

# Realizzazione di un assistente vocale per il triage ospedaliero

Tesi di laurea di primo livello

Simone Montali - 09 luglio 2020

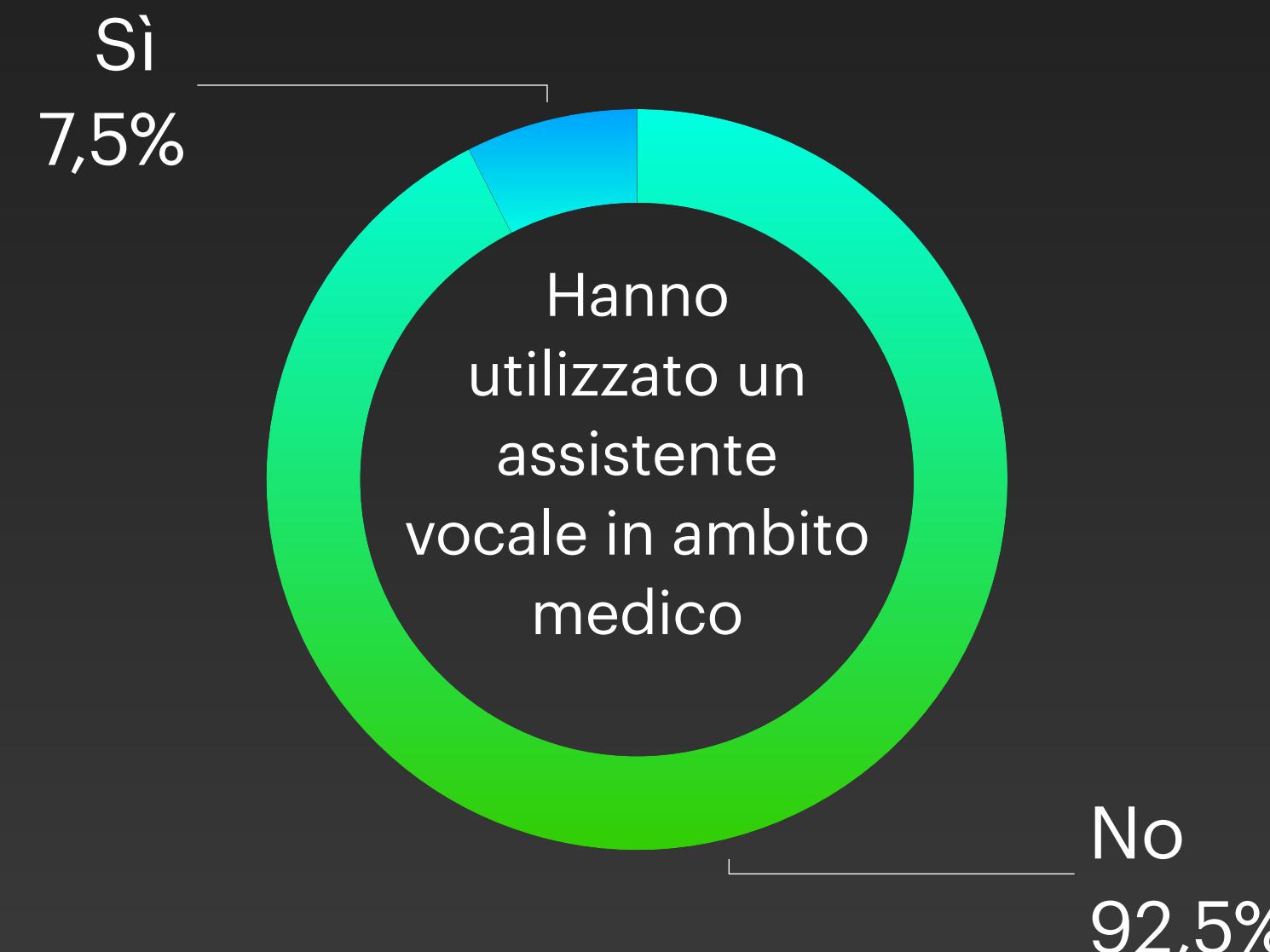


UNIVERSITÀ  
DI PARMA

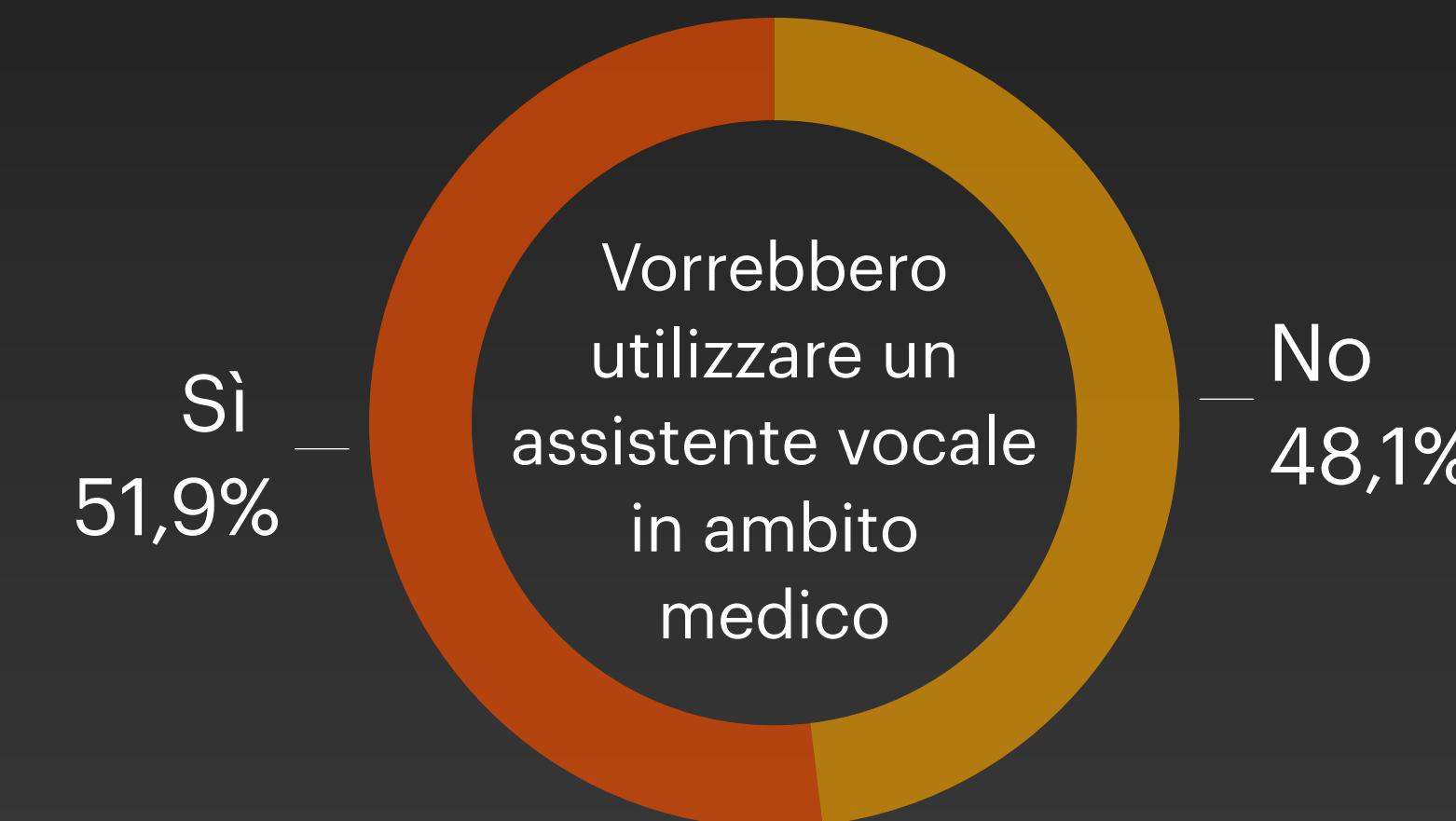
# Realizzazione di un **assistente vocale** per il triage ospedaliero

Cosa si intende con assistenti vocali? Quali sono le tendenze?

- Software in grado di **interpretare dialoghi umani** e **rispondere** attraverso voci sintetizzate
- In ambito medico, alcune ricerche di mercato hanno sottolineato l'interesse del pubblico ad un utilizzo di questi strumenti



Fonte: VoiceBot 2019



# Realizzazione di un assistente vocale per il **triage ospedaliero**

Che cos'è il triage?

- Il **triage** è il sistema utilizzato per **classificare** i soggetti coinvolti in infortuni, gravi o leggeri che siano, secondo **classi di emergenza** in base alla gravità delle lesioni riportate o del loro quadro clinico.
- Nasce in ambiti di guerra e maxiemergenze, ma è oggi utilizzato nei pronto soccorso per aiutare più velocemente i pazienti più a rischio.
- Non esiste uno standard nazionale, ma si tratta generalmente di 4 codici:

## Rosso

Paziente critico, cedimento  
di una o più funzioni vitali

## Giallo

Potenziale rischio di vita e/o  
compromissione delle funzioni vitali

## Verde

Valutazione medica  
differibile senza rischi

## Bianco

Paziente senza emergenza,  
visitabile dal medico di famiglia

# Protocolli e procedimenti

Come si procede con il triage? Com'è cambiato con il COVID19?

- Il protocollo più utilizzato in Italia è il **CESIRA**: **C**oscienza, **E**morragie, **S**hock, **I**nsufficienza Respiratoria, **R**otture Ossee, **A**ltro
- Durante la pandemia globale di **COVID19**, sono state studiate delle procedure di pre-triage per ridurre al minimo i contagi in ospedale
  - Generalmente, percorsi obbligati in cui i pazienti con sintomi compatibili vengono visitati in zone separate

Cammina?

È cosciente?

Ha emorragie?

È in stato  
di shock?

Ha insufficienze  
respiratorie?

Presenta rotture  
ossee?

Altre patologie?

# Mycroft

*"There's an entire community of developers looking to access this technology, but so far, it's been the purview of a few large companies. The technology is walled-off, proprietary, and secretive."*

Joshua Montgomery,  
CEO di Mycroft AI, Inc.



Cos'è Mycroft? Perché open source?

- Il progetto Mycroft nasce dalla totale mancanza di alternative open source agli assistenti vocali più utilizzati
- Il termine **open source**, nell'ambito dell'informatica, indica una tipologia di software il cui codice sorgente è pubblico.
  - Questo permette più sicurezza, possibilità di espansione, design pubblico...
- Mycroft nasce nel 2015 in seguito ad un crowdfunding.
- Conta oggi una grande comunità (4400+ ★ su GitHub)

# Stack di funzionamento

Come funziona un assistente vocale?



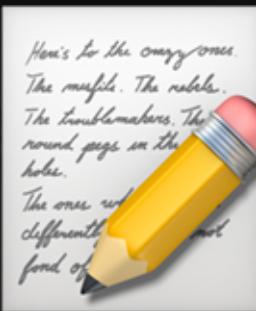
Rilevamento della wake word



## Interpretazione dell'intent



Generazione di una risposta



Speech To Text



Text To Speech

# Realizzazione di skills per Mycroft

Cosa rende Mycroft così potente?

- La caratteristica più utile di Mycroft è la sua **modularità**: ogni funzionalità è in un modulo diverso, detto **skill**.
  - Esisterà, ad esempio, *la skill per il meteo, quella per le barzellette, quella per la domotica...*
- È possibile implementare nuove skills tramite espansione della classe Python **MycroftSkill**
- Tramite files contenenti intent di esempio, viene addestrata la rete neurale di **Padatious**, l'intent parser di Mycroft.
- Tramite files contenenti dialoghi, si definiscono le risposte del bot nelle varie lingue.

# Handling di un sintomo del paziente

Cosa fa l'assistente vocale?

- Quando il paziente richiede aiuto, l'assistente vocale **lo intervista** riguardo ai suoi sintomi, al dolore provato, all'anagrafica...
- Viene assegnato al paziente un **codice** (Rosso, Giallo, Verde) in base alla gravità del sintomo.
- Salva tutte le informazioni raccolte in un file **JSON**.
  - L'utilizzo di uno standard come JSON permette di integrare l'assistente vocale con le **infrastrutture già presenti** in clinica.
- Se il sintomo è compatibile con il COVID19, si procede con una **diagnostica** più specifica:
  - Al paziente viene richiesto se ha *febbre, tosse, mal di gola, fatica respiratoria, contatti con infetti, mancanza di gusto*.
  - In base alle risposte, viene assegnato un **punteggio** di plausibilità di infezione da COVID19.

# Handling di un sintomo del paziente

## Cosa fa l'assistente vocale?

The screenshot shows a terminal window titled "mycroft-core : bash" with the following content:

- File Edit View Bookmarks Settings Help**
- Log Output:** 0-3 of 3 mycroft-core 20.2.4  
Establishing Mycroft Messagebus connection...  
Connected to Messagebus!  
^--- NEWEST ---^
- History =====**
- Log Output Legend =====**
  - DEBUG output
  - skills.log, other
  - voice.log
- Mic Level ==**

208	*
*	*
-*-	115.20
*	*
*	*
*	*
- Input (':' for command, Ctrl+C to quit) =====**  
> []
- mycroft-core : bash X mycroft-gui : bash X**

# Handling di un sintomo del paziente

Cosa fa l'assistente vocale?

Decorator dell'intent: collega il file con gli esempi di intent alla funzione

Decorator per i sintomi generici: richiede al paziente le informazioni anagrafiche, il dolore...

Decorator per i sintomi COVID: intervista il paziente riguardo ai sintomi tipici del COVID19

```
# BREATH
@intent_file_handler('symptoms.breath.intent')
@symptom_handler
@covid_symptom
def handle_breathing(self, message):
    """This function handles a "breathing fatigue" symptom.

    Breathing fatigue is recognized as a red code, and is a
    COVID-compatible symptom.

    Args:
        message: the message object returned from Mycroft

    GUI: show open mouth emoji
    """
    self.gui.show_text("😮 ")
    self.med_record["main_symptom"] = "breathing"
    self.med_record["code"] = "red"
    self.speak_dialog('symptoms.breath')
```

Visualizzazione nella GUI di un'emoji

Salvataggio delle informazioni nella scheda medica

Pronuncia della risposta

# E se il sintomo non fosse tra quelli definiti?

Creazione di un classificatore

# Fallback skills

Che cos'è una fallback skill di Mycroft?

- Mycroft funziona riconoscendo determinati *intent* nelle richieste dell'utente
  - E se la richiesta **non fosse definita** negli intent?
  - Le **fallback skills** sono skill che vengono attivate quando nessun'altra skill riconosce l'intent
  - Hanno una priorità di chiamata: se una fallback skill *cattura* il messaggio, non viene inoltrato alle altre
  - Questa tipologia di skill ci permette di non releggare l'utente a specifici messaggi, ma ad **analizzare in un secondo momento** la sua richiesta tramite un classificatore esterno

# Classificazione del testo

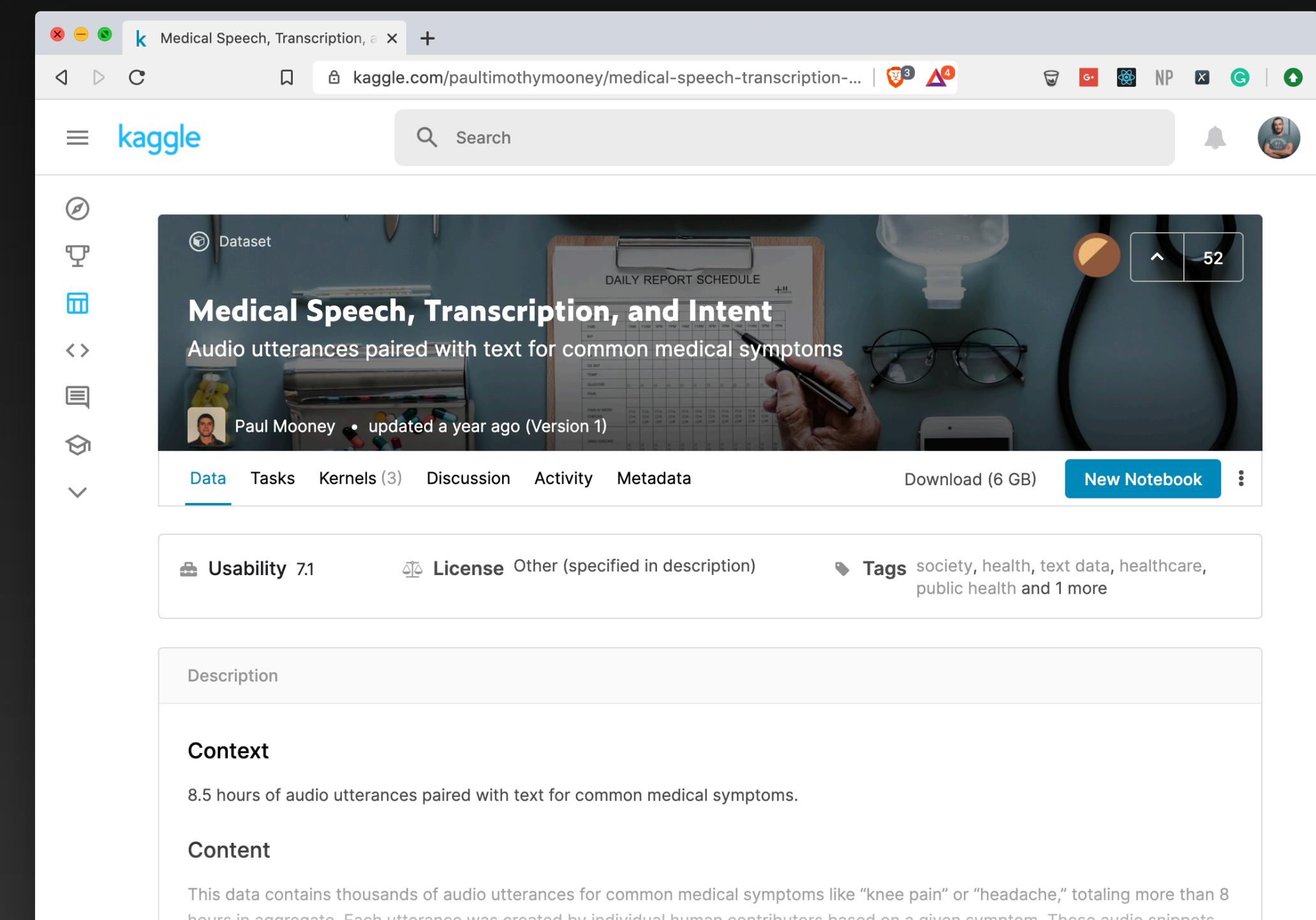
Che cos'è il Natural Language Processing? Che cos'è un classificatore?

- Con **Natural Language Processing** intendiamo quella branca dell'intelligenza artificiale che si occupa di processamento del testo
  - Lo scopo è quello di creare algoritmi che possano analizzare, rappresentare e quindi *comprendere* i testi in linguaggio naturale
- Vorremmo ora realizzare un **classificatore**: dato un testo in input, produrrà in output la classe di appartenenza
- Utilizzeremo questo classificatore per estrarre la *macrocategoria* di sintomo da una richiesta generica del paziente, in modo da indirizzarlo al reparto ospedaliero corretto

# Dataset e creazione del classificatore

Come addestrare il classificatore? Con quali dati?

- Cercando su internet è possibile trovare dataset creati ad-hoc per problematiche come questa.
  - Il dataset utilizzato per questa tesi contiene circa 8000 richieste di pazienti, classificate in più di 20 classi, come *dolore al piede, mal di schiena, raffreddore...*
  - Essendo in lingua inglese, è stata necessaria una traduzione verso l'italiano
  - Quest'ultima è stata fatta grazie ad uno script Python sfruttante il modulo **googletrans**



# Dataset e creazione del classificatore

Come addestrare il classificatore? Con quali dati?



```
data_clas = (TextList.from_csv(path, 'translations.csv',
                               cols='phrase')
              .random_split_by_pct(.2)
              .label_from_df(cols='prompt')
              .databunch(bs=42))

MODEL_PATH = "/tmp/model/"
learn = text_classifier_learner(data_clas,
                                 model_dir=MODEL_PATH,
                                 arch=AWD_LSTM)

learn.fit_one_cycle(5)
learn.export('/content/exported_model')
```

- Si è utilizzata, per la creazione del classificatore, la libreria **fastai**
- Tramite questa è stato creata una rete neurale ricorrente (**AWD-LSTM**), addestrata poi sul suddetto dataset
- Il modello generato è poi stato esportato ed inserito nel bot Mycroft (senza necessità di ri-addestramento)

# Applicazione del modello a Mycroft

Come viene sfruttato il modello?

- Registriamo una **FallbackSkill** in Mycroft che intercetti le richieste dei pazienti
- Dopo aver ricevuto il messaggio, **lo classifica** tramite il modello e chiede conferma all'utente
- Il bot recupera, dal file *classes.json*, la **gravità** del sintomo e l'**emoji** da mostrare nella GUI
- Se l'utente risponde in modo affermativo, l'interazione prosegue **come di consueto**.

```
❶ @symptom_handler
❷ def handle_fallback(self, message):
❸     utterance = message.data.get("utterance")
❹     symptom = self.classes[int(self.learner.predict(utterance)[0])]
❺     self.gui.show_text(symptom["emoji"])
❻     did_i_get_that = self.ask_yesno(
❼         'symptoms.fallback', {"symptom": symptom["name"]})
⽿     if did_i_get_that == "no":
⽻         self.did_i_get_that = False
⽼         self.speak_dialog('sorry')
⽽     else:
⽻         self.did_i_get_that = True
⽼         if symptom["covid"]:
⽽             self.ask_covid_questions()
⽽         self.med_record["main_symptom"] = symptom["name"]
⽽         self.med_record["code"] = symptom["code"]
```

**A volte i pazienti vogliono  
solo saperne di più.**

Richiesta di informazioni di carattere medico

# Ricerca di dati affidabili

Quale può essere una buona fonte di informazioni mediche? Come scaricarle?

- È palese che le informazioni debbano essere **affidabili**.
- Il Ministero della Salute rende disponibile, sul proprio sito, un'**enciclopedia medica**
- Possiamo **automatizzare il download** di queste informazioni tramite due approcci
  - Approccio API, scaricandole alla richiesta
  - Approccio periodico, scaricandole tutte, periodicamente



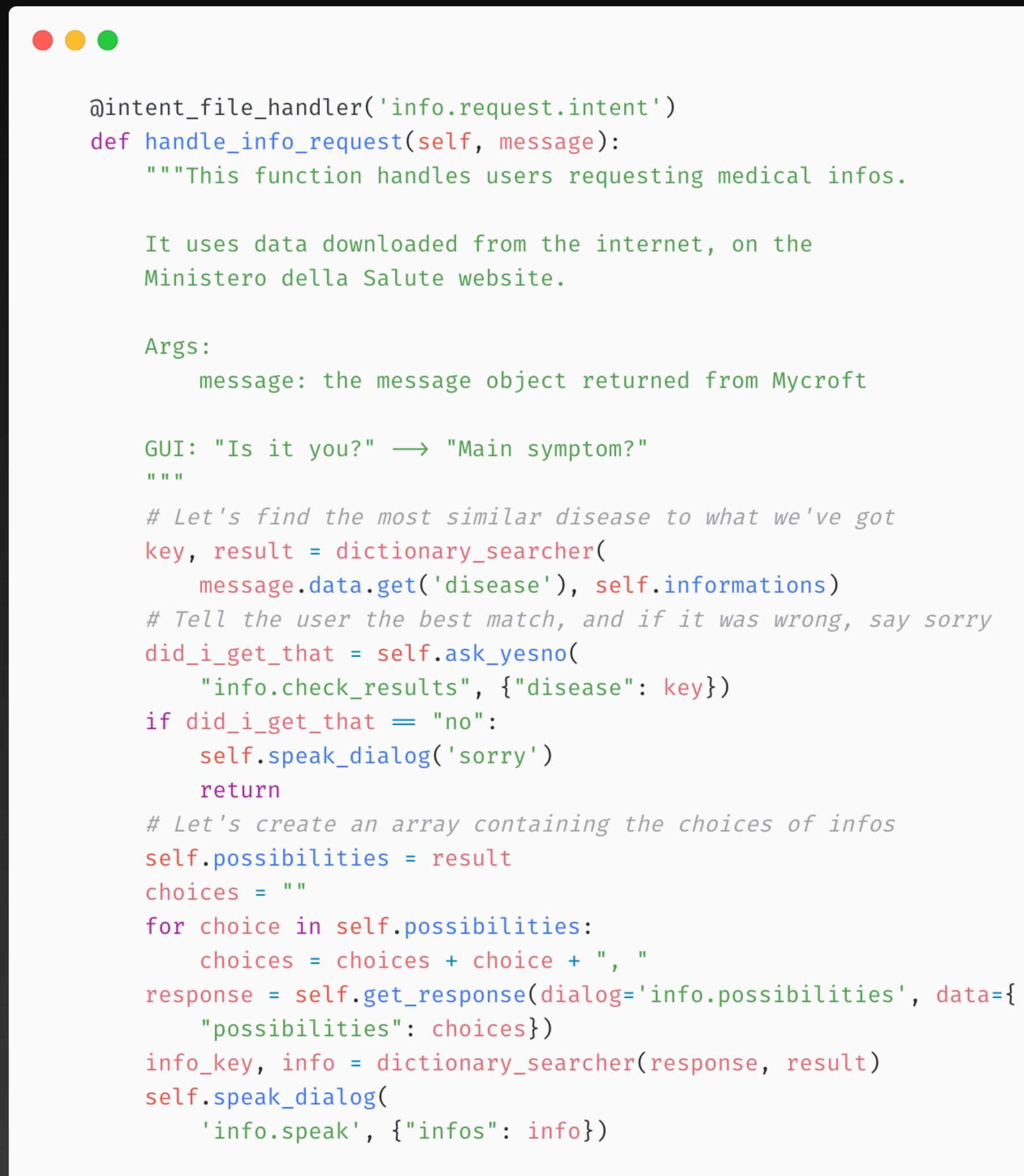
# Download dei dati

Cos'è Selenium? Come sfruttarlo per scaricare i dati?

- **Selenium** è un framework nato con lo scopo di testare applicazioni web
- Permette di interagire in modo automatico con un browser attraverso vari linguaggi di scripting, tra cui Python
- Creare un'API per la ricerca di informazioni mediche è quindi semplice: utilizzando la libreria `http.server` è possibile definire degli **handler per le richieste HTTP**
- Anche l'approccio periodico è possibile: basterà definire uno script che, elencate tutte le possibili voci dell'encyclopedia, le scarica una alla volta ed aggiunge ad un **file JSON**
  - Quest'ultimo script potrà poi essere inserito in una **cron table** per lo svolgimento periodico

# Integrazione delle informazioni in Mycroft

Come sfruttare queste informazioni in Mycroft?



```
@intent_file_handler('info.request.intent')
def handle_info_request(self, message):
    """This function handles users requesting medical infos.

    It uses data downloaded from the internet, on the
    Ministero della Salute website.

    Args:
        message: the message object returned from Mycroft

    GUI: "Is it you?" → "Main symptom?"
    """
    # Let's find the most similar disease to what we've got
    key, result = dictionary_searcher(
        message.data.get('disease'), self.informations)
    # Tell the user the best match, and if it was wrong, say sorry
    did_i_get_that = self.ask_yesno(
        "info.check_results", {"disease": key})
    if did_i_get_that == "no":
        self.speak_dialog('sorry')
        return
    # Let's create an array containing the choices of infos
    self.possibilities = result
    choices = ""
    for choice in self.possibilities:
        choices = choices + choice + ", "
    response = self.get_response(dialog='info.possibilities', data={
        "possibilities": choices})
    info_key, info = dictionary_searcher(response, result)
    self.speak_dialog(
        'info.speak', {"infos": info})
```

- Basterà ora definire un intent Mycroft per la richiesta di informazioni, da cui Padatious dovrà **estrarre il nome** della malattia richiesta
- Siccome è possibile che il paziente utilizzi termini simili ma non uguali, cercheremo il termine **più vicino**, piuttosto che la corrispondenza letterale
- Trovate le informazioni, il bot proporrà all'utente le **varie tipologie** di informazioni conosciute, come *sintomi, prevenzione, cura...*

# Quali sono i requisiti di questa tecnologia?

Su che dispositivo può funzionare? A che costi?

# Requisiti tecnici

Quali sono le necessità tecnologiche del progetto?

- Siccome l'elaborazione dello STT e del TTS sono **eseguite in cloud**, l'operazione che richiede più potenza di calcolo è l'interpretazione degli intent
- Sebbene Padatious richieda l'addestramento di una rete neurale, una volta eseguita quest'operazione l'esecuzione del bot è **molto leggera**
- Le scarse necessità tecniche del software rendono possibile l'installazione su sistemi molto economici, come il **Raspberry Pi 3**
- Il costo basso di questo sistema (che si aggira intorno ai 100 euro), permette a pressoché **tutte le cliniche del mondo** l'installazione dello strumento
- Per la GUI, si richiede l'installazione dell'ambiente grafico **KDE Plasma**, gratuito ed open source

# Conclusioni

Quali sono le potenzialità? Cosa è migliorabile?

- Strumenti come questo potrebbero aiutare un sistema ospedaliero spesso in difficoltà economiche, permettendo di aiutare **più pazienti, più velocemente, meglio**
- Il design **aperto ed espansibile**, oltre ad essere eticamente necessario, rende possibile l'integrazione con sistemi esistenti, la modifica, la correzione
- Un dataset migliore permetterebbe di affidarsi totalmente al classificatore, piuttosto che a **due livelli di skills**
- L'integrazione di più traduzioni permetterebbe di **abbattere la barriera linguistica** verso chi non conosce la lingua nazionale
- L'automazione delle procedure più *impersonali* del triage permetterebbe agli infermieri di **concentrarsi sui pazienti più in difficoltà**

# Codice sorgente

Dove posso trovare il codice di questa tesi?

Tutto il codice di questa tesi è open source con licenza GPL-3.0 ed è disponibile su GitHub

**Tesi:** <https://simone.codes/tesi>

**Skill principale:** <https://simone.codes/hospital-triage-skill>

**Skill fallback:** <https://simone.codes/hospital-fallback-skill>

**Scaper dell'Enciclopedia:** <https://simone.codes/medical-infos-api>

# Domande?

Qualche curiosità?

Grazie per l'attenzione.