluators, the evaluators must be highly skilled. A benefit of not requiring users is that inspection methods can be performed often. Test methods often require only one evaluator, but require from three to thirty or more users to perform the actual tests [33]. The role of the evaluator is to design and administer the tests, and later study the acquired data. Because the tests are performed by users, the skill of the evaluators for test methods is not as crucial for revealing errors as it is for inspection methods. Further, real users are much more likely to reveal domain related errors than evaluators. Both inspection and test methods need to be used to reveal as many errors as possible as early as possible.

5.2 Thinking-Aloud

Thinking-aloud is a common testing method that can be done with only a few users, but is quite time consuming and intrusive for the user. The benefit of thinking-aloud over field observations is that it can reveal errors that would be difficult to capture from a distance while simultaneously revealing the user's thought process. Thinking-aloud requires the user to state everything that comes to mind while completing a given task. Common statements include complaints (e.g., "this is so slow"), compliments (e.g., "that's cool") and questions (e.g., "what do I do now?"). It is crucial that the evaluators do not lead the users through the task as that will introduce evaluator bias. Instead, the evaluator's role should be as passive as possible. There are two common approaches to thinking-aloud that could be used: an early approach by Ericsson and Simon [24] and a more recent approach by Boren and Ramey [16]. Ericsson and Simon's approach minimizes the role of the evaluator. The evaluator is only allowed to remind the user to

continue speaking if they fall silent for a set amount of time. Field observations by Boren and Ramey [16] showed that thinking-aloud evaluators often deviated from Ericsson and Simon's strict rules with reminders that "ranged from short to long, from directive to nondirective, from personal to impersonal, and everywhere in between." [16] Boren and Ramey's approach bridges the gap between theory and implementation with a small set of rules that gives the evaluator the ability to gently guide the user if required. The evaluator can remind the user to continue speaking, use acknowledgment tokens (i.e. "mm hmm"), and even offer indirect suggestions. This, they are argue facilitates more natural communication between the evaluator and the user without reducing the validity of the data. A comparison [41] of the two approaches shows that both approaches reveal similar errors. Boren and Ramey's approach results in less instances of users being lost and in more successfully completed tasks, but may affect the validity of the data in tests that measure task performance and lostness [41]. For this study, Boren and Ramey's approach to thinking-aloud will be used because of the added flexibility it provides.

5.3 Objective

The paramedics will be tasked with filling out two reports. This will be accomplished by providing them with two realistic paper ACRs that will then be transferred to our system. These ACRs do not contain any real patient data. For the ACRs to appear realistic, they needed to represent a real world scenario with realistic observations and be filled out using language a paramedic would use. We therefore contacted an experienced paramedic to create such scenarios and to fill out the ACRs. While our UI has already undergone numerous iterations as a result of heuristic evaluation and consultations with paramedics, a thinking-aloud test with paramedics will provide us with valuable data that may reveal further errors. The objective of this research was to study the user interface (UI) of our system, the accuracy with which data from the paper ACR can be transferred to our system, the Word Error Rate (WER) of fields that use the language model, and user satisfaction.

5.4 Test System

Only the Client and the Server are used in this study. The Viewer, which is used by physicians and nurses to view ACRs, is not used in this study.

Because of the limited time we have with our participants for this study, our system does not adapt the acoustic model used by the speech recognizer to individual speakers. While this allows anyone to immediately begin using our system, it limits the maximum accuracy the speech recognizer can achieve. Adapting the acoustic model of our system with a few dozen sentences would noticeably increase accuracy and would be part of a deployed system.

Each field of the ACR in our system is represented by a grammar or a trigram language model to be used by the speech recognizer. The grammars are designed for high accuracy and only provide limited flexibility such as optional words and using digits instead of words. Each field that uses a grammar has its own grammar. The trigram language model is constructed using three ACRs, two of which are used for this study. Each field that uses a language model shares a common language model.

5.4.1 Hardware

Client Two HTC G1 smart phones running Android 1.6.

Server A 2 GHz 2 GB RAM Apple Macbook running OSX 10.6. A Linksys WRT54G2 802.11b/g router.

5.5 Methodology & Procedures

There were three parts to this study. First, participants were explained the concept of thinking-aloud as described by Boren and Ramey [16]. The participants were given a think-aloud trial run in the form of a questionnaire to familiarize them with the concept and to provide the researchers with some background information. The study proceeded once the participants informed the researchers that they understood the concept and were ready to continue.

Second, participants were tasked with completing two Ambulance Call Reports (ACRs). Before beginning, they were given a brief one page manual that explained the basic functionality of our system and were allowed to refer to it while filling out the ACRs. They were then given the first (see Appendix A) of two paper ACRs and the mobile device with our software solution already running, and asked to transfer the information from the paper ACRs to our mobile solution while thinking-aloud throughout. To reduce the chance of participants becoming discouraged as a result of accumulating errors, they were allowed to re-enter data. A digital camera was placed on a tripod next to the participant such that its field

of view captured the handling of the paper reports and our mobile device. The participant's face was not captured by the camera. The digital camera also served as an audio recorder and captured what the participant was saying throughout. Once the participants were done the first ACR, after a five minute break, they were given the second ACR (see Appendix B).

Following completion of this task, participants were given a brief questionnaire to answer. The questionnaire involved completion of Likert scale questions and asked about participant satisfaction with our system in detail and as a whole. In total, the length of the study for each participant was approximately two hours. Two mobile devices were on hand so that two participants could complete the study concurrently. A further two participants arrived after the first participants had completed the study.

5.6 Participants

Four paramedics participated in this study. Three were male and one was female. The youngest participant was 26 years old and the oldest 30 years old. Participants had from four and half to five and a half years of experience as paramedics. All four participants currently use Rampart CST as the ACR solution with two to three years of experience with it. All four participants stated that they use touch screens everyday. Only one participant stated they have used speech recognition before.

5.7 Results

The times required to complete the ACRs are shown in Table 5.1. ACR 1 was filled out first and ACR 2 was filled out second. For ACR 1, the fastest and slowest times were 54 minutes and 70 minutes respectively. For ACR 2, the fastest and slowest times were 34 minutes and 38 minutes respectively.

The times were calculated based on unfiltered data. For all the following tables and figures, the data was filtered to remove any invalid data. The "Treatment / Procedure / Medication" field was discarded because it was automatically filled with data that was impossible to accurately include in our analysis. The "Final Primary Problem" field was discarded because of a bug that did not update the field correctly.

The actual number of recordings and the control number of recordings to complete the ACRs are shown in Table 5.2 and Table 5.3. The control number

	ACR 1	ACR 2
Paramedic 1	59	34
Paramedic 2	68	38
Paramedic 3	54	36
Paramedic 4	70	38

Table 5.1: Time required in minutes to complete ACR 1 and ACR 2 using our system.

of recordings assumes the minimum number of recordings to complete the ACR. The number is determined by allocating one recording per field that uses a grammar and one recording per sentence of a field that uses a language model. It is theoretically possible for a participant to achieve a lower number of recordings than the control while still completing the report by uttering multiple sentences in one recording. For ACR 1, the lowest and highest percentage difference between the actual number of recordings and the control number of recordings was +36% and +57% respectively. For ACR 2, the lowest and highest percentage difference between the actual number of recordings and the control number of recordings was +29% and +54% respectively.

	Actual	Difference
Control	192	0%
Paramedic 1	292	+52%
Paramedic 2	277	+44%
Paramedic 3	262	+36%
Paramedic 4	301	+57%

Table 5.2: The actual number of recordings compared to the control number of recordings required to complete ACR 1.

To calculate ACR accuracy, we compare the number of fields filled out correctly to the number of fields to be filled out in total. For both fields that use grammars and fields that use a language model, a correct field is defined as a transfer from the paper ACR to the ACR on our system that has a Word Error Rate (WER) of 0%. The WER is defined as

$$\frac{I + D + S}{N}$$

	Actual	Difference
Control	173	0%
Paramedic 1	223	+29%
Paramedic 2	259	+50%
Paramedic 3	266	+54%
Paramedic 4	243	+40%

Table 5.3: The actual number of recordings compared to the control number of recordings required to complete ACR 2.

where I is the number of words inserted, D the number of words deleted, S the number of words substituted, and N the number of words in the reference sentence. For example, given the sentence "patient complaining of short of noted breath" and the reference sentence "patient complaining of shortness of breath", results in 1 substitution (i.e., short) and 1 insertion (i.e., noted) for a WER of 33%. We have modified the Levenshtein edit distance algorithm [1] for sequences of characters to work with sequences of words to calculate WER. Using an edit distance algorithm which produces a numbered result instead of a simple true or false allows us to study how close a field was to being correct. Table 5.4 and Table 5.5 show the number of correct fields, missed fields, added fields, and total fields filled in compared to the control. These fields can either use a grammar or a language model. The number of correct fields in the control is lower than the number of recordings because some fields require multiple recordings to complete.

	Correct Fields	Missed Fields	Added Fields	Total Fields
Control	179 (100.0%)	0	0	179
Paramedic 1	161 (89.9%)	0	4	183
Paramedic 2	160 (89.4%)	4	15	194
Paramedic 3	110 (61.5%)	21	32	211
Paramedic 4	173 (96.6%)	0	5	184

Table 5.4: The number of correct fields, missed fields, added fields. and total fields filled in compared to the control for ACR 1.

Because our system uses both grammars and a language model for fields in uneven numbers, the ACR accuracy is also broken down into only fields that use a grammar and only fields that use a language model. Table 5.6 shows the ACR

	Correct Fields	Missed Fields	Added Fields	Total Fields
Control	157 (100.0%)	0	0	157
Paramedic 1	141 (89.8%)	1	1	158
Paramedic 2	143 (91.1%)	4	20	177
Paramedic 3	129 (82.2%)	2	3	160
Paramedic 4	145 (92.4%)	1	2	159

Table 5.5: The number of correct fields, missed fields, added fields, and total fields filled in compared to the control for ACR 2.

accuracy when only fields that use a grammar are taken into account. Table 5.7 shows the ACR accuracy when only fields that use a language model and have a WER of 0% are taken into account.

	ACR 1	ACR 2
Control	158 (100%)	135 (100%)
Paramedic 1	151 (95.6%)	127 (94.1%)
Paramedic 2	148 (93.7%)	127 (94.1%)
Paramedic 3	101 (63.9%)	120 (88.9%)
Paramedic 4	155 (98.1%)	128 (94.8%)

Table 5.6: The ACR accuracy when only fields that use a grammar are taken into account.

A WER of 0% for fields that use a language model is not required, nor often expected, yet this data can still be of use to the reading nurse or physician. Therefore, we also calculate the ACR accuracy when only fields that use a language model and have a WER of less than 15% are taken into account. Table 5.8 shows the effects on ACR accuracy that this relaxation makes.

Results for Paramedic 3 filling out ACR 1 reveal an anomaly. A brief investigation of the data reveals that the participant skipped a field and proceeded to fill out the remaining fields a field behind the correct field. This error caused a large sequence of otherwise correctly filled out and correctly ordered fields to be labeled as incorrect. Therefore, this data is excluded from the Figure 5.1 and Figure 5.2.

Figure 5.1 shows the average ACR accuracies for ACR 1 and ACR 2 using 3 and 4 paramedics respectively. When taking into account only fields that use

	ACR 1	ACR 2
Control	21 (100%)	22 (100%)
Paramedic 1	10 (47.6%)	14 (63.6%)
Paramedic 2	12 (57.1%)	16 (72.7%)
Paramedic 3	9 (42.9%)	9 (40.9%)
Paramedic 4	18 (85.7%)	17 (77.3%)

Table 5.7: The ACR accuracy when only fields that use a language model and have a WER of 0% are taken into account.

	ACR 1	ACR 2
Control	21 (100%)	22 (100%)
Paramedic 1	11 (52.4%)	17 (77.3%)
Paramedic 2	13 (61.9%)	18 (81.8%)
Paramedic 3	11 (52.4%)	9 (40.9%)
Paramedic 4	9 (90.5%)	18 (81.8%)

Table 5.8: The ACR accuracy when only fields that use a language model and have a WER of less than 15% are taken into account.

grammars, average ACR accuracies for ACR 1 and ACR 2 were 95.8% and 93.0% respectively. When taking into account only fields that use a language model, average ACR accuracies for ACR 1 and ACR 2 were 63.5% and 63.6% respectively. This discrepancy is hidden when average ACR accuracies take into account both types of fields because of the much greater number of fields that use grammars. When both fields that use grammars and fields that use the language model are taken into account, average ACR accuracies for ACR 1 and ACR 2 are 92.0% and 88.9% respectively. Figure 5.2 shows the ACR WER for fields using a language model. These values were calculated by using the sum of all words in the fields and the sum of all of words in the reference sentences.

Finally, Table 5.9 shows the results of the questionnaire which participants were asked to fill out after completing both ACRs. It is based on a 5 level Likert scale where 1 is strongly disagree and 5 is strongly agree. The median value is shown for every question.

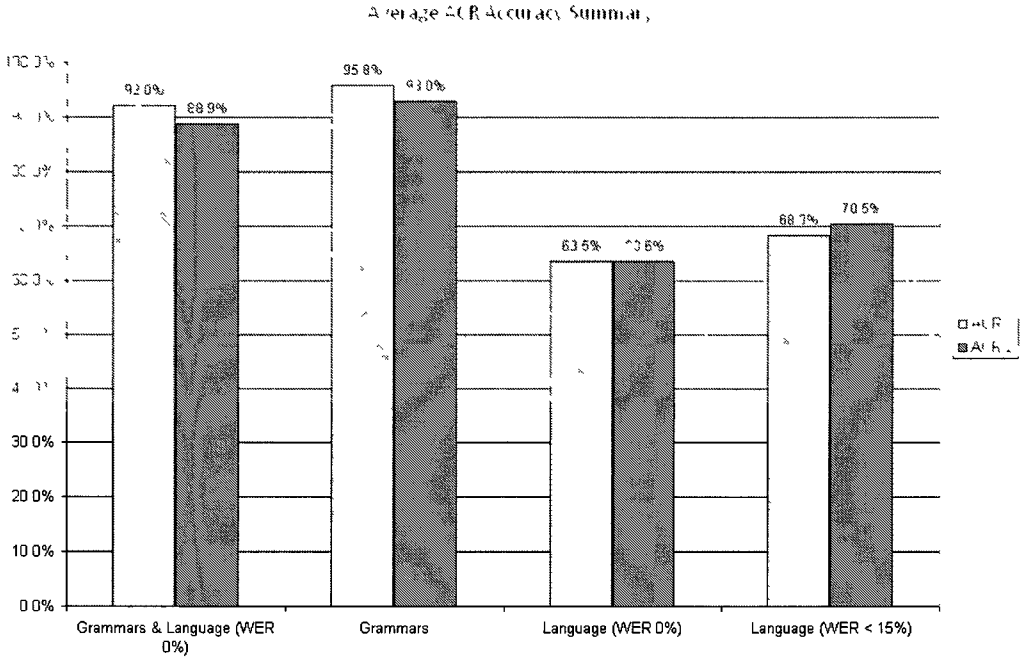


Figure 5.1: Average ACR accuracy summary.

5.8 Discussion

5.8.1 Completion Time & ACR Accuracy

The two ACRs the participants filled out contained similar number of fields. ACR 1 required 179 fields to be filled out and ACR 2 required 157 fields. They were expected to be completed in 192 recordings and 173 recordings respectively. An almost 50% drop in the time required to complete the second ACR while maintaining similar ACR accuracies and similar increases in actual recordings required shows that participants quickly adjusted to our system. Based on our observations, a third ACR would have likely shown a further, although less dramatic, reduction in time required. The times can further be reduced with faster speech recognition processing. The Server struggled to handle two ACRs being completed at once. Delays of up to 10 seconds occurred when two long recordings were sent to the Server simultaneously. Fine tuning our speech recognizer or simply a more powerful computer would alleviate these issues.

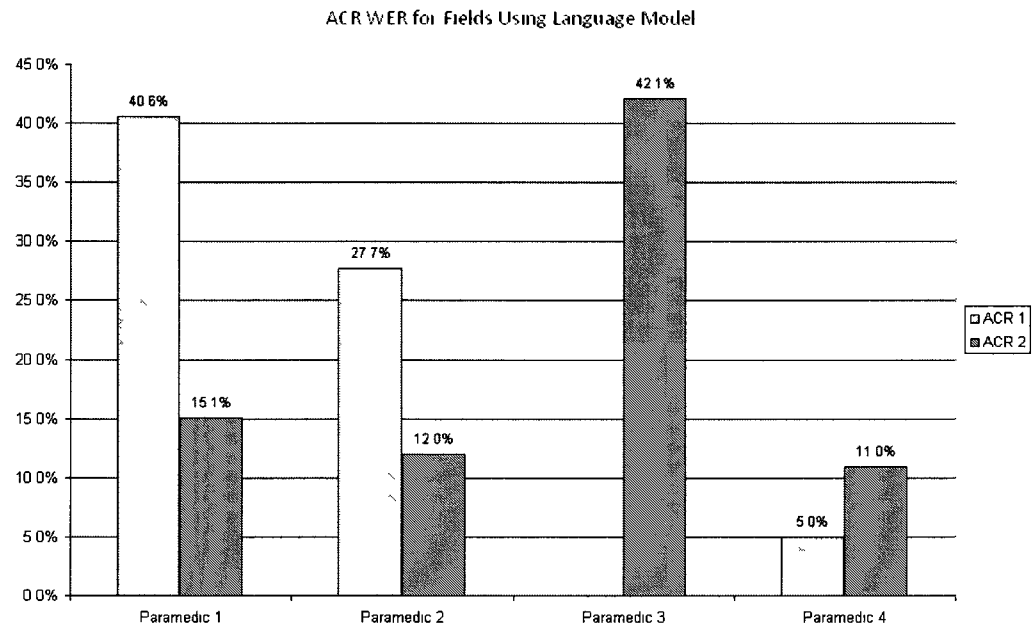


Figure 5.2: Word Error Rate for fields using a language model.

Tables 5.4 and 5.5 show an average ACR accuracy of 92.0% for ACR 1 and 88.9% for ACR 2 when taking into account fields that use a grammar and fields that use a language model. These results were attained by giving the participants the ability to correct errors and are thus likely higher than if this was a typical accuracy and Word Error Rate study. At the same time, these results are too low for system that allows corrections to be made. Tables 5.2, 5.3 show that mistakes resulted in a 36% to 57% increase in the number recordings to complete ACR 1 and a 29% to 54% increase in the number recordings to complete ACR 2. A more detailed analysis below reveals that these values can be significantly reduced without major changes to our system.

When taking into account only fields that use grammars, average ACR accuracies increased for ACR 1 to 95.8% and for ACR 2 to 93.0% (see Table 5.6). There are numerous contributing factors to this high level of accuracy. The grammars in our system have a strict syntax and small vocabulary. As discussed in Chapter 3 and Chapter 4, such grammars increase the likelihood of correct recognition by

I found the system easy to use	4
I enjoyed using speech for data input	4
I felt the language used for speech input to be adequate	4
I would use this system in the field	4
I found the recognition accuracy to be sufficiently high	3.5
I found the delay for processing to be acceptable	3
I would consider using this system in real time while patient care is being provided	3.5
I would be able to complete a report faster with this system than with current solutions	3.5

Table 5.9: The questionnaire given to participants after they completed both ACRs. The median value is shown.

decreasing the size of the search graph. The majority of grammars in our system are designed to accept connected digits, twenty four hour time, or checkbox descriptions. If the user uses the correct syntax, correct recognition is highly likely. Further, the speech recognizer in our system is set up when using grammars to force match a result regardless of how poorly it scores the incoming features. This allows even poorly articulated utterances to often return correct results. The easy correction of errors also contributed to the high accuracy. Because fields using grammars did not append results of multiple recordings, rerecording such fields simply overwrote the prior result. A major factor keeping accuracy lower than expected is that participants simply did not follow the correct syntax for a field. This occurred most often for time fields where participants were expected to record the time in twenty four hour format. Instead of speaking the time in plain English such as "fifteen thirty one", they would speak "one five three one". Simply adding digits to the grammar's syntax would resolve many errors. Further, users are more likely to understand the syntax with more use.

When taking into account only fields that use the language model and have Word Error Rates of 0%, average ACR accuracies decreased for ACR 1 to 63.5% and for ACR 2 to 63.6% (see Table 5.7). While the language model was expected to be outperformed by the grammars, such a severe difference was unexpected. The language model was constructed using three ACRs, two of which were used in this study. The small size of the resulting language model together with the fact that the same sentences were used for both modelling and input was expected to produce higher accuracies.

When recordings by the same participant for identical fields were compared, it was revealed that participants would often introduce a long pause to the beginning of the recording by pressing the record button and then reading what they had to say. This pause would be filled with background noise and be interpreted by the recognizer as speech resulting in additional words in the result. While a filter was added to remove this noise, tuning it to more aggressively remove noise may help resolve these errors. Training the paramedics to only start recording when they are ready to speak would further reduce the likelihood of these errors.

When recordings by different participants for identical fields were compared, it was revealed that errors were also produced as a result of different pronunciations. The speech recognizer will only recognize words according to a predefined pronunciation in a vocabulary list. While the tool used to construct a language model automatically includes multiple pronunciations, it does not include every possible pronunciation. Further, alternative pronunciations of words it does not understand must be added manually. This is the case for the many medical terms used while filling out an ACR. While we added alternative pronunciations for various words in our vocabulary, we did not for the word "patent". In medical terms, the "a" sound is pronounced similar to the "a" sound in "patient" which it was often mis-recognized as. More surprisingly, it was also often mis-recognized as "eight". These types of errors can be resolved by studying the pronunciations of words by paramedics and adding them to the vocabulary.

To allow users to correct errors while maintaining the ability to append results, the commands "clear" which cleared the entire field, and "undo" which undid the most recent appended result, were added to the language model. It was thus surprising that many errors and extra recordings were a direct result of the commands being mis-recognized. The resulting mis-recognized sentence was appended to the previous incorrect result compounding the error. One participant had such a low success rate with "clear" that they began using only "undo" with which they had a greater success rate. Unfortunately "undo" only allows the previous result to be undone, and functions as a redo if used twice in a row. The clear and undo functionality needs to be reworked. Adding buttons to clear and undo is a solution, but it would use up more of the already limited screen real estate. Another potential solution is to increase the length of the command to two words such as "a-c-r undo" and "a-c-r clear". Doing so allows the language model to use bigram probabilities instead of unigram probabilities which will produce better search graphs and thus higher recognition accuracies. A two word command is also less likely to be misinterpreted as a non command than a single word command.

The longest field the participants had to input contained 49 words in 5 sentences. A single word error would result in this field being labeled as incorrect even though the data remained useful. It has been shown that WERs of up to 25% are acceptable for lecture and presentation webcasts [49]. For medical reports, WERs of up to 20% have been shown to be acceptable before relevant information is lost [32]. We therefore decided to test the ACR accuracy again when fields using the language model with an even more cautious acceptable WER of less than 15% are taken into account. Table 5.8 shows that average ACR accuracies improved in 7 out of 8 ACRs with one achieving an increase of almost 15% percent. Relaxing the acceptable WER must be done carefully to avoid compromising the data in an attempt to raise the ACR accuracy. Further studies would have to be done to determine what WER is acceptable for ACRs.

5.8.2 Thinking-Aloud

Participants were told to think-aloud while filling out the ACRs. While we used Boren and Ramey's more flexible implementation of thinking-aloud, we found that participants did not require acknowledgment tokens to continue speaking. Naturally, participants tended to think-aloud less when their task was proceeding well, and more when experiencing difficulties. Because the evaluator was sitting next to the participant, both mannerisms and the handling of the device were also captured. The data obtained from these observations provided us with a much better understanding of where our UI performed well and where it did not. Further, these observations can often be directly related to the ACR accuracies and Word Error Rate results discussed earlier.

Observation 1

Each field in our system displayed a hint stating the syntax required for correct speech recognition. Users were instructed to follow these hints and even given a more detailed explanation of them in a one page manual. Even with these precautions, participants often did not follow the correct syntax. Observations revealed at least two reasons for why this occurred.

Because an ACR is heavily codified, many fields use a syntax that required digits. Further, fields which contain numbers such as age, weight, and blood pressure use a syntax that allowed both numbers and digits. As a result, participants became accustomed to using digits and would attempt to fill out fields that use a twenty four hour time syntax using digits. While practice should resolve

this issue, allowing digits to be used in such fields would prevent this issue from occurring at all.

Another reason for these errors is that hints would only be displayed on fields that contained no data. The hint would be hidden once the field was filled with any data. If a participant were to fill out the field incorrectly the first time, unless it was cleared in the case of fields that used a language model, they had no way to view the syntax. One potential solution is to display the syntax in a pop up box while recording. Another is to allow every field to be cleared in some manner thus revealing the hint again.

Observation 2

When a numeric field was mis-recognized, participants would often revert to using digits regardless if the field supported them. Such an observation is not entirely unexpected. A method which works great only part of the time can be less desirable than a method which works well enough most of the time even if the latter is slower on average. The only solution to this is to either increase the accuracy of the preferred syntax to such a level that participants will have confidence in it, or to allow digits to be used in all numeric fields as described earlier.

Observation 3

There are two ways to record a field: by pressing the record button on the specific field, and by pressing the larger record button for the section. Pressing the section record button while already recording stops the current recording and starts the next recording. Further it automatically scrolls to the next field. Participants took advantage of this functionality as it eliminated the need to specifically press the stop button and to scroll. An issue arises when this functionality is used when the following field should actually be skipped. If the field cannot be cleared as is the case for fields that use grammars, the user is forced to simply ignore the error and continue. A possible solution is to add a skip button that can be pressed while recording to discard the current recording and skip to the following field. While this would reduce the chance of unwanted fields being filled with data, functionality must be added that allows all fields to be cleared.

Observation 4

While participants had little to no experience with speech recognition prior to this study, they still had various preconceptions which can be difficult to change.

A classic misconception is that you must speak louder over the phone to people who are long distances away. While this may have been true many decades ago with analog lines, it is quite likely that most users never used such phones in their lives. People have a similar misconception about speech recognition. When a recording was mis-recognized, participants would tend to speak louder or hold the device closer to their mouths. While this is the correct action to take if the microphone is held too far or the user speaks too quietly, taking these actions too far actually reduces the chance of correct recognition. Microphones can only accurately work to some maximum sound pressure level. Any higher and they simply begin clipping the input signal. Both moving the microphone closer and speaking louder results in higher sound pressure levels. Another misconception with speech recognition is that speaking slower results in a higher chance of correct recognition. While modern speech recognizers work with variable speaking rates because they are based on Hidden-Markov-Models (HMM) that allow states to transition to themselves, a word is more likely to be mispronounced when spoken too slowly. There is no quick solution to user misconceptions. The system must simply work reliably over a long period of time.

Observation 5

Although participants were told that we were not expecting absolute accuracy, every participant attempted corrections to some degree. While some participants attempted more corrections than others, it was their reactions afterwards that provided the most information. If after numerous corrections the resulting data in the field was correct, participants appeared satisfied regardless of the number of attempts required. Since speech recognition has not yet reached levels approaching human recognition, it may be wiser to develop comprehensive correction functionality before attempting to maximize speech recognition accuracy.

Observation 6

Our system does not use any title bars to maximize the amount of fields that can be displayed on the screen. Such a design led to some confusion among participants who expected to be able to scroll to the first field of the following section from their current section. One participant believed that they could only continue after filling out the last field of the current section even though the field was to contain no data. With a title bar, participants would have been more likely to realize that they are in a specific section and must go back to the menu to select the following section. Further, with a title bar we can design our UI

to skip to the following section if a user attempts to scroll down at the end of a section while clearly conveying to the user what section they are in. With a larger screen, a title bar should be added to the UI.

Observation 7

Although our study asked the participants to use speech recognition to fill out the fields, our system allows the ACR to be filled out completely using dialog boxes and an on-screen keyboard. To switch between the two input options, there is a button on the screen that switches between speech input and manual input for every field. Such a solution reduces clutter while still providing the required functionality. Unfortunately, one participant pressed the switch input button and did not know that they could press it again to return to speech input. Only after several fields were filled out manually did that user realize there must be a way to switch back to speech input. While this was not a major issue, perhaps a different button description is required.

5.8.3 Questionnaire

The results of the questionnaire in Table 5.9 show that the effort put into UI, grammar, and language model designs was rewarded. It was promising to see that the participants would be willing to use this system in the field, although the neutral response to using the system while patient care is being provided shows that the system needs to be less intrusive. Poorly designed correction functionality contributed heavily to this intrusion and must be redesigned. The neutral response to the processing delay is expected. While participants could continue to the next field before the current field was returned a result, they would often wait to confirm the result was correct. Telling the user that a result is being processed gives feedback to the user that the system is working, but does not give them enough information to confidently continue to the next field. Finally it was unlikely that our system would outperform the system that the participants currently use. They have been using their system for many years and our system for only two ACRs. One paramedic noted that when their current system was introduced, some paramedics required over two hours to complete their first ACR and that even experienced users struggle with the system. Considering all four participants completed their first ACR with our system in approximately one hour, and the second in slightly over half an hour, we are cautiously optimistic that our system can eventually compete with the current system in this regard.

5.9 Conclusion

Having real world users test our system was sometimes stressful, but also very rewarding. The participants did not have difficulties with thinking-aloud. While their voices were sometimes low, they continued to talk throughout their tasks and appeared to enjoy participating in the study. Even with only four participants, the study produced a substantial amount of data. While final accuracies of the ACRs were high, they were achieved by allowing participants to correct fields that were mis-recognized. Participants expected to be able to correct any fields which our system did not allow and were frustrated when they could not. For fields that our system did allow corrections, the functionality was flawed and contributed unnecessary recordings and errors. The correction functionality must be expanded to all fields and its flaws corrected. Thankfully, these problems can be fixed in relatively little time.

The most important result of this study is that it shows that speech recognition is mature enough to be seriously considered for critical tasks such as filling out ACRs. Even with our system in its current state, ACRs can be completed with high accuracy. Further, none of the issues we observed appear insurmountable giving us confidence that a fully functioning ACR solution using speech recognition will eventually be realized.

Chapter 6

Conclusion

Designing an ACR system from the ground up was a large undertaking and augmenting it with speech recognition only increased the challenge. For this thesis, we have designed a speech recognition enabled ACR system that can be worn by paramedics. It consists of a Client which the user directly interacts with, a Server which is connected to the Client wirelessly and performs the speech recognition, and a Viewer that can be used to view the completed reports. Report accuracy was a major concern throughout development. We used grammars for as many fields as possible as they can result in higher recognition rates compared to language models. For fields where language models were used, we implemented correction functionality to handle the expected lower recognition rates. We did not wish to compound speech recognition errors with user errors as a result of poor UI design. The UI was therefore designed to be simple and clutter free in an attempt to display as much useful information on the small screen as possible. Finally, users are given the ability to input data through speech recognition, and manually through dialog boxes and an on-screen keyboard.

The study produced valuable data about the UI and the accuracy of our system. Average ACR accuracies were 90% taking all fields into account. While final accuracies of the ACRs were high, results showed that fields that used a grammar were much more likely to be correct than fields that used a language model. Average ACR accuracies were 95% when only taking fields that use a grammar into account. Average ACR accuracies were between 63% and 69% when only taking fields that use a language model into account. A detailed analysis of the data showed that the accuracy of fields using a language model were low because of design flaw in our correction functionality and vocabularies lacking multiple pronunciations. The study also revealed the limitations of using

a small display size for our Client. Users became lost while filling out the ACRs because of the limited amount of information that can be displayed at one time. This was most evident with syntax hints only being visible in empty fields causing users to guess the correct syntax on successive attempts.

Canada's aging population is expected to put a strain on our health care budget over the coming decades. By 2030, seniors will make up 22% of the population compared to the current 14% [2]. In an effort to improve patient care and reduce costs, Canada will be investing $20 billion into e-Health over the next decade [58]. Our system is just one of many speech recognition systems in development. It distinguishes itself by being designed for use by paramedics instead of physicians as is often the case. With further improvements, our system can likely reach accuracies 95% and above making it viable for real world applications. Based on the knowledge we acquired during the course of these thesis, we believe that speech recognition enabled ACRs should no longer be relegated to fictional future solutions, and are in fact realistic solutions that can be implemented today.

6.1 Future Work

While we believe the system design was successful, results from the study in Chapter 5 revealed that optimizations can still be made. For instance, the Server component should eventually be merged into the Client as mobile devices become more powerful. While server based speech recognition systems are beneficial in situations where multiple users use one device, users of our system each have their own device. Merging these two components would remove the maximum distance issue of our wireless connection while increasing security.

The effects of noise on the accuracy of our system must be studied as it is known to greatly reduce accuracy in some situations. Paramedics are continuously surrounded by vehicle noise and surrounding chatter. Potential solutions are high quality noise canceling microphones, and training of the acoustic model to contain background noise. Our system must be made resistant to the negative effects of noise.

Our system is currently optimized to fill out the reports top to bottom, where as information may be not be gathered in the same order. While our system allows ACRs to be filled out in any order, accelerators such as auto scrolling and the recording of successive fields without explicitly stopping recording in between are not designed for this. Such a design came as a result of the limited display size and resolution of the Client. Larger display sizes and higher resolutions can potentially alleviate this and many other UI limitations. Therefore, the effects

of porting our Client to a larger (e.g., 7-inch) device must also be studied in the future.

Finally, a collaboration with Paramedic Services should be pursued in the future. It would allow us to conduct field observations, focus groups, and field trials. Field trials are especially important if our system is to be deployed in a real world situation. A collaboration with paramedics in training could also be pursued. Paramedics in training would not have the experience with current systems that experienced paramedics have allowing us to better compare our system to other ACR systems.

Glossary

ACP Advanced Care Paramedic. Preceded by PCP. Proceeded by CCP. 12

ACR Ambulance Call Report. Contains information relevant to the call. Filled out by paramedics for every call handled. 15

AMPDS Advanced Medical Priority Dispatch System. Dispatch system used in Toronto and Niagara region. 10

CCP Critical Care Paramedic. Preceded by ACP. 12

DPCI Dispatch Priority Card Index. Dispatch system used in Ottawa. 9

grammar Stores the syntax of acceptable utterances. Typically used for commands. 29

language model Based on a n-gram model, stores the probability of a word appearing given the preceding n-1 words. Typically used for dictation. 28

PCP Primary Care Paramedic. Proceeded by ACP. 11

phoneme The most basic linguistic unit used to differentiate spoken utterances. 20

WER Word Error Rate. The number of word insertions, deletions, and substitutions divided by the number of words in a reference sentence. 73

Bibliography

[1] Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[2] Canada's Aging Population. `http://dsp-psd.pwgsc.gc.ca/Collection/H39-608-2002E.pdf`, 2002.

[3] City of Ottawa - About Paramedics. `http://www.ottawa.ca/residents/ambulance/structure_contact/operations/working_en.html`, 2010.

[4] City of Ottawa - Ambulances & Paramedics. `http://www.ottawa.ca/residents/ambulance/index_en.html`, 2010.

[5] City of Ottawa - Definitions and Explanatory Notes. `http://www.ottawa.ca/city_hall/ottawa_performance/quarterly_performance/2009/q3/definitions_en.html`, 2010.

[6] City of Ottawa - Paramedic Services. `http://www.ottawa.ca/city_hall/ottawa_performance/quarterly_performance/2009/q1/para_en.html`, 2010.

[7] Sphinx4. `http://cmusphinx.sourceforge.net/sphinx4/`, 2010.

[8] Alexandre Alapetite, Henning Boje Andersen, and Morten Hertzum. Acceptance of speech recognition by physicians: A survey of expectations, experiences, and social influence. *Int. J. Hum.-Comput. Stud.*, 67:36–49, January 2009.

[9] Apache. FtpServer. `http://mina.apache.org/ftpserver/`, 2010.

[10] J. Baker. The dragon system–an overview. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 23(1):24 – 29, feb. 1975.

[11] K. Beck. Embracing change with extreme programming. *Computer*, 32(10):70 –77, October 1999.

[12] M. I. Bernstein. Interactive systems research: Final report to the director, advanced research projects agency. Technical report, SYSTEM DEVELOPMENT CORP SANTA MONICA CALIF, November 1975.

[13] Thomas H. Blackwell, Jeffrey A. Kline, J. Jeffrey Willis, and G. Monroe Hicks. Lack of association between prehospital response times and patient outcomes. *Prehospital Emergency Care*, 13(4):444–450, 2009.

[14] B. W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61 –72, May 1988.

[15] Eric Boersma, Arthur CP Maas, Jaap W Deckers, and Maarten L Simoons. Early thrombolytic treatment in acute myocardial infarction: reappraisal of the golden hour. *The Lancet*, 348(9030):771 – 775, 1996.

[16] T. Boren and J. Ramey. Thinking aloud: reconciling theory and practice. *Professional Communication, IEEE Transactions on*, 43(3):261 –278, September 2000.

[17] John Brandon. 802.11n: Fact vs. Fiction. `http://archive.laptopmag.com/features/common-802-11n-myths-debunked.ht`, February 2007.

[18] Canalys. Android smart phone shipments grow 886% year-on-year in Q2 2010. `http://www.canalys.com/pr/2010/r2010081.html`, August 2010.

[19] James Chen, Paras Lakhani, Nabile Safdar, and Paul Nagy. Letter to the editor re: Voice recognition dictation: Radiologist as transcriptionist and improvement of report workflow and productivity using speech recognition-a follow-up study. *Journal of Digital Imaging*, 22:560–561, 2009. 10.1007/s10278-009-9197-5.

[20] K. H. Davis, R. Biddulph, and S. Balashek. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642, 1952.

[21] R. de Mori, L. Gilli. and A.R. Meo. A flexible real-time recognizer of spoken words for man-machine communication. *International Journal of Man-Machine Studies*, 2(4):317 – 322, IN1, 323–326, 1970.

[22] Eric G Devine, Stephan A Gaehde, and Arthur C Curtis. Comparative Evaluation of Three Continuous Speech Recognition Software Packages in the Generation of Medical Reports. *Journal of the American Medical Informatics Association*, 7(5):462–468, 2000.

[23] Scott Durling and Jo Lumsden. Speech recognition use in healthcare applications. In *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '08, pages 473–478, New York, NY, USA, 2008. ACM.

[24] K. Anders Ericsson and Herbert A. Simon. Verbal reports as data. *Psychological Review*, 87(3):215 – 251, 1980.

[25] K. Anders Ericsson and Herbert A. Simon. How to Study Thinking in Everyday Life: Contrasting Think-Aloud Protocols With Descriptions and Explanations of Thinking. *Mind, Culture, and Activity*, 5(3):178–186, October 1998.

[26] Lee D. Erman and Victor R. Lesser. The hearsay-ii speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12:213–253, 1980.

[27] G. Eysenbach. What is e-health? *J Med Internet Res*, 3(2):e20, Jun 2001.

[28] G. Fant. Automatic recognition and speech research. *STL-QPSR*, 11(1):016–031, 1970.

[29] J. Fitch. Response times: myths, measurement & management. *JEMS*, 30(9):47–56, September 2005.

[30] Jr. Forney, G.D. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268 – 278, mar. 1973.

[31] D.B. Fry. Theoretical aspects of mechanical speech recognition. *Radio Engineers, Journal of the British Institution of*, 19(4):211 –218, apr. 1959.

[32] C. Glaser, C. Trumm, S. Nissen-Meyer, M. Francke, B. Kuttner, and M. Reiser Spracherkennung Auswirkung auf workflow und befundverfugbarkeit. *Der Radiologe*, 45:735–742, 2005. 10.1007/s00117-005-1253-7.

[33] Andreas Holzinger. Usability engineering methods for software developers. *Commun. ACM*, 48(1):71–74, 2005.

[34] Xuedong Huang, Fileno Alleva, Mei-Yuh Hwang, and Ronald Rosenfeld. An overview of the sphinx-ii speech recognition system. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pages 81–86, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

[35] IDC. Press Release. `http://www.idc.com/about/viewpressrelease.jsp?containerId=prUK22402810&sectionId=null&elementId=null&pageType=SYNOPSIS`, June 2010.

[36] Robert M. Issenman and Iqbal H. Jaffer. Use of Voice Recognition Software in an Outpatient Pediatric Specialty Practice. *Pediatrics*, 114(3):e290–293, 2004.

[37] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532 – 556, apr. 1976.

[38] Hyunseok Peter Kang, S. Joseph Sirintrapun, Rick J. Nestler, and Anil V. Parwani. Experience With Voice Recognition in Surgical Pathology at a Large Academic Multi-Institutional Center. *American Journal of Clinical Pathology*, 133(1):156–159, 2010.

[39] Dennis H. Klatt. Review of the arpa speech understanding project. *The Journal of the Acoustical Society of America*, 62(6):1345–1366, 1977.

[40] Mika Koivikko, Tomi Kauppinen, and Juhani Ahovuo. Improvement of report workflow and productivity using speech recognition-a follow-up study *Journal of Digital Imaging*, 21:378–382, 2008. 10.1007/s10278-008-9121-4.

[41] E. Krahmer and N. Ummelen. Thinking about thinking aloud: a comparison of two verbal protocols for usability testing. *Professional Communication, IEEE Transactions on*, 47(2):105 – 117, 2004.

[42] Michel R. Le May, Derek Y. So, Richard Dionne, Chris A. Glover, Michael P.V. Froeschl, George A. Wells, Richard F. Davies, Heather L. Sherrard, Justin Maloney, Jean-Francois Marquis, Edward R. O'Brien, John Trickett, Pierre Poirier, Sheila C. Ryan, Andrew Ha, Phil G. Joseph, and Marino Labinaz. A Citywide Protocol for Primary PCI in ST-Segment Elevation Myocardial Infarction. *New England Journal of Medicine*, 358(3):231–240, 2008.

[43] K.-F. Lee, H.-W. Hon, and R. Reddy. An overview of the sphinx speech recognition system. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(1):35 –45, jan. 1990.

[44] E. B. Lerner and R. M. Moscati. The Golden Hour: Scientific Fact or Medical "Urban Legend"? *Academic Emergency Medicine*, 8(7):758–760, July 2001.

[45] Richard P. Lippmann. Speech recognition by machines and humans. *Speech Communication*, 22(1):1 – 15, 1997.

[46] Bruce T. Lowerre. *The harpy speech recognition system.* PhD thesis, Pittsburgh, PA, USA, 1976.

[47] Bruce Lucas, Will Walker, and Andrew Hunt. ECMAScript Action Tags for JSGF. `http://x-language.tripod.com/ECMAScriptActionTagsforJSGF.html`, September 1999.

[48] David N Mohr, David W Turner, Gregory R Pond, Joseph S Kamath, Cathy B De Vos, and Paul C Carpenter. Speech Recognition as a Transcription Aid: A Randomized Comparison With Standard Transcription. *Journal of the American Medical Informatics Association*, 10(1):85–93, 2003.

[49] Cosmin Munteanu, Gerald Penn, Ron Baecker, and Yuecheng Zhang. Automatic speech recognition for webcasts: how good is good enough and what to do when it isn't. In *Proceedings of the 8th international conference on Multimodal interfaces*, ICMI '06, pages 39–42, New York, NY, USA, 2006. ACM.

[50] Miroslav Nagy, Petr Hanzlicek, Jana Zvarova, Tatjana Dostalova, Michaela Seydlova, Radim Hippman, Lubos Smidl, Jan Trmal, and Josef Psutka. Voice-controlled data entry in dental electronic health record. *Studies in health technology and informatics*, 136:529–534, 2008.

[51] J. Nielsen. The usability engineering life cycle. *Computer*, 25(3):12 –22, mar. 1992.

[52] Julian Odell and Kunal Mukerjee. Architecture, user interface, and enabling technology in windows vista's speech systems. *IEEE Transactions on Computers*, 56:1156–1168, 2007.

[53] Government of Ontario. Ambulance Act, ONTARIO REGULATION 257/00. http://www.e-laws.gov.on.ca/html/regs/english/elaws_regs_000257_e.htm, 2010.

[54] Government of Ontario. Ontario Paramedic Program Overview. http://www.health.gov.on.ca/english/public/program/ehs/edu/paramed.html, 2010.

[55] Office of the Auditor General. Audit of the Ottawa Paramedic Service 2008. http://www.ottawa.ca/city_hall/mayor_council/auditor_general/audit_reports/2008/images/paramedic_service.pdf, 2008.

[56] Hans Oh, Carlos Rizo, Murray Enkin, and Alejandro Jadad. What is ehealth (3): A systematic review of published definitions. *J Med Internet Res*, 7(1):e1, Feb 2005.

[57] Brett Patterson. The Wheel Reinvented. http://www.emergencydispatch.org/articles/thewheel.html, 2007.

[58] Paul Christopher Webster. Canada's ehealth software "tower of babel". http://www.cmaj.ca/earlyreleases/2nov10-canada-ehealth-software-tower-of-babel.dtl, November 2010.

[59] John Pezzullo, Glenn Tung, Jeffrey Rogg, Lawrence Davis, Jeffrey Brody, and William Mayo-Smith. Voice recognition dictation: Radiologist as transcriptionist. *Journal of Digital Imaging*, 21:384–389, 2008. 10.1007/s10278-007-9039-2.

[60] J. R. Pierce. Whither speech recognition? *The Journal of the Acoustical Society of America*, 46(4B):1049–1051, 1969.

[61] Douglas J. Quint. Voice recognition: Ready for prime time? *Journal of the American College of Radiology*, 4(10):667 – 669, 2007.

[62] D. R. Reddy. Computer recognition of connected speech. *The Journal of the Acoustical Society of America*, 42(2):329–347, 1967.

[63] W. W. Royce. Managing the development of large software systems: concepts and techniques. In *ICSE '87: Proceedings of the 9th international conference on Software Engineering*, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[64] A. L. Samuel. "whither speech recognition?"—a rebuttal. *The Journal of the Acoustical Society of America*, 47(6B):1616–1616, 1970.

[65] David L Schwappach. The emotional impact of medical error involvement on physicians: a call for leadership and organisational accountability. *Swiss medical weekly*, 139(1-2):9–15, January 2009.

[66] MD MSc Stiell, Ian G., MHA Nesbitt, Lisa P., PhD Pickett, William, MD Munkley, Douglas, MD Spaite, Daniel W., CHIM Banek, Jane, MBA EMCA Field, Brian, BScN MHA Luinstra-Toohey, Lorraine, MD Maloney, Justin, MD Dreyer, Jon, MD Lyver, Marion, MAEd PhD Campeau, Tony, PhD Wells, George A., and for the OPALS Study Group. The OPALS Major Trauma Study: impact of advanced life-support on survival and morbidity. *CMAJ*, 178(9):1141–1152, 2008.

[67] Steve Tauroza and Desmond Allison. Speech Rates in British English. *Applied Linguistics*, 11(1):90–105, 1990.

[68] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 bit wep in less than 60 seconds. In *WISA '07: Proceedings of the 8th international conference on Information security applications*, pages 188–202, Berlin, Heidelberg, 2007. Springer-Verlag.

[69] Ye Tian, Ji Wu, Zuoying Wang, and Dajin Lu. Robust noisy speech recognition with adaptive frequency bank selection. In *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, 2002.

[70] T Vira, M Colquhoun, and E Etchells. Reconcilable differences: correcting medication errors at hospital admission and discharge. *Quality and Safety in Health Care*, 15(2):122–126, 2006.

[71] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. *Sphinx-4: A Flexible Open Source Framework for Speech Recognition*, 2004.

[72] Keith S. White. Speech recognition implementation in radiology. *Pediatric Radiology*, 35:841–846, 2005. 10.1007/s00247-005-1511-x.

[73] Jared J. Wolf. Hwim, a natural language speech understander. volume 16, pages 560 –565, dec. 1977.

[74] S.J. Young, N.H. Russell, and J.H.S Thornton. Token passing: a simple conceptual model for connected speech recognition systems. Technical report, 1989.

[75] Steve Young. A review of large-vocabulary continuous-speech. *Signal Processing Magazine, IEEE*, 13(5):45, September 1996.

[76] Atif Zafar, J Marc Overhage, and Clement J McDonald. Continuous Speech Recognition for Clinicians. *Journal of the American Medical Informatics Association*, 6(3):195–204, 1999.

Appendix A

ACR 1

Ontario — **Ambulance Call Report**

NOT REAL PATIENT DATA

Crew Type: ☐ EMA ☐ PCP ☑ ACP ☐ CCP ☐ FLIGHT ☐ FR
Call Type: ☐ BLS ☐ SR ☐ AED ☑ IV ONLY ☐ ALS ☐ CCT

ADMINISTRATION

Surname: Lafontaine
Given Name: Pat
Mailing Address: 1350 Bank St
City / Town: Ottawa
Province: On
Postal Code: K1B 124

Service Name: Ottawa Paramedic Service

CACC	Call Number / Patient Number	Call Date (YMD)
940	16000003	10/06/21

Vehicle Number	Service	Station	Status	Start (km)	PU (km)	Dest. (km)
4115	725	00	88	494	494.6	495.2

UTM Code / Flight #	Disp. Pri. Cde.	Dispatch	Return	Patients	Seq
	2	4	3	1	1

Hospital Registration Number	Date of Birth (YMD)
	75/02/04

Health Card No.	Ver.
4567891011213	CD

Special Codes:

Warning Systems — To Scene / To Destination: None ☑ (to destination); Lights; Lights & Siren ☑ (to scene)

Pick-up Location or Sending Facility — City / Town — Same as above ☑ — Pick-up Location Code: 1B — Sending Hospital / Landing Site:

CLINICAL INFORMATION

Chief Complaint: SOB — Primary Problem: 21 — Secondary Problem: — Time of Occurrence: 0800

Incident History — P Q R S T — A E I O U — Mechanism of injury / direction of force

35 y/o ♀ c/o of severe SOB since ~ 0800. Pt self-administered Ventolin MDI prior to arrival c̄ ø relief. O/A, pt states she is having an "asthma attack". Pt denies any recent illnesses. Pt denies contact c̄ poss allergen

Traumatic Injury Site/Type: Location / Type / Mechanism

Relevant Past Hx: ☐ Previously Healthy 1 ☐ Cardiac 2 ☐ Respiratory 3 ☐ Stroke/TIA 4 ☐ Hypertension 5 ☐ Seizure 6 ☐ Diabetes 7 ☐ Psychiatric 8 ☑ Other 9 ☐ Cancer

asthma, depression, anxiety

Medications: ☐ None ☐ Nitrates ☐ Digoxin ☐ Insulin ☐ ASA ☐ Ventolin ☐ Oral Diabetic Meds ☐ Lasix ☐ Oral Contraceptive ☐ Not Determined

Ventolin, Advair, Wellbutrin

Allergies: ☐ NKA ☐ ASA ☐ Sulpha ☑ Penicillin ☐ Codeine ☐ Other ☐ Not Determined

Penicillin

Treatment prior to arrival: ☐ None ☐ Fire ☐ Police ☐ E.F.R.T. ☐ Other Ambulance ☐ Bystander ☑ Other

Ventolin MDI by patient

☐ Further Clinical Information, see procedures

Cardiac Arrest Information — Arrest Witnessed By: ☐ EMS Personnel ☐ Bystander ☐ Unwitnessed — ☐ DNR ☐ Fire/Police ☐ Family Member ☐ Bystander — CPR Started By: ☐ P.C.P. ☐ A.C.P./C.C.P. ☐ Other Medical ☐ Other — Duration of CPR Prior to arrival of transporting crew: —

Age	Gender	Weight (kg)	C.T.A.S.
35	F	90	☐ 1 - Resus ☑ 2 - Emerg. ☐ 3 - Urgent ☐ 4 - Less Urg ☐ 5 - Non-Urg.

DECEASED: ☐ Pronounced Dead By BHP ☐ Obviously Dead

Skin (Initial Assessment) — Colour: ☐ Flush ☑ Normal ☐ Pale ☐ Jaundiced ☐ Cyanosed — Temp: ☐ Hot ☑ Normal ☐ Cool — Cond.: ☐ Dry ☑ Normal ☐ Wet

PHYSICAL EXAM

General Appearance pt found lying on couch. pt conscious & extremelly lethargic. pt in severe distress

Head/Neck Trachea ☑ - Midline Shifted ☐ - R ☐ - L JVD ☑ - Elevated ☐ - Not Elevated

airway patent, ø subcutaneous emphysema, ø oral edema

Chest Air Entry: ☑ = Bilaterally Decreased ☐ - R ☐ - L Breath Sounds: ☐ - Clear ☐ - Wheezes ☐ - Crackles ☐ - Rub ☑ - Absent

patient in severe respiratory distress, 2-word dyspnea. ø subcutaneous emphysema. Breath sounds absent bilat

Abdomen ☑ - Soft ☐ - Rigid ☐ - Distended ☐ - Tender ☐ - Mass ☐ - Pulsatile ☐ - RU ☐ - LU ☐ - LL ☐ - RL ☐ - Center

soft, non-tender, ø pulsatile mass

Back/Pelvis ☑ - Unremarkable

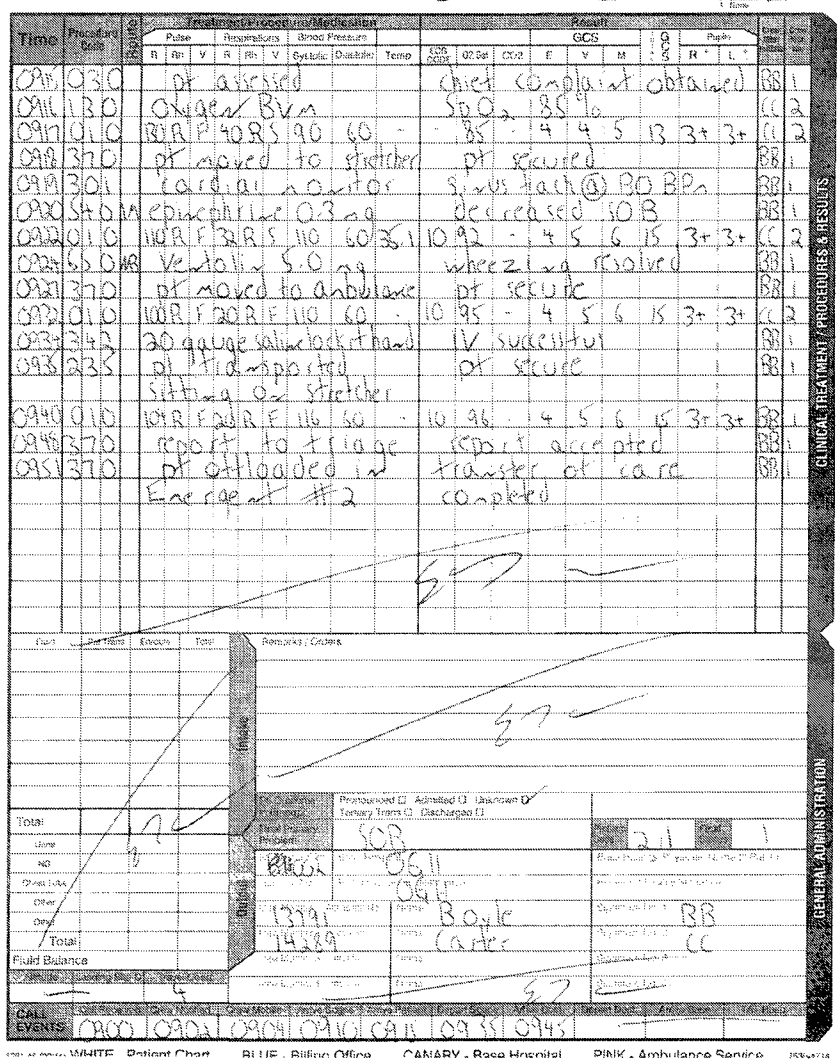
Extremities ☑ - Unremarkable Peripheral Edema ☑ - Absent ☐ - Present Pedal Pulse ☐ - Absent ☐ - Present

☐ Further Physical Findings, see procedures

7530-4714

Call Number if more than 1 page | Page of

NOT REAL PATIENT DATA



Appendix B

ACR 2

NOT REAL PATIENT DATA

Ambulance Call Report

ADMINISTRATION

Surname

Given Name

Mailing Address

City / Town

Province — Postal Code

Hospital Registration Number — Date of Birth (YMD) / /

Health Card No. — Ver.

Crew Type: ☐ EMA ☐ PCP ☑ ACP ☐ CCP ☐ FLIGHT ☐ FR

Call Type: ☐ BLS ☐ SR ☐ AED ☑ IV ONLY ☐ ALS ☐ CCT

Service Name: Ottawa Paramedic Service

CACC	Call Number / Patient Number	Call Date (YMD)
940	16000002	10 / 07 / 04

Vehicle Number	Service	Station	Status	Start (km)	PU (km)	Dest. (km)
4194	725	008	88	4102	4104	4110

UTM Code / Flight #	Dsp. Pri. Cde	Dispatch	Return	Patients	Seq
	07	4	4	1	1

Special Codes

Warning Systems: To Scene / To Destination — ☐ None ☐ · ☐ Lights ☐ · ☑ Lights & Siren ☑

Pickup Location or Sending Facility: Elgin & Catherine

City / Town — Same as above ☐ — Pickup Location Code: 18 — Sending Hospital: — — Landing Site: —

CLINICAL INFORMATION

Chief Complaint: Multi system trauma — Primary Problem: 42 — Secondary Problem: 65 66 — Time of Occurrence: 1405

Incident History — PORST AEIOU — Mechanism of injury / direction of force

~ 20 y/o ♂ hit by vehicle travelling ~ 60 km/hr while crossing street. Pt thrown ~ 30 ft. Pt unconscious for ~ 5 min prior to arrival as per bystander. O/A pt unconscious c multisystem trauma

Traumatic Injury Site/Type

	Location	Type	Mechanism
1	10	36	61
2	16	41	61
3	32	38	61

Relevant Past Hx: ☐ Previously Healthy — 1 ☐ Cardiac 2 ☐ Respiratory 3 ☐ Stroke/TIA 4 ☐ Hypertension 5 ☐ Seizure 6 ☐ Diabetes 7 ☐ Psychiatric 8 ☐ Other 9 ☐ Cancer

Unknown

Medications: ☐ None ☐ Nitrates ☐ Digoxin ☐ Insulin ☐ ASA ☐ Ventolin ☐ Oral Diabetic Meds ☐ Lasix ☐ Oral Contraceptive ☑ Not Determined

Allergies: ☐ NKA ☐ ASA ☐ Sulpha ☐ Penicillin ☐ Codeine ☐ Other ☑ Not Determined

Treatment prior to arrival: ☑ None ☐ Fire ☐ Police ☐ F.F.R.T. ☐ Other Ambulance ☐ Bystander ☐ Other

☐ Further Clinical Information, see procedures

Cardiac Arrest Information — Arrest Witnessed By: ☐ EMS Personnel ☐ Bystander ☐ Unwitnessed — ☐ DNR ☐ Fire/Police ☐ Family Member ☐ Bystander — CPR Started By: ☐ P.C.P. ☐ A.C.P./C.C.P. ☐ Other Medical ☐ Other — Duration of CPR prior to arrival: — — Est. Time of Arrest: — — Time of first shock delivered: —

PHYSICAL EXAM

Age	Gender	Weight (KG)
~ 20	M	80

C.T.A.S.: ☐ 1 - Resus. ☑ 2 - Emerg. ☐ 3 - Urgent ☐ 4 - Less Urg. ☐ 5 - Non-Urg.

DECEASED: ☐ Pronounced Dead By BHP ☐ Obviously Dead

Skin (Initial Assessment) — Colour: ☐ Flush ☑ Normal ☐ Pale ☐ Jaundiced ☐ Cyanosed — Temp: ☐ Hot ☑ Normal ☐ Cool — Cond.: ☐ Dry ☑ Normal ☐ Wet

General Appearance pt found lying prone on street, pt unconscious, pt in severe distress

Head/Neck Trachea ☑ - Midline Shifted ☐ - R ☐ - L JVD ☐ - Elevated ☑ Not Elevated Ø subcutaneous emphysema airway patent. gag reflex intact. hematoma noted to left parietal skull. pupils unequal. Ø instability on palp C-spine stable

Chest Air Entry: ☑ - Bilaterally Decreased ☐ - R ☐ - L Breath Sounds: ☑ - Clear ☐ - Wheezes ☐ - Crackles ☐ - Rub ☐ - Absent

spontaneous respirations noted chest wall stable. Ø subcutaneous emphysema

Abdomen ☐ - Soft ☐ - Rigid ☑ - Distended ☑ - Tender ☐ - Mass ☐ - Pulsatile ☐ - RU ☐ - LU ☐ - LL ☐ - RL ☐ - Center

abdomen distended & tender on palp. Ø pulsatile mass

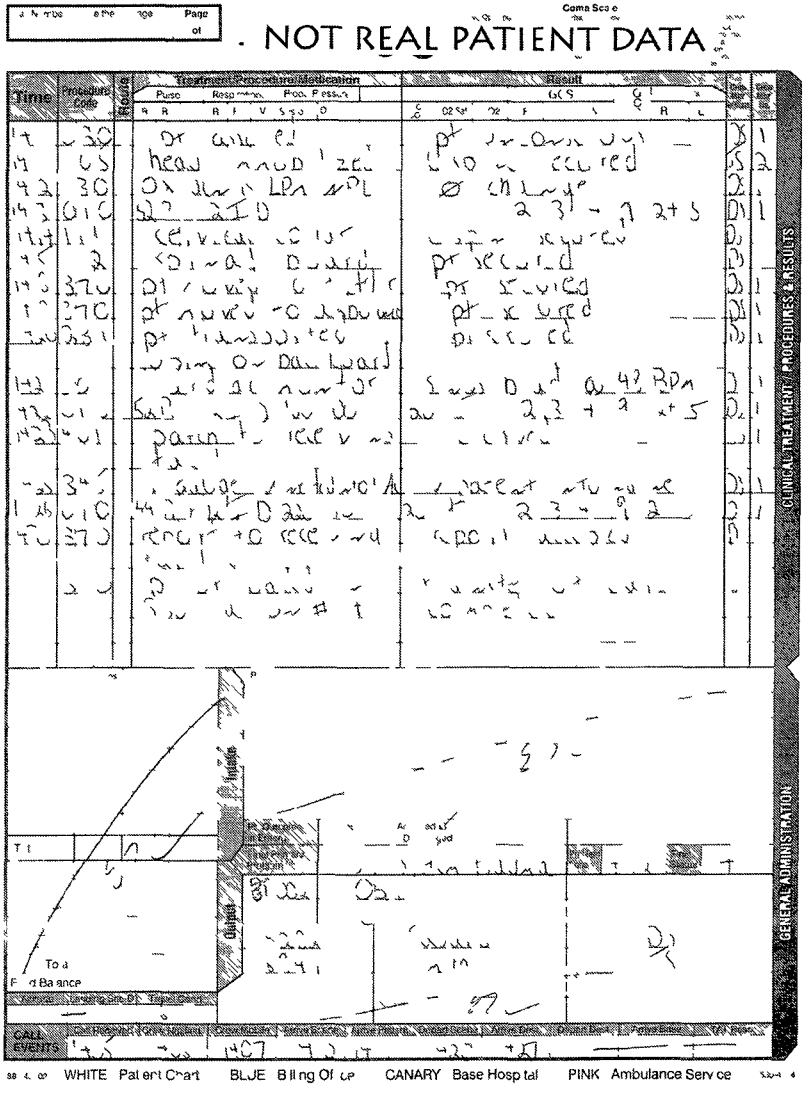
Back/Pelvis ☑ - Unremarkable pelvis unstable on palp

Extremities ☐ - Unremarkable Peripheral Edema ☑ - Absent ☐ - Present Pedal Pulse ☐ - Absent ☑ - Present

rt upper leg deformity, leg shortened & externally rotated

☐ Further Physical Findings, see procedures

1881-45 (02/11) — 7530-4714


NOT REAL PATIENT DATA
Time
Procedure Code
Route
Treatment, Procedure, Medication
Result
CLINICAL TREATMENT • PROCEDURES • RESULTS
GENERAL ADMINISTRATION
CALL EVENTS
WHITE Patient Chart
BLUE Billing Office
CANARY Base Hospital
PINK Ambulance Service