

Voice assistants in hospital triage operations

Montali Simone

Dipartimento di Ingegneria e Architettura
Università di Parma
Parma, Italy
simone.montali1@studenti.unipr.it

Mordonini Monica

Dipartimento di Ingegneria e Architettura
Università di Parma
Parma, Italy
monica.mordonini@unipr.it

Tomaiuolo Michele

Dipartimento di Ingegneria e Architettura
Università di Parma
Parma, Italy
michele.tomaiuolo@unipr.it

Abstract—This paper analyzes the creation and usage of a voice assistant for the triage of emergency room patients. The project strongly relies on Mycroft, an extensible open source voice assistant. The patients are able to declare their symptoms to the computer, which recognizes the urgency and acts accordingly. The software can even provide useful medical informations to the users.

Index Terms—voice assistant, mycroft, hospital, er, emergency, triage

I. INTRODUCTION

While the medical research progressed heavily in the last years, the IT infrastructures involved didn't. This was probably caused by the need of having ultra-reliable software, since no errors are permitted in ER rooms. The use cases of voice assistants in hospitals are numerous. This paper inspects the automation of the triage procedures, usually done by nurses, to help patients faster and more efficiently. The voice assistant is an agent able to connect to Mycroft's cloud services, recognize the patients' symptoms and assign them a priority. It also features a second classifier, based on a deep learning network, trained on a dataset composed of ER rooms interviews. This second classifier is used every time the first one isn't capable of recognizing the symptoms. The agent is even capable of interacting with the italian health ministry's website to retrieve the informations requested by the users. The triage operations produce reports in a standard format, making the integration with existing systems easy, cheap and fast. This article is composed as follows:

- Section II introduces this project's context
- Section III presents the Mycroft project and its peculiarities
- Section IV describes the creation of the first module of this project
- Section V illustrates the workings of the second classifier
- Section VI describes the retrieval and conveying of medical informations and tips
- Section VII describes this project's applicability and requirements

II. VOICE ASSISTANTS AND TRIAGE

A. Voice Assistants

Voice assistants are software capable of interpreting human dialogs and reply through synthesized voices. The most famous examples are Amazon Alexa, Apple Siri, Google Assistant... The assistant waits for a *wake word*, hears the request, transforms it into text, interprets it and *speaks* a reply through a TTS engine. Voice assistants are rarely used in hospitals, but a report from **Voicebot** [1], an organization supporting the spread of these technologies, stated that while just 7.5% of the interviewed people had used a voice assistant in healthcare, 52% would like to do so. Surprisingly, this includes even the higher age groups.

B. Triage

Triage is the process in which the patients requesting help in the ER are catalogued by their urgency and their priority. The term was born during wars and maxi-emergencies, but similar operations are done everyday, in hospitals all around the world. The most famous *methods* are the **START** and the **CESIRA**. The italian health ministry never defined a standard, but generally the patients get divided in 4 groups: red(critical, immediate access), yellow (life threatening, access within 15 minutes), green(access within 60-90 minutes), white. The triage produces a report containing the patient's anagraphic data, the symptoms, the urgency, date and time of access to the hospital.

C. Triage and COVID19

The COVID19 global pandemic changed the triage procedures heavily: the key is keeping patients with COVID19-compatible symptoms far from the other ones. Usually, this involves the building of *pre-triage* tents, where patients get tested. Integrating a fast COVID19 diagnostic in the voice assistant could reduce the risk of infections.

III. MYCROFT

A. Why Mycroft?

The most popular voice assistants, like Google Assistant or Amazon Alexa, are proprietary software made by corporations. While this may be good for the usability and functionality, it lacks a critical feature: privacy. Since health data is tremendously precious, we need to protect it, and doing that while using a proprietary voice assistant is kind of impossible. Choosing an open source alternative allows us to guarantee **data privacy, extensibility, customizability**, and last but not least, **a free tool that every hospital could use**.

B. The Mycroft project

Mycroft was born to provide an open source voice assistant to users, developers, researchers. [3] It was initially funded with a crowdfunding in 2015, and obtained great success. The project is based on a subset of tools built by Mycroft, Inc and other companies:

- Precise, a wake word listener based on a *Gated Recurring Unit*, a recurrent neural network similar to an LSTM with a forget gate [4]. It is trained over audio tracks, not text, making the software compatible with different accents, dialects, languages.
- The Speech To Text component isn't enough stable yet. Because of this, this project uses the Google STT engine [5]. The requests are bucketed and proxied through Mycroft's servers to provide a privacy layer.
- The intent interpretation is based on two tools: **Padatious** and **Adapt**. While Adapt is based on keywords recognition, Padatious is based on a neural network trained over full phrases. Padatious can even extract entities from the intent, like a city name or a disease.
- While Mycroft built a Text To Speech engine, **Mimic**, it isn't currently available in Italian. Because of this, we're using the Google STT engine, with proxied requests.
- A graphical user interface was built over the Qt library using KDE Plasmoids, graphical components available in the KDE Plasma desktop environment.

IV. SKILL CREATION

The greatest power Mycroft has to offer is its modularity: every responsibility is tied to a single module, named **skill**, which interprets the related requests and provides an answer. Some examples might be a joke-telling skill, a reminder skill, a timer skill. When Mycroft receives a request, it checks all of the skills' intents for matches. If a match is found, the related function is called. Skills are coded in Python, and have a standard structure. To generate the boilerplate code for a new skill, Mycroft provides a command line interface.

A. Skills file structure

Skills have a standard file structure and file types:

- `dialog` files: these files contain the assistants's answer dialogs. Multiple phrases are included and randomly chosen: this makes the assistant more *human*.

- `intent` files: these files are used to train the Padatious' neural network. They contain example phrases for every intent.
- `voc` files: these files contain Adapt keywords. Since this project strictly relies on Padatious, these are almost not used.
- `entity` files: these files contain entities to be extracted by padatious, like city names, diseases...
- `settingsmeta.yaml`: this YAML file defines the skill's settings, to be edited on the Mycroft Home website.
- `__init__.py`: this file contains the skill's main code. It is based on an extension of the `MycroftSkill` class, which contains the classes and decorators needed to define a new skill.

B. hospital-triage-skill

The skill is structured as follows: every symptom type is caught in an intent. Basing on the symptom type, different things happen: a urgency code is assigned to the patient (red, yellow, green), anagraphical data is collected, minor symptoms are checked... All the collected data gets saved to a Python dictionary to be exported in JSON. All the *standard* operations, like asking for a pain evaluation or the patient's name, are handled by decorators.

C. The covid_symptom decorator

The COVID19 global pandemic strongly changed how people behave, interact, get cured. As already discussed in section II-C, hospitals need to keep COVID19-compatible patients separate from the other ones, and from each other. Automating the COVID19 diagnostic may be crucial now. Therefore, the voice assistant needs to check if the patient may be infected and potentially contagious. Because of this, all of the symptoms that are slightly connected with COVID19 trigger a diagnostic handled by the `covid_symptom` decorator. This asks the patient for known symptoms, and multiplies a score (by default equal to 1) by their *multiplier*:

- **Fever**: multiplied by 2.0
- **Sore throat**: multiplied by 1.3
- **Cold**: multiplied by 1.3
- **Breathing fatigue**: multiplied by 1.6
- **Cough**: multiplied by 1.6
- **Contact with infected people**: multiplied by 3.0
- **Missing sense of taste**: multiplied by 1.7

Every suspected patient starts with a score of 1, and if the final score reaches a threshold (by default equal to 15, but can be adjusted basing on the pandemic's recession) the patient is advised to stay where he is and wait for a doctor to visit him.

D. Helpers

The skill needed some helpers to cope with challenges the Italian language implies. An example might be the temperature extraction from an utterance: some inputs may be *"trentasette e mezzo"*, *"trentasette punto cinque"*, *"trentasette virgola cinque"*, *"trentasette cinque"*, **all meaning the same thing**. Another example is the number validation: the bot may interpret a *"sei"* as the verb *essere* instead of a number.

E. The exported data

All of the collected and inferred data gets exported to a JSON file. Using a standard like JSON makes integrating the bot with existing systems easy: the hospitals don't need to entirely change their informative systems, they just need to integrate some small components. An example of triage report of a COVID19 infected patient may be the following:

```
{
  "can_talk": "yes",
  "main_symptom": "breathing",
  "code": "red",
  "fever": 38.7,
  "has_sore_throat": "yes",
  "has_cold": "no",
  "has_breathing_difficulties": "yes",
  "has_cough": "yes",
  "has_had_contacts": "yes",
  "misses_taste": "yes",
  "covid_score": 22.6304,
  "symptom_declaration": "ho il fiatone",
  "age": 21,
  "other_symptoms": "ho mal di testa",
  "pain_index": "8",
  "full_name": "marco rossi"
}
```

F. Conversation example

A conversation with the bot may be the following:

- **User:** Can you help me?
- **Voice assistant:** Are you the one seeking medical treatment?
- **User:** Yup.
- **Voice assistant:** What is your main symptom?
- **User:** I have a fever.
- **Voice assistant:** Ok, a fever. Have you measured it recently?
- **User:** Yes.
- **Voice assistant:** Please, tell me your temperature.
- **User:** Thirtyseven point eight.
- **Voice assistant:** I need some additional informations for the diagnosis. I will now ask you some yes/no questions. Do you have a sore throat?
- **User:** Yes.
- **Voice assistant:** Do you have a cold?
- **User:** Yes.
- **Voice assistant:** Do you have any trouble breathing?
- **User:** No.
- **Voice assistant:** Do you have a cough?
- **User:** Yes.
- **Voice assistant:** Have you been in contact with people infected by the new coronavirus?
- **User:** Yes.
- **Voice assistant:** We're notifying a doctor. It is possible that you may be infected by COVID19. Could you tell me how old you are?

- **User:** Twentytwo.
- **Voice assistant:** How would you define your discomfort from 1 to 10?
- **User:** Eight.
- **Voice assistant:** If there's any other symptoms I need to know, please tell me
- **User:** Nothing else.
- **Voice assistant:** Could you please tell me your full name?
- **User:** Simone Montali.
- **Voice assistant:** Simone Montali, is that right?
- **User:** Yes.
- **Voice assistant:** Thanks for your patience, go at the RED desk.

V. FALLBACK SKILLS AND NLP

A problem now arises: what if the patient has a disease which was not covered in the previous symptoms? The intent would not be caught. Sadly, diseases are so vast and numerous that it would be impossible to hardcode them: we need a *statistical* approach. The target is now building a text classifier that is able to classify the patient's declaration to a symptomatology or at least an hospital ward.

A. Dataset

Since the medical data is fortunately well protected by privacy laws all around the world, finding medical datasets is not easy. However, creating a dataset from scratch is still possible: this is exactly what **Appen**, an australian start up, did. The dataset used in this project [7] contains audio snippets, transcriptions and symptomatology of patients requesting help to nurses. Since the Mycroft's STT engine handles the conversion to text, we only need to train a classifier that maps patient requests to symptom types. An example of this dataset's entries may be:

- **Phrase:** My son had his lip pierced and it is swollen and the skin inside on his lip is grey and looks infected.
- **Prompt:** Infected wound

B. Dataset translation

Since the voice assistant was mainly born for italian users, we need to translate the dataset to italian. Using Google Translate's APIs would be easy but costly. We could use a scraper to query Google Translate's webpage, but that would be slow. Fortunately, third party Python modules have been made: `googletrans` is the most famous. Using it, we were able to translate the dataset with a simple script.

```
for index, row in data.iterrows():
    phrase = translator.translate(
        row["phrase"],
        dest="it").text
    prompt = translator.translate(
        row["prompt"],
        dest="it").text
    if translator.detect(phrase).lang == "it"
        and translator.detect(prompt).lang == "it":
        translated_data.at[i, "phrase"] = phrase
```

```
translated_data.at[i, "prompt"] = pr
i = i+1
```

When Google Translate can't recognize a phrase's meaning, it skips it. The `translator` object, besides translating things, can even detect languages. This allows us to do a little workaround: if the detected language isn't italian in both the phrase and the symptomatology, the entry is skipped. This reduced the dataset from 8000 entries to almost 2000.

C. Classifier training

Having an italian dataset, we can now proceed with the classifier's training. To do so, using the **fastai** library simplifies things: it provides the `text.learner` class, which is able to train a model and export it. This enables us to train the model a single time, and simply import it into the voice assistant.

D. Data preparation and training

We can now import our dataset in a Jupyter notebook, check for NaNs, and create the `text.learner` with **fastai**. For the project, we used the AWD-LSTM structure, a recurrent neural network which has really high performance on text processing. This type of network has a **Long Short Term Memory** [10], which enables the training to find patterns in distant parts of the text, **Average SGD** and **Dropconnect**, a dropout that drops weights instead of activations [9]. We can now train the network and check the confusion matrix:

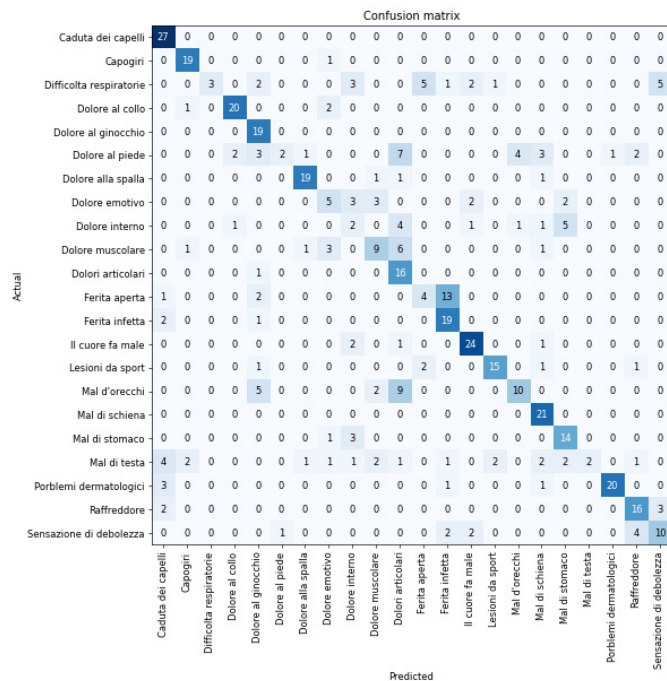


Fig. 1. Confusion Matrix

While it isn't perfect, the diagonal in the matrix shows that **the model was trained correctly**. Talking about accuracy, the model reached **more than 60%**:

epoch	train_loss	valid_loss	accuracy	time
0	3.092835	3.079816	0.051282	01:43
1	2.957276	2.627386	0.198718	01:42
2	2.600713	1.969201	0.457265	01:32
3	2.192694	1.707950	0.598291	01:38
4	1.923900	1.633354	0.632479	01:35

Fig. 2. Accuracy

We can now easily export it and integrate it in the voice assistant by a simple import.

E. Fallback skills creation

As explained earlier, Mycroft works by recognizing intents in the user's requests. These were defined in the last skill, but the symptomatology is more than the ones coded. However, Mycroft makes it possible to define **fallback skills**, a skill that gets triggered every time the intent is not recognized. By creating one, we're able to get the patient's utterance and analyze it with our classifier. Fallback skills are ordered by a priority, but since our voice assistant will have just one, it doesn't matter. After registering the fallback skill, we can import the model. The symptomatology inserted in the model are defined in a separate `classes.json` file, containing each symptom's urgency, GUI emoji, name, and its compatibility with COVID19. An example would be:

```
{
  "name": "Dolore al collo",
  "emoji": "[removed]",
  "code": "yellow",
  "covid": false
}
```

Having done that, the symptoms can now be handled like any other one. Ideally, we'd like to check if the classifier worked right by asking the patient. An example of conversation may be:

- **User:** *I need help.*
- **Voice assistant:** *Are you the one seeking medical treatment?*
- **User:** *Yes.*
- **Voice assistant:** *What is your main symptom?*
- **User:** *My shoulder dislocated while I was playing golf.*
- **Voice assistant:** *So, we're talking about shoulder pains, right?*
- **User:** *Yes.*
- **Voice assistant:** *Could you tell me how old you are?*
- **User:** *Twenty.*
- **Voice assistant:** *How would you define your pain from 1 to 10?*
- **User:** *Nine.*
- **Voice assistant:** *If there's any other symptoms I need to know, please tell me*

- **User:** *Nothing else.*
- **Voice assistant:** *Could you please tell me your full name?*
- **User:** *Mario Rossi.*
- **Voice assistant:** *Mario Rossi, is that right?*
- **User:** *Yes.*
- **Voice assistant:** *Thanks for your patience, go at the YELLOW desk.*

VI. INFORMATION RETRIEVAL

This tool's use cases might not be limited to triage and helping in emergencies. In fact, a patient could just want to know some medical informations about a particular disease without having to talk to a doctor.

A. Data source

It is obvious that the data source we're looking for has to be authoritative: Wikipedia is not reliable enough. Luckily, the Italian health ministry published a medical encyclopedia on their website. These informations are complete, reliable and interesting. To be able to use them, we can scrape the website using tools like **Selenium**. Selenium is a framework used to test web applications; it provides a simple interface to automate a web browser [11]. We could use two approaches for the data collection: an API approach, which downloads the informations basing on queries, or a periodic one. The first one is possible thanks to the Python `http.server` library, in which we can define a handler for GET requests. This will look for the query on the encyclopedia, download the page, insert the data into a JSON and return it. Since the informations on the website rarely vary, we can download them periodically and keep them saved. By defining a script that iterates over every item in the encyclopedia, we can download all the needed informations. Using **cron**, a job scheduler for Linux, it's possible to automatically download the data periodically.

B. Integration of the data into the voice assistant

Having the data, we can now define a new intent in the voice assistant to request informations. Padatious' entity extraction simplifies this: we can train it with phrases like *"Tell me something about {disease}"*, creating a *disease* entity containing all the known diseases. These are not strictly exact matches: we're looking for the best match, so that if the user asks something about *tumor in the lung*, *lung tumor* is matched. After matching the name, the assistant checks the known data, like *prevention*, *cure*, *therapy*, *symptoms*, and asks the patient to choose. An example of conversation might be the following:

- **User:** *What do you know about celiac disease?*
- **Voice assistant:** *The best match is celiac condition, is that what you're looking for?*
- **User:** *Yes.*
- **Voice assistant:** *What would you like to know? Choose between Description, Symptoms, Causes, Therapy, Complications*
- **User:** *Tell me the causes.*
- **Voice assistant:** *This is what I know: celiac disease is a multifactorial condition...*

VII. APPLICABILITY AND TECHNICAL REQUIREMENTS

Since this voice assistant could enhance the hospitals' workflow all around the world, it has built to be as cheap and easy as possible. The TTS and STT engines are computed in cloud: this removes the most computationally heavy module from our responsibility. The intent parser, Padatious, needs an ultra light training just once every start of the bot. Even if the computer is not powerful, it requires just a few minutes. The fastai training is computed in cloud: the model gets exported and used without any further training. This doesn't require any computational power. The only real requirement is defined by the GUI: since it is built on KDE Plasmoids, the desktop environment has to be KDE Plasma. Mycroft recommends using KDE Neon, a Ubuntu LTS distribution with KDE Plasma installed. The devices that support this project are, therefore, almost infinite: even a simple Raspberry Pi or an old computer can run Mycroft.

VIII. CONCLUSIONS

The global pandemic the world had to bear with has highlighted how hospitals, in the last years, have been forgotten by politicians, people, nations. In the last 20 years, in Italy, the number of institutes diminished from 311 thousands to 191 thousands, while the public spending was raised from 60 miliardi to 112 [12]. With the aid of computers, health workers could get some relief and concentrate on what matters more: curing people. Voice assistants could *flatten the digital divide*, being easy to use for everyone, including the elder. Translation of the assistant to multiple languages could be an incredible tool for immigrants, tourists and travelers who can't speak the national language. Right now, the most urgent improvement to make is finding or creating a better dataset: if the classifier worked better, we could skip the hardcoding of the most common symptoms and just leave one skill. The ultra low cost of this project's requirements makes it possible for every hospital in the world to install a tool like this. The open source design (all the source code is licensed with a GPL-3.0 license) guarantees freedom, privacy, expansibility, customizability, and, last but not least, the possibility of saving lives without having to pay a price.

IX. SOURCE CODE

Every single line of code of this project is open source and available on GitHub:

- Mycroft skill for the triage operations and the information requests: `montali/hospital-triage-skill`
- Fallback skill for the NLP classification and diagnosis of symptoms: `montali/hospital-fallback-skill`
- Health Ministry's encyclopedia scraper: `montali/medical-infos-api`

REFERENCES

- [1] Voicebot - Orbita, "Voice Assistant Consumer Adoption in Healthcare", In: (ott. 2019).
- [2] Ministero Della Salute, "Linee di indirizzo nazionali sul triage inferospedaliero",

- [3] Rebecca Harrington, "Customize An Open-Source HAL For Your Home", 2016
- [4] Felix Gers; Jürgen Schmidhuber; Fred Cummins (1999). "Learning to Forget: Continual Prediction with LSTM". Proc. ICANN'99, IEE, London. 1999: 850–855
- [5] Google, Speech-To-Text
- [6] Qt, Official website
- [7] Paul Mooney - Appen, "Medical Speech, Transcription, and Intent", 2019.
- [8] fastai, Official docs
- [9] Stephen Merity; Nitish Shirish Keskar; Richard Socher, Regularizing and Optimizing LSTM Language Models, 2017
- [10] Sepp Hochreiter e Jürgen Schmidhuber, "Long short-term memory", In: Neural computation 9,8 (1997), pp. 1735–1780.
- [11] Selenium, Official repository
- [12] Agenzia Giornalistica Italia, "La riduzione di ospedali e posti letto negli ultimi 10 anni", In: (mar. 2020)