


Computer Science
Johns Hopkins University
3400 North Charles Street
Baltimore MD 21218

This project fulfills a requirement for the

degree.

Advisor's signature: _____

Date: Nov 27 2023

Advisor's Name:

Cloud Based Medical Systems For Automatic Filling of Patient Forms

Jonathan Young
Johns Hopkins University Computer Science Department
jyoun127@jhu.edu

Abstract—Cloud technologies are poised to revolutionize various industries heavily dependent on software solutions. The healthcare industry is no different, with vast quantities of patient data stored and processed through smartphone apps and servers. This project aims to explore and test the feasibility of automating the process of filling out medical forms based on a conversation between a doctor and patient through the Cloud. This report first details how sample conversations between the doctor and patient are generated. Next, the general architecture of the pipeline is detailed. Lastly, the results of the performance of the pipeline are displayed and analyzed, alongside detailing the limitations of this cloud pipeline.

I. INTRODUCTION

Cloud computing services, from AWS to Azure, have radically altered the landscape of software applications. Traditionally, companies develop and maintain their own in-house architecture and software. With cloud, companies can rely on 3rd party services for reliable software and hardware while also reducing costs and overhead. This model is known as software as a service [1].

One major industry that can benefit from cloud development is the healthcare industry. Much of the patient data is already stored in cloud services and with greater demand for telehealth and reliance on novel technologies such as machine learning, medical applications are poised to benefit from emerging cloud services [2].

This project demonstrates that more complex medical tasks can be automated utilizing cloud services. Specifically, this project aims to fill out medical forms based on a conversation between doctor and patient. There are two overarching steps for this project. The first step is developing a method to create mock conversations between patients and their doctors as actual doctor patient transcripts are not readily available due to privacy concerns. The second step is to create a cloud pipeline that processes an audio recording between a patient and doctor's conversation quickly and writes the information to medical forms accurately.

II. TEST DATA

Due to the sensitive nature of conversations between patients and their doctors, it is almost impossible to find large quantities

of real transcripts. However, one way to simulate these interactions is through using existing generative language models such as ChatGPT to simulate these conversations. Collection of various ChatGPT outputs based on multiple prompts allows for the testing of various patients' circumstances. For instance, asking ChatGPT to "generate 10 examples of an interaction between a doctor and patient, where the patient takes medication and has a chronic illness" results in 10 conversations that can measure the pipeline's strengths and weaknesses over a multitude of indicators such as the recognition of patient personal data, medical terminology, etc. For the purposes of this project, 10 different medical scenarios with different dialogues, were generated for testing. Results of this testing with these phrases are shown later. A sample of a ChatGPT generated conversation to test allergy and medicine recognition can be found below:

“Doctor: It's crucial to be mindful of those reactions. Allergies can range in severity, and it's important to monitor any changes in symptoms. Have you taken any over-the-counter medications or tried any home remedies to alleviate your allergy symptoms?”

Patient: Yes, I've been taking antihistamines during allergy seasons, and they provide some relief. I've also tried using nasal sprays and saline rinses to alleviate congestion.”

III. ARCHITECTURE

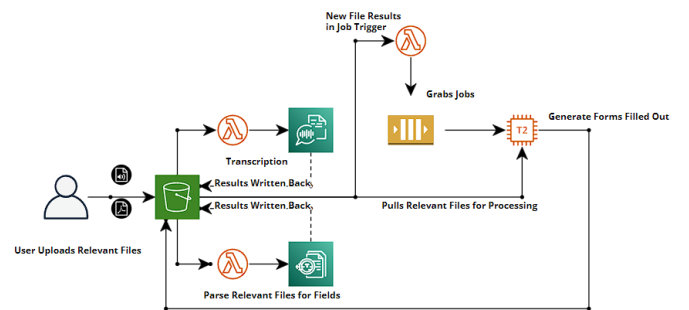


Fig 1 – The Architecture

The pipeline of this AWS system is entirely reliant on a series of services that process data, services that store that data, and services that connect dependent services with each other.

A. AWS Transcribe

Processing

The starting point for this pipeline starts with AWS transcribe, a transcription service. For this project, transcribe is configured for medical transcriptions, where there is an emphasis on precision of recognizing speech. To utilize this service, the user calls a python script to upload the audio file to S3, the storage service. Once the S3 bucket has received the audio file, a lambda function is automatically triggered that feeds the file from s3 to AWS transcribe. The predefined language is English, and the results are stored in the same folder as the uploaded file in S3. Although in this project this component is not used, as audio recordings aren't available, a deployment version of this pipeline would include this step.

B. AWS Textract

AWS Textract is a service to recognize fields and elements within a file that contains forms or text, such as an image or pdf. There are two utilizations of AWS Textract in this project.

During the same time as the transcription job mentioned in the prior step, a medical form file is uploaded to S3 from the user to begin recognizing its fields. Here, we mean fields to be fillable entries on the medical form such as name, date, etc. This is done through Textract's form extraction API. The API returns information about fields and their locations on a given form.

The second task is to use Textract to make queries of a medical table generated by a system, as described in this paper's *Table Generation* section. Textract contains a built in NLP engine that one can use to query information from a document.

C. AWS Medical Comprehend NLP Engine

The next step in this process is to run the transcripts, generated from part A, through AWS Medical Comprehend, an NLP engine specifically trained for medical contexts. Medical Comprehend identifies key medical entities and information from the transcript. To do this, a lambda function is triggered due to the transcription file being generated from AWS Transcribe, which subsequently starts a Medical Comprehend job to begin recognizing entities. The results are also written back to S3.

Once the results of this job are finished, a sequential lambda function recognizes that the medical comprehension service has completed and deposits the location of this result, the location of the Textract result, and the location of the requested forms to be filled out in S3 onto the SQS queue. Note that this lambda function checks if all the required files are present. It will not deposit the aforementioned information onto the queue if the files have not yet been completed.

D. EC2

The next step involves EC2. EC2, at its basic core, is a virtual machine. This virtual machine is one of many existing in a cluster. For the purposes of this project, the cluster is limited to 5 EC2 instances but can be scaled based on demand. For the purpose of demonstrating real world use, we created a cluster to handle large loads and mitigate response time.

These EC2 instances continually poll the SQS for jobs and begin processing relevant files for filling out a form. This processing is done via a program called *The Program* in section E and is hosted on an EC2.

E. The Program

The program continuously polls the queue to process data. If there is a job, the poll returns with the file locations of the Medical Comprehend result, Textract Result, and lastly, the form to be filled in S3. The program is now ready to proceed to the next step.

Table Generation

The end goal of this project is to fill out patient data from Medical Comprehend, which contains medical entities, to the form itself. However, there is one roadblock that exists, that is, filling out form variable fields dynamically.

To solve this issue, the program generates a table containing information generated from Medical Comprehend and uses AWS Textract's query capability. Textract's queries will come from the form fields themselves, and the answers will come from the medical table. An example of a generated table with data from medical comprehend can be found below.

MEDICATION				
GENERIC_NAME	FORM	DOSAGE	FREQUENCY	ROUTE_OR_MODE
Albuterol	inhaler	2 puffs	every 4-6 hours as needed	
corticosteroid		2 puffs	twice daily	
Montelukast		10 mg	daily at bedtime	
Fluticasone	spray	2 sprays	daily	nasal, each nostril

Fig 2 – An example of elements containing sub-elements (descriptors)

An example can be found below of a query and result (Fig 3.B) and where it acquired that answer (Fig 3.A)

MEDICATION				
GENERIC_NAME	FORM	DOSAGE	FREQUENCY	ROUTE_OR_MODE
Albuterol	inhaler	2 puffs	every 4-6 hours as needed	
corticosteroid		2 puffs	twice daily	
Montelukast		10 mg	daily at bedtime	
Fluticasone	spray	2 sprays	daily	nasal, each nostril

Fig 3.A – Textract Query Using the Image from Figure 2

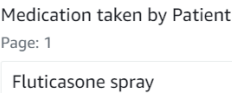


Fig 3.B – Textract Query Using the Image from Figure 2

However, there is a challenge when it comes to generating these tables. Tables often come in various formats, where each layout can significantly influence the querying result from Textract.

The three formats of the tables can be found below, and the results of which table results in the best querying result can be found in the *Results* section.

ANATOMY

ANATOMY	Descriptor
nose	
eyes	
heart	

Fig 3.A – Table in Center with Title of Table to Side

MEDICATION				
GENERIC_NAME	FORM	DOSAGE	FREQUENCY	ROUTE_OR_MODE
Albuterol	inhaler	2 puffs	every 4-6 hours as needed	Inhaled
corticosteroid		2 puffs	twice daily	
Montelukast		10 mg	daily at bedtime	
Fluticasone	spray	2 sprays	daily	nasal, each nostril

Fig 3.B – Table and Title Aligned

=====
ANATOMY
SYSTEM ORGAN SIT : nose

SYSTEM ORGAN SIT : eyes

=====

Fig 3.C – Text Base Tables

Once the program has retrieved results from the Textract query and utilized the results of the Textract operation to locate the fields to be filled out, the program proceeds to fill out the form and writes the result back to S3. The pipeline concludes.

IV. RESULTS

A. Table Type Results

Allergy and Medical Terminology Identification Rates

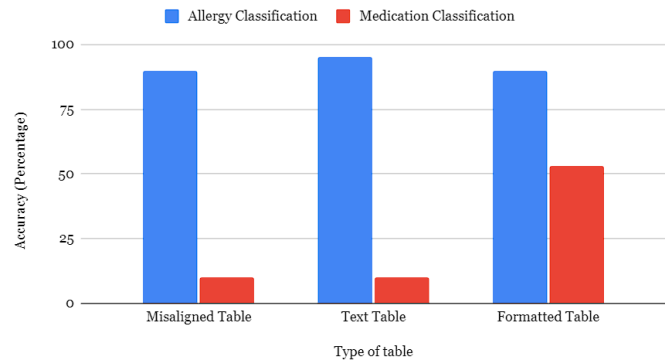


Figure 4 – Results of Various Table Types

In order to determine which table format is the best for Textract queries, we utilized the 10 different conversations to fill out a school sports athletic form. The conversations contain information regarding medication and allergies. To introduce variance, not all conversations had allergy information, but all conversations had medication information. We measured accuracy as to how well the pipeline filled out the correct allergy and medical information in the form.

One noticeable outcome from the graph is that all 3 table types correctly identified allergies. Since there were significantly less allergies than medication information mentioned in the transcripts, this result demonstrates that the pipeline is adept at recognizing when an attribute was absent and abstaining from filling in fields regarding that attribute but is more likely to miss fill data when filling out attributes that are present.

In terms of filling out fields incorrectly with the wrong data the results are shown below. Note the error rate does not include the absence of correct information.

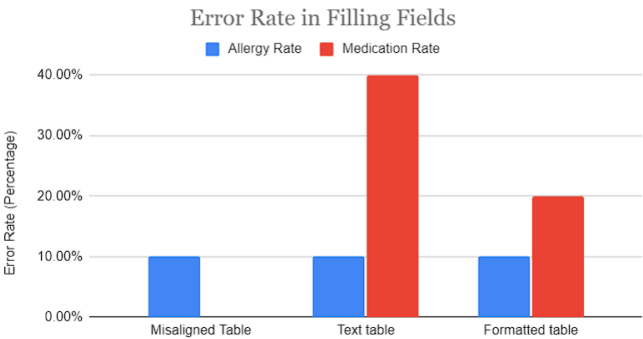


Fig 8 – Misidentification Versus Correct Identification Rate

The figure above highlights that tables that result in fields being correctly filled out may also result in an increased likelihood of them filling out fields incorrectly compared to tables that don't fill out fields at all. This is especially true for the formatted table. It is able to recognize text better than the other two tables and has a higher success rate identifying fields and filling them out in the form but does so with notable errors.

Overall, the graph illustrates that the formatted table, as seen in Figure 3.b, performs the best. Therefore, it is best suited for the pipeline.

B. Limitations

It was found that the program is largely adequate for filling out basic information about the patient, such as their name, but had trouble classifying the different types of information for the same data type. For instance, AWS Textract interpreted date, for signatures, similar to date of birth, even though they are two different fields.

Name: /Ms. John Doe John.Doe Date of birth: January 15, 1980

Fig 5 – Successful Write of Patient Information

Signature of parent or guardian: Date: January 15, 1980

Fig 6 – Misidentification of Date versus Birth date

The program was not adequate, however, for filling out medical terminology. Though it was able to identify the relevant fields for identification, there were multiple mishaps that could have happened. The most pressing mistake is that it was unable to identify the answer or failed to provide the complete answer. An example of this is when the form tested has the field: *Medicines*

and supplements: List all current prescriptions, over-the-counter medicines, and supplements (herbal and nutritional), which was never filled out despite the medication table having an answer for this field. See fig 9.

MEDICATION				
GENERIC_NAME	FORM	DOSAGE	FREQUENCY	ROUTE_OR_MODE
Albuterol	inhaler	2 puffs	every 4-6 hours as needed	Inhaled
corticosteroid		2 puffs	twice daily	
Montelukast		10 mg	daily at bedtime	
Fluticasone	spray	2 sprays	daily	nasal, each nostril

Fig 9 – The Medication Table Having an Answer for This Field

Another type of mistake is filling out the wrong fields. See Fig 10.

EXAMINATION									
Height:				Weight:					
BP:	/		(/)	Pulse:		Vision: R 20/	L 20/
								Corrected:	42 12845678

Fig 10 – A Misclassification of a Field

C. Examples

Below are some examples of a successful filling out of medical forms.

SHARED EMERGENCY INFORMATION

Allergies:

penicillin shellfish

Medications:

Fig 11 – Allergies Example

List past and current medical conditions.

asthma, labored breathing, wheezing

Have you ever had surgery? If yes, list all past surgical procedures.

Fig 12 – Medical Conditions Example

Do you have any allergies? If yes, please list all your allergies (ie, medicines, pollens, food, stinging insects).

seasonal allergies

Fig 13 – Another Allergies Field

D. Conclusion

Despite the pipeline’s shortcomings, the pipeline does indeed demonstrate the feasibility of automating complex medical administrative tasks using cloud services. In terms of real-world use, the pipeline is adequate enough to give users a head start in filling out forms with some existing answers to fields, but not necessarily for completely filling out the form itself due to a noticeable error rate.

REFERENCES

[1] “What are the benefits of cloud computing?,” IBM, <https://www.ibm.com/topics/cloud-computing-benefits> (accessed Aug. 28, 2023).

[2] M. Masud, G. S. Gaba, K. Choudhary, R. Alroobaea, and M. S. Hossain, “A robust and lightweight secure access scheme for cloud based E-healthcare services,” *Peer-to-Peer Networking and Applications*, May 2021, doi: <https://doi.org/10.1007/s12083-021-01162-x>.