UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Realizzazione di un chatbot: messaggistica istantanea per raccontare la storia dell'informatica basandosi su database di oggiSTI analizzati con NLP

Candidato: *Tiziano Labruna*

Relatore: *Giovanni A. Cignoni*

Correlatore: *Alessandro Lenci*

Anno Accademico 2016-2017

Indice generale

Introduzione

La parola *chatbot* è una crasi tra *chat* e *robot* e descrive un software capace di interagire con le persone in linguaggio naturale.
I primi chatbot erano in grado di fornire semplici risposte, spesso come risultato di trasformazioni dei messaggi degli utenti, ottenendo conversazioni prevedibili, ma che davano una parvenza di comprensione del linguaggio umano.
Negli ultimi anni, diversi social network hanno riscoperto queste tecnologie, offrendo piattaforme di sviluppo di chatbot che sono utilizzate con crescente successo da utenze di vario tipo e con diversi obiettivi.
I chatbot sono così diventati un importante canale con cui è possibile stabilire un contatto con gli utenti di un servizio (per esempio un'azienda e i suoi clienti).
Un chatbot è in grado di rispondere a domande e fornire informazioni in modo rapido ed efficace, soddisfacendo le richieste di numeri di utenti che sarebbe insostenibile gestire con operatori umani.

HMR è un progetto di ricerca che ha tra i suoi obiettivi quello di raccontare fatti, personaggi, scoperte, anche aneddoti, riguardanti la storia dell'informatica. Un chatbot può essere una buona soluzione per rispondere ai "clienti" di HMR, ossia tutti coloro interessati a conoscere questo tipo di contenuti.
Su quest'idea si è basato un tirocinio, nel periodo tra aprile e settembre 2017, che ha indagato la fattibilità del progetto. Questa tesi è nata come continuazione del tirocinio e ha portato alla realizzazione di un chatbot di supporto a HMR che fornisce informazioni sulla storia dell'informatica recuperandole da un database esistente, realizzato nell'ambito di un altro progetto di tirocinio/tesi.
In particolare, il database usato è quello di oggiSTI e per analizzare i testi contenuti al suo interno vengono utilizzati metodi di Natural Language Processing, costruendo così la base di conoscenza usata per le principali funzionalità di "conversazione" del bot.

Nel corso di questa relazione sarà inizialmente descritto il contesto nel quale ci stiamo inserendo (cap. 1), per poi analizzare in dettaglio il Natural Language Processing, come strumento utile ai nostri scopi (cap. 2).
In seguito saranno riportati i risultati dell'analisi di fattibilità del progetto (cap. 3), stabilendone limiti e obiettivi, per poi indicare i requisiti (cap. 4) da dover soddisfare, esposti secondo i criteri dell'ingegneria del software.
Infine saranno descritte le tecniche e le soluzioni messe in atto per arrivare alla realizzazione del chatbot (cap. 5) soddisfacendo i requisiti.

1. Chatbot: contesto e strumenti

1.1 Storia dei chatbot

La storia dei chatbot inizia molto prima di quanto si potrebbe pensare. Siamo nell'Inghilterra di metà Novecento quando Turing, ponendosi la domanda "Can machines think?" (Turing, 1950), propone un test che lega l'intelligenza alla capacità di sostenere una conversazione. Da allora la sfida a creare un software capace di simulare il linguaggio umano in modo sempre più accurato non si è mai arrestata. In questo capitolo verranno analizzati brevemente alcuni dei chatbot che sono stati i protagonisti di questa storia.

1.1.1 1950: "The Imitation Game" di Turing

Nel 1950 Alan Turing scrive un articolo intitolato *Computing machinery and intelligence* nel quale si pone il problema di stabilire un criterio per determinare se una macchina possa pensare. Il criterio si basa su un gioco, "*Il gioco dell'imitazione"*, nel quale sono presenti un computer A, un umano B e un giudice C, che deve stabilire chi è A e chi è B. Il giudice pone a entrambi delle domande a cui A e B rispondono in forma scritta. A vince il gioco nel momento in cui C sbaglia nel giudicare l'identità di A, credendolo umano.

Il gioco di Turing, nonostante le numerose critiche che lo giudicano un criterio non sufficiente a stabilire se una macchina sappia pensare (Wikipedia, voce *Chinese room*), ha fatto sì che nascesse una sorta di sfida nella quale, nel corso degli ultimi decenni, si sono susseguiti software che simulano il linguaggio umano in modo sempre più accurato.

Nel 1990 fu istituita una competizione chiamata *Loebner Prize* (Loebner Prize, sito web) che si basa sull'esecuzione del test di Turing per premiare il computer il cui comportamento sia più simile al pensiero umano.

1.1.2 1966: Eliza

Creato da Joseph Weizenbaum, Eliza è un programma che si pose come obiettivo quello di creare una parodia di una conversazione tra un terapeuta e il suo paziente nella sua fase iniziale, nella quale avviene uno scambio di semplici domande che non richiedono grande profondità emotiva.

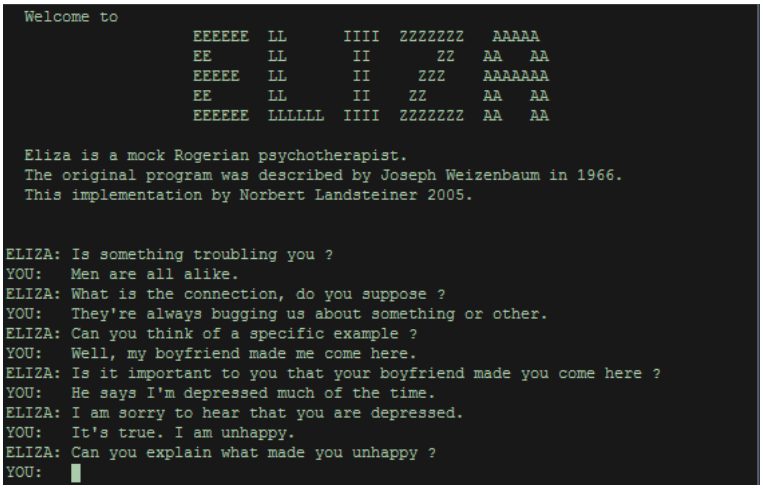

Figura1: Esempio di chat con Eliza

Eliza (vedi Figura1) si basa sull'uso di espressioni regolari per trasformare gli input degli utenti nei suoi output, in modo da dare continuità alla conversazione, oltre che un'apparente coerenza. Altre volte produce osservazioni acontestuali, ma perfettamente in linea con l'atteggiamento di uno psicoterapeuta, come "Very interesting. Plese, go on" o "Can you elaborate on that?" (Eliza, computer therapist, sito web).

1.1.3 1972: Parry

Restando in ambito medico, Parry simulava il comportamento di un individuo paranoico e schizofrenico. Fu sviluppato dallo psichiatra Kenneth Colby come strumento di pratica per i suoi studenti prima che essi potessero trattare con pazienti reali.

In una leggera variazione del test di Turing, fu chiesto a degli psichiatri di leggere trascritti di Parry e pazienti reali per distinguere quale dei due fosse il computer, ottenendo un risultato incredibile al tempo: solo il 48% delle volte riuscirono a identificare una differenza nel comportamento. (Parry, The A.I. chatterbot from 1972)

1.1.4 1988: Jabberwacky

L'obiettivo dichiarato del bot creato dal programmatore Rollo Carpenter era quello di superare il test di Turing. Jabberwacky era in grado di simulare la voce umana in modo divertente e umoristico. Attualmente sono ancora in corso sviluppi sul bot rivolti a implementare il sistema su robot o *talking pets* basati su un apprendimento completamente sonoro. (Jabberwacky, sito web)

Cleverbot è una variante di Jabberwacky rilasciato nel 1997 che ottenne grandi risultati: nel 2011 ha partecipato a un test di Turing all'*IIT Guwahati* in India venendo giudicato umano al 59,3% (Software tricks people into thinking it is human, sito web).

1.1.5 1992: Dr Sbaitso

Anche Dr Sbaitso fu progettato per simulare il comportamento di uno psicologo in grado di risolvere i problemi emotivi degli utenti, ed era utilizzabile da personal computer con sistema operativo MS-DOS. Fu sviluppato dalla Creative Labs con l'obiettivo di dare dimostrazione della capacità delle schede audio di generare una voce sintetizzata.

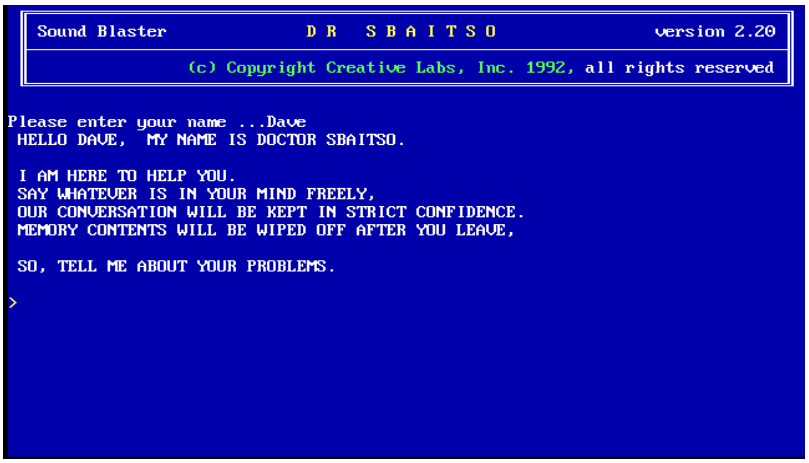

Figura 2: Esempio di chat con Dr. Sbaitso

La maggior parte delle domande era del tipo "WHY DO YOU FEEL THAT WAY?" (vedi Figura 2), evitando così interazioni più complicate. Quando riceveva una frase che non era in grado di capire, di solito rispondeva con "THAT'S NOT MY PROBLEM" (Dr. Sbaitso, sito web).

1.1.6 1995: A.L.I.C.E.

Alice (Artificial Linguistic Internet Computer Entity) venne realizzato come software *open source* basato sul *natural language processing*. Fu progettato con il linguaggio A.I.M.L. dallo scienziato Richard S. Wallace.

Il sistema di interpretazione di Alice si basava su un approccio minimale. Il significato di una frase veniva elaborato tramite specifiche parole chiave o termini (radici), evitando analisi approfondite e complesse.

Alice ha vinto per ben 3 volte il Loebner Prize, nel 2000, 2001 e 2004. (A.L.I.C.E. Artificial Intelligence Foundation, sito web)

1.1.7 2001: Smarterchild

Smarterchild (vedi Figura 3) fu un chatbot di grande successo ed era disponibile su *AOL Instant Messenger* e *MSN Messenger*. Sviluppato da *ActiveBuddy, Inc.* arrivò ad essere usato da oltre 30 milioni di utenti.

Dal rapido successo di SmarterChild sono derivati bot orientati al marketing per Radiohead, Austin Powers, Intel, Keebler, The Sporting News e altri. (Smarterchild, sito web)

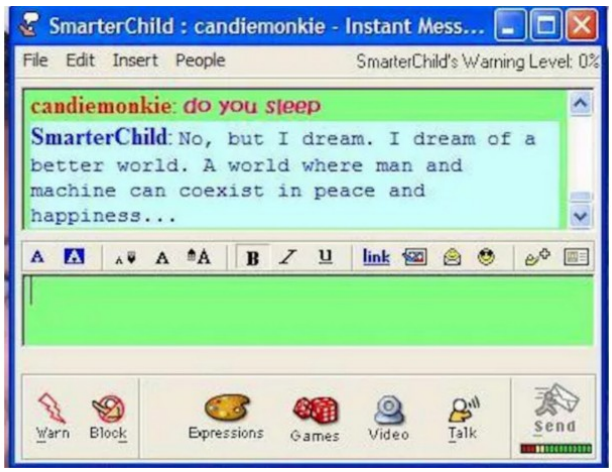

Figura 3: Esempio di chat con Smarterchild

1.1.8 2006: IBM Watson

Watson è stato un sistema di intelligenza artificiale, sviluppato da IBM, in grado di rispondere a domande espresse in linguaggio naturale.

In principio Watson fu creato per concorrere a un quiz televisivo americano chiamato *Jeopardy!*. Alla prima partecipazione, tuttavia, riuscì a rispondere a solo il 35% delle domande. In seguito a numerosi miglioramenti da parte di un team IBM, Watson ci riprovò nel 2011 e stavolta riuscì a sconfiggere i campioni umani del quiz. Durante il gioco, Watson funzionava senza essere connesso a Internet, sfruttando un'occupazione di 4 terabytes di spazio disco.

In seguito, fu usato in molti altri contesti di tutt'altra rilevanza, come la gestione delle decisioni nel trattamento del cancro ai polmoni al *Memorial Sloan-Kettering Cancer Center*.

1.2 Chatbot: tecniche di I.A.

Nei primi chatbot venivano usati degli algoritmi piuttosto semplici per analizzare il messaggio in input e restituire una risposta in output, che puntavano a dare l'impressione che il computer stesse comprendendo ciò che si voleva comunicare dando delle risposte coerenti. Col passare del tempo e l'evolversi delle tecnologie, sono stati creati metodi di Intelligenza Artificiale sempre più sofisticati con cui i chatbot erano in grado di simulare conversazioni sempre più vicine a passare il test di Turing.

In questo capitolo verranno elencate e introdotte brevemente le principali tecniche usate dai chatbot esistenti ed esistiti, da cui trarre spunto per poter dotare di intelligenza il nostro bot.

1.2.1 Preparazione del testo

Prima di poter applicare strategie di interpretazione del testo è necessario compiere una serie di elaborazioni. In particolare sono importanti le seguenti fasi:

- Text cleaning – il testo viene ripulito da tutti gli elementi che possono alterare successive analisi (per esempio degli spazi all'inizio e alla fine del messaggio).
- Verifica dei caratteri del testo – si controlla se nel testo sono presenti caratteri equivalenti ad altri che potrebbero invalidare successive analisi (per esempio nell'italiano "è" ed "é" hanno lo stesso valore semantico e potrebbero essere confusi da errori grammaticali di cui bisogna tener conto).
- Normalizzazione del testo – trasformazione dei caratteri maiuscoli in minuscolo, per far sì che una stessa parola scritta con una maiuscola invece che una minuscola venga interpretata allo stesso modo (questo approccio non è sempre ottimale, in quanto a volte le maiuscole possono avere valore discriminativo, es. Agnelli vs. agnelli).

(Lenci, Montemagni e Pirrelli, 2005)

1.2.2 Pattern Matching

In Informatica per "pattern matching" si intende l'atto di controllare se in una sequenza di token è presente un certo "pattern", ossia una combinazione di caratteri che rispetti un certo schema.

Per quanto riguarda l'interpretazione degli input da parte di un chatbot, il Pattern Matching può essere utile per riconoscere determinati insiemi di messaggi. Per esempio, grazie al Pattern Matching è possibile rispondere "Ciao!" a tutti i messaggi che contengono la parola "ciao" o la parola "salve"; oppure riconoscere se un messaggio è di tipo interrogativo controllando se l'ultimo token è "?".

Uno strumento molto utile per il riconoscimento dei pattern è l'*espressione regolare*, che offre un sistema di notazione per identificare insiemi di stringhe.

(Wikipedia, voce *Pattern Matching*)

1.2.3 Trasposizione di parole

Tecnica usata largamente da chatbot del tipo di Eliza, consiste nel riformulare il messaggio di input per generare un output corrispondente. Per esempio se l'utente scrive "sei un computer" la risposta del chatbot sarà "quindi pensi che sono un computer".

Le sostituzioni che vengono effettuate utilizzando questa tecnica riguardano principalmente i pronomi personali (tu → io) e i verbi (sei → sono), trasformando quindi tutte le forme in prima persona in forme in seconda persona e viceversa.

(How to build Eliza Chatterbot, sito web)

1.2.4 Memorizzazione del contesto

Memorizzare il contesto è una strategia utilizzata per tenere traccia di ciò che è stato detto in precedenza e poterlo riutilizzare per la conversazione. Diventa necessaria nel momento in cui la risposta del chatbot non può basarsi solo sull'ultimo messaggio inviato dall'utente, ma deve trarre le informazioni da un qualche messaggio precedente.

Esempio:

Utente: "Mi chiamo Mario"

Chatbot: "Ok Mario"

Utente: "Qual è il mio nome?"

Chatbot: "Mi hai detto prima di chiamarti Mario"

Nel caso in cui, invece, l'utente non avesse ancora dichiarato il suo nome, la risposta del chatbot dovrà essere qualcosa tipo "Ancora non mi hai detto come ti chiami". È facile capire, quindi, come l'utilizzo del contesto per formulare una risposta sia estremamente importante per poter dare al chatbot parvenze molto meno meccaniche e più umane.

Avere memoria dei messaggi precedenti è utile anche per rilevare quando l'utente invia un messaggio in modo ripetuto, o per evitare che sia lo stesso chatbot a inviare messaggi uguali, controllando il valore dell'ultimo messaggio prima di scegliere il prossimo.

1.2.5 Ranking di possibili risposte

Una tecnica largamente utilizzata dai moderni chatbot (Chatbot Tutorial, sito web) è quella del *ranking*.

A partire da un messaggio inviato da un utente, ci possono essere più *pattern* considerati accettabili, ossia più interpretazioni che il chatbot può dare a uno stesso messaggio. Per esempio, se il messaggio è "Come ti chiami?", il bot riconoscerà la stringa "Come" per la quale sa dare una certa risposta, ma riconoscerà anche la stringa "Come ti chiami" per la quale ha una certa altra risposta. Tra i due pattern riconosciuti è certamente più accurato il secondo nell'ottica di poter dare una risposta coerente a ciò che viene domandato.

È chiaro quindi come non tutti i pattern abbiano uno stesso valore e risulta per questo necessario associare ad ognuno di essi un valore specifico di accuratezza.

1.2.6 Servizi che utilizzano tecniche di I.A.

Esistono alcuni servizi web che rendono disponibile l'utilizzo delle tecniche già esposte, oltre ad alcune altre, per l'analisi istantanea del testo.
Per quanto riguarda l'analisi dei testi italiani, la varietà di questi strumenti è molto minore rispetto all'inglese, tuttavia ne esistono alcuni interessanti e qui in seguito

saranno esposti quelli che sono stati considerati per il progetto, con le rispettive descrizioni delle prove effettuate per ognuno di essi.

1.2.6.1 Meaning Cloud

Meaning Cloud (Meaning Cloud, sito web) offre diversi servizi di analisi del linguaggio naturale per molte lingue, tra cui l'italiano. È possibile effettuare analisi sia sul testo semplice, sia sul testo contenuto all'interno di un file Excel, essendo disponibile un add-in direttamente installabile in locale.

Sono state effettuate delle prove per valutare se i servizi offerti da Meaning Cloud potessero soddisfare le esigenze del progetto, tuttavia i risultati ottenuti sono stati piuttosto scarsi. Il servizio web appare puntato soprattutto sull'analisi di testi inglesi, per i quali gli output sono spesso ricchi e accurati, mentre per quelli italiani molte volte non viene trovata nessuna corrispondenza o viene trovata in modo errato.

1.2.6.2 Tree Tagger

Tree Tagger (Tree Tagger, sito web) è uno strumento di annotazione del testo che serve ad estrarre la POS[1] ed il lemma da ogni parola. Sviluppato da Helmut Schmid, viene utilizzato per ben 22 lingue diverse, tra cui anche l'italiano.

Le prove effettuate hanno avuto esiti positivi in quanto a funzionamento ed efficacia, tuttavia gli output ottenuti, se pur corretti, non potevano essere sufficienti per i nostri scopi: i soli POS e lemmi delle parole non bastano per poter effettuare le analisi di cui necessitiamo per il progetto.

1.2.6.3 Tint

The Italian NLP Tool (Tint, sito web) è un servizio per l'analisi del linguaggio naturale italiano, basato su Java. Dispone della maggior parte delle comuni analisi linguistiche e può essere usato sia come applicazione indipendente che come servizio web.

A seguito delle prove effettuate, questo servizio, se pur molto ben funzionante, non è risultato essere compatibile con il progetto.

1.2.6.4 Tanl Italian Pipeline

Tanl Italian Pipeline (Tanl Italian Pipeline, sito web) è un servizio web per l'analisi di testo italiano, che effettua 5 tipi di analisi differenti, dalla semplice individuazione dei POS alla costruzione di un albero di dipendenze semantiche.

Per effettuare le prove è stato necessario installare una serie di componenti sul server del progetto, dopodiché i risultati sono stati molto soddisfacenti, se pur non sempre perfetti, ed è stato considerato come l'opzione migliore ai fini del progetto.

1 Part of speech, o parte del discorso in italiano, è la categoria grammaticale a cui appartiene una parola.

1.3 Chatbot: le varie piattaforme

I *chat bot*, o *chatbot*, sono dei servizi basati su regole e un qualche tipo di intelligenza artificiale con i quali è possibile interagire tramite un'interfaccia chat. Il tipo di servizio può essere molto diversificato, può essere strettamente funzionale oppure al solo scopo di svago e può essere offerto da molti produttori di applicazioni chat diversi.

La prima piattaforma di messaggistica ad introdurre questo tipo di interazioni è stata *Telegram*, nel giugno 2015, seguita da *Skype*, *IMessage* e *WeChat*.

1.3.1 Piattaforme attualmente disponibili

Ad oggi, le principali piattaforme che offrono la possibilità di creare un proprio chatbot sono:

- Telegram
- Skype
- Facebook Messenger
- Twitter

1.3.1.1 Telegram

Telegram è un servizio di messaggistica istantanea, creato come organizzazione non a fini di lucro dai fratelli russi Nikolai e Pavel Durov.
Con il rilascio di *Telegram 3.0* nel giugno 2015, vennero introdotti i bot, un'idea nuova per le applicazioni di messaggistica, in seguito imitata da molti altri.
Sin dai suoi esordi, Telegram si caratterizza per la libertà di utilizzo delle sue *API* e del suo codice. Anche in questo caso, nonostante vennero subito rilasciati dei bot ufficiali (Bots: An introduction for developers, sito web), venne lasciata aperta la possibilità a chiunque di creare il suo proprio bot.

1.3.1.2 Skype

Nel marzo 2016 *Microsoft* annuncia un kit di sviluppo per importare bot già programmati per altre piattaforme, e crearne di nuovi (Say hello to Skype chat bots, sito web).
Oltre alle chat testuali, l'intenzione è di andare a gestire anche i messaggi vocali, per includere Cortana: l'assistente vocale potrà così aiutare gli utenti a interagire con diverse app, per identificare persone, cercare o sottolineare parti di testo o mostrare informazioni aggiuntive sui contenuti discussi.

1.3.1.3 Facebook Messenger

Facebook Messenger è l'applicazione più usata nel mondo con più di 1 miliardo di utenti ed è strettamente collegata a Facebook, la piattaforma di social network globalmente dominante. Nell'aprile 2016 dal palcoscenico della F8 Conference a San Francisco, Mark Zuckerberg ha annunciato l'entrata in campo di Messenger sul palcoscenico dei bot, rivelando l'obiettivo di diventare il canale preferenziale per le aziende per comunicare con i clienti: tramite un bot, un utente Facebook potrà ordinare prodotti, o ottenere notizie personalizzate da parte dei media.

1.3.1.4 Facebook Messenger Lite

Si tratta di una versione leggera di Fb Messenger, arrivata in Italia all'inizio di maggio 2017 che permette di scambiare messaggi allo stesso modo dell'applicazione madre (quindi anche conversazioni con i bot) anche a smartphone con prestazioni basse.

1.3.1.5 Twitter

Il primo Novembre 2016 sul blog di Twitter (Twitter Platform, sito web) appare l'annuncio che comunica ai suoi utenti che sarà possibile usare i messaggi privati dell'applicazione per poter conversare direttamente con le aziende, grazie a dei bot che fungono da servizio clienti, per esempio dando informazioni su una spedizione, o sui prodotti in vendita. Tuttavia Twitter non offre supporto per lo sviluppo di bot da parte di privati.

1.3.2 Popolarità delle applicazioni

Per poter scegliere la piattaforma migliore sulla quale far girare il bot è fondamentale sapere quali sono le app più usate dagli utenti. A tale scopo può esserci utile sapere quali sono le applicazioni di comunicazione più scaricate:

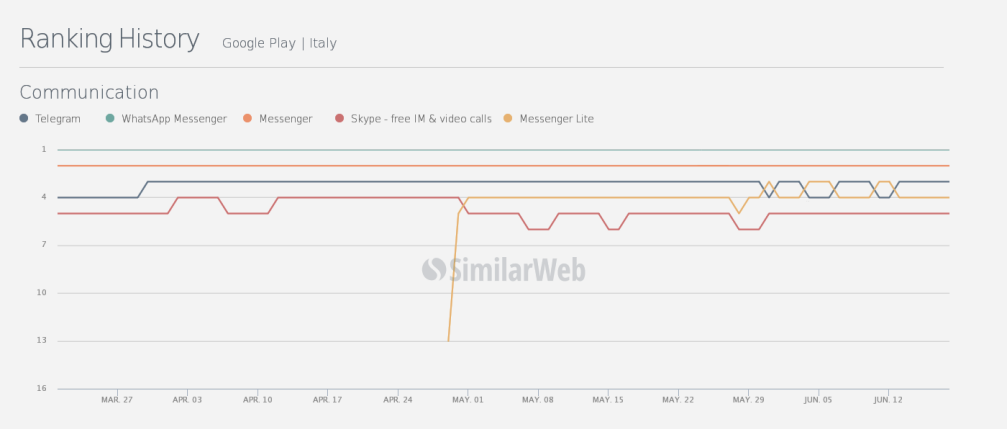

Figura 4: Statistiche relative ai download effettuati su Google Play in Italia nel periodo Marzo-Giugno 2017 (SimilarWeb, sito web)

Come si può notare in Figura 4, la situazione attuale vede Whatsapp al primo posto, seguita da Facebook Messenger, da Telegram, da Messenger Lite e infine da Skype.
Nel grafico di Figura 5 vediamo un altro confronto in termini di numero di utenti per ogni applicazione:

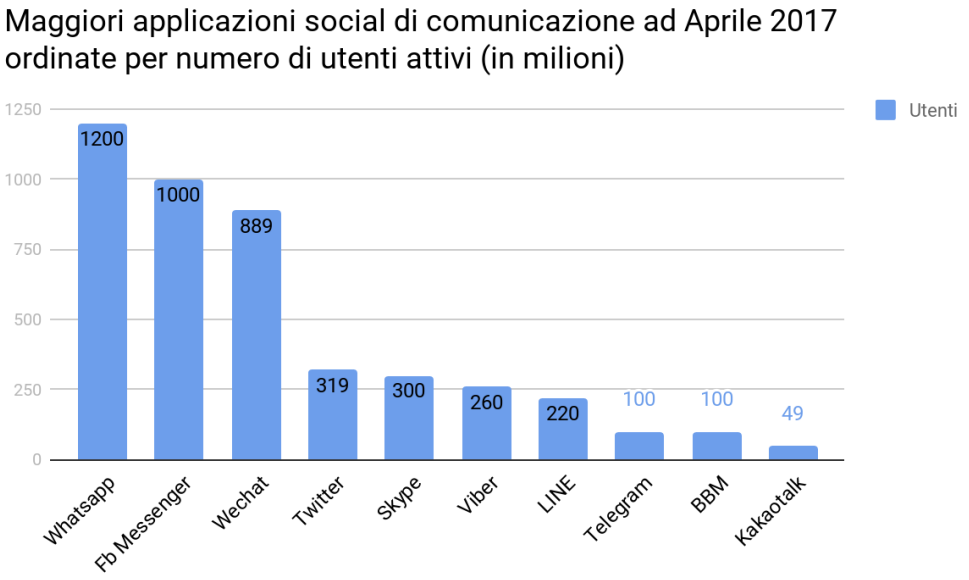

Figura 5: (Statista, sito web)

Whatsapp è nettamente dominante (il suo numero di utenti supera il miliardo), ma deve essere esclusa dalla scelta, in quanto non dispone di un supporto per lo sviluppo di bot.
Essendo Messenger Lite la versione leggera di Messenger, ma facente parte dello stesso sistema di chat, possiamo escludere anche questa, inglobandola dentro la piattaforma Facebook Messenger.

Twitter, è a tutti gli effetti un'applicazione (fruibile sia da pc che da smartphone) che permette di scambiare messaggi con altri utenti e come abbiamo detto anche con i chatbot.
Tuttavia, ci sono alcuni motivi per cui deve essere anch'essa esclusa dalla nostra scelta:

1. Si tratta di un social network in cui la funzione primaria è quella di pubblicare *tweet*, ossia messaggi pubblici mandati in broadcast. Non può essere considerata un'applicazione di messaggistica istantanea (e per questo non è presente nel grafico della Figura 4).
2. La funzionalità di direct message, per chattare in modo privato tra due utenti, è molto marginale e non dispone di nessuna funzione aggiuntiva che non sia il semplice invio di un messaggio di testo.
3. I numerosi bot di Twitter, circa il 15% degli account (Il cinguettio dei «bot»: su Twitter il 15% degli account è un software, sito web), sono per la maggior parte bot che funzionano comunicando con gli utenti non tramite messaggi privati ma tramite *tweet*, cioè inviando dei messaggi pubblici come risposta automatica ai *tweet* degli utenti.
4. Twitter non offre nessun tipo di supporto agli sviluppatori che intendano programmare un proprio chatbot e questo rende il lavoro più dispendioso e meno standardizzato.

A questo punto la nostra scelta si riduce a sole 3 opzioni: Fb Messenger, Telegram e Skype.

1.3.3 Confronto delle funzionalità

Tramite la tabella sottostante saranno messe a confronto le principali caratteristiche dei chatbot offerti dalle 3 piattaforme e per ogni caratteristica sarà indicato se è o meno utile ai fini del progetto.

	Facebook Messenger	Telegram	Skype	Utile per il progetto
Può inviare notifiche personalizzate	✓	✓	✓	✓
Permette di chattare vocalmente con il bot	✗	✗	✓	✗
Può essere aggiunto a un gruppo	✗	✓	✗	✓
Posso richiamare il bot da una qualunque chat	✗	✓	✗	✓
Posso dare delle risposte preimpostate	✓	✓	✓	✓
Posso effettuare pagamenti tramite il bot	✓	✓	✓	✗
Posso inserire un input tramite una tastiera personalizzata	✗	✓	✗	✓
Posso usare dei comandi preimpostati	✗	✓	✗	✓
Esiste un bot store	✗ (potrebbe essere rilasciato con il prossimo F8)	✓	✗ (solo i bot ufficiali)	✓

È possibile applicare template personalizzati ai messaggi				
È possibile formattare il testo				 (marginalmente)

1.3.4 Analisi conclusive e scelta

Dopo aver selezionato le applicazioni di messaggistica più usate in Italia nelle quali può essere possibile programmare e rendere funzionante un chatbot, dobbiamo decidere quale tra queste è quella che fa più al caso nostro. Per questo non dobbiamo perdere di vista quello che è il nostro obiettivo e cioè realizzare un bot con il quale gli utenti possano chattare in modo facile e che sappia fornire tutte le informazioni richieste.
Nella tabella precedente si nota che ci sono alcune caratteristiche possedute solo da Messenger, altre solo da Telegram e altre solo da Skype; tuttavia non tutte queste caratteristiche sono utili allo stesso modo al nostro scopo.
Skype presenta la funzione molto avanzata di poter comunicare con il bot in maniera vocale, ma questo non è un grande vantaggio nel nostro caso. L'uso principale di Skype infatti è proprio quello di effettuare chiamate vocali e videochiamate, la maggior parte degli utenti che installa Skype lo fa per potersi mettere in contatto con persone a lui lontane, non certo per chattare con un bot; e neanche avrebbe senso scaricare un'applicazione pesante che offre numerose funzioni extra solamente per usare una chat.
Fb Messenger, come caratteristica distintiva, permette di personalizzare i template dei messaggi, ma anche questo non è un vantaggio rilevante, in quanto riguarda il lato estetico più che quello funzionale. C'è da dire però che, nonostante nel ranking di scaricamento delle applicazioni Messenger stacchi Telegram di una sola posizione, la differenza numerica assoluta è considerevole: nel mondo, più di 1 miliardo di persone ha scaricato Messenger, mentre solo 100 milioni Telegram.
Il confronto in realtà non è paritario, in quanto la prima è stata rilasciata nel 2011, forte della fama mondiale che già aveva Facebook (fondata nel 2004), mentre la seconda è nata nel 2013 e ha raggiunto una vera diffusione internazionale soltanto nel 2015.
Ad oggi, l'applicazione di Pavel Durov, sta avendo una crescita molto rapida (350 mila nuovi utenti al giorno) ed è prevedibile che di qui a non molto tempo il confronto numerico sarà ben diverso.
Ma ciò che fa la differenza sono le funzioni che ha Telegram in più rispetto alle concorrenti: è possibile creare un gruppo ed aggiungere il bot al gruppo, in modo che più persone possano fare domande al bot e le risposte siano visualizzabili a tutti; è possibile chiamare in causa il bot e fargli una domanda da qualunque chat privata; è possibile personalizzare la tastiera in modo da visualizzare più opzioni

preimpostate di risposta; è possibile creare una lista di comandi, così che l'utente la possa scorrere e selezionare un comando per il quale avrà una determinata risposta dal bot. Per questi motivi, e non solo, la scelta è ricaduta su Telegram.

1.3.5 Telegram: aspetti generali

Nella pagina ufficiale di Telegram (Telegram Messenger, sito web) si leggono alcuni aspetti tecnici che caratterizzano l'applicazione. Tra i suoi punti di forza c'è il fatto di essere completamente multipiattaforma e quindi poter essere usata da qualunque dispositivo in maniera indipendente. È possibile inviare file di molte estensioni diverse e di grandi dimensioni. La comunicazione avviene in modo peer-to-peer con crittografia, permettendo un'elevata sicurezza (è anche possibile inviare messaggi con un timer di auto-distruzione). La consegna dei messaggi avviene in modo molto veloce e questo è possibile anche grazie al fatto che i suoi server sononumerosi e sparsi in tutto il mondo. Inoltre Telegram è un'applicazione gratuita e non vengono visualizzati inserti pubblicitari durante l'utilizzo.

Questi, in aggiunta alle funzionalità dei suoi bot analizzate in precedenza, sono alcuni dei motivi che hanno contribuito alla scelta di Telegram come applicazione su cui far girare il chatbot del progetto.

1.3.6 Telegram: funzionamento dei bot

Sarà adesso analizzato il funzionamento dei bot Telegram e il modo in cui vengono gestiti i messaggi ricevuti dagli utenti (update) e le relative risposte.
Telegram supporta due modi mutuamente esclusivi di processare gli update:

- getUpdates, un meccanismo pull
- setWebhook, un meccanismo push

In entrambi i casi, gli update vengono immagazzinati nel server Telegram per un tempo massimo di 24 ore.
Nel seguito sono descritte le differenze di funzionamento dei due metodi, usando diagrammi di sequenza UML.

1.3.6.1 Metodo getUpdates

Con il metodo getUpdates (Telegram Bot API, sito web), si verificano 2 momenti distinti che possiamo analizzare tramite il diagramma di sequenza UML della Figura 6. All'invio di un messaggio di un utente al bot, quest'ultimo spedisce un update a Telegram, ossia un oggetto contenente il testo del messaggio, il codice dell'utente e altre informazioni. A questo punto l'update viene salvato nella memoria dei server di Telegram e al momento dell'arrivo di nuovi update, questi si aggiungono alla coda.

In un momento successivo, l'amministratore del bot può richiedere (tramite per esempio un programma che risiede su un server) di visualizzare le update del bot e questo viene fatto tramite il metodo getUpdates.

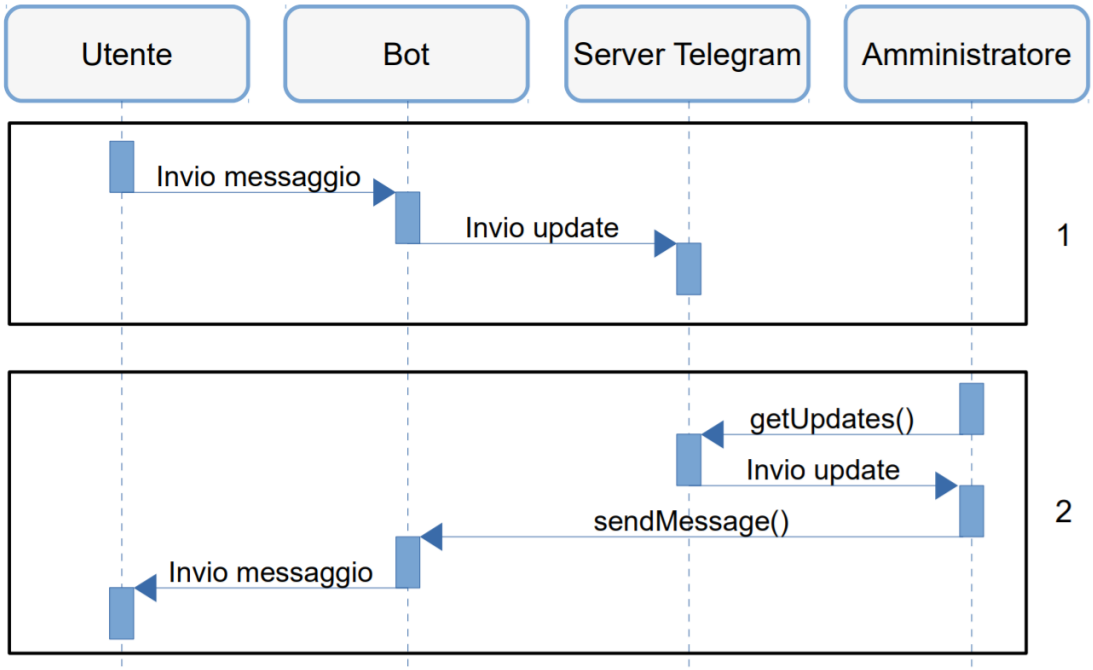

Figura 6: Funzionamento del metodo getUpdates

Come mostrato in Figura 6, una volta ricevute le update da Telegram, queste verranno analizzate e in base a un qualche algoritmo verrà restituita una risposta. Per inviare il messaggio di risposta si utilizza il metodo sendMessage, che permette al bot di inviare un messaggio all'utente che viene specificato.

Come abbiamo visto, quindi, questo metodo richiede che ci sia una continua richiesta di informazione per poter verificare se ci sono messaggi pendenti a cui dover dare una risposta e non risulta essere conveniente nel caso in cui si voglia avere un bot funzionante stabilmente nel tempo.

1.3.6.2 Metodo setWebhook

Con il metodo che utilizza lo Webhook (Marvin's Marvellous Guide to All Things Webhook, sito web) sia l'invio del messaggio che la sua elaborazione e successivo invio di una risposta avvengono in un unico momento.
Iniziamo con l'analizzare il funzionamento generale tramite il diagramma di sequenza di Figura 7.

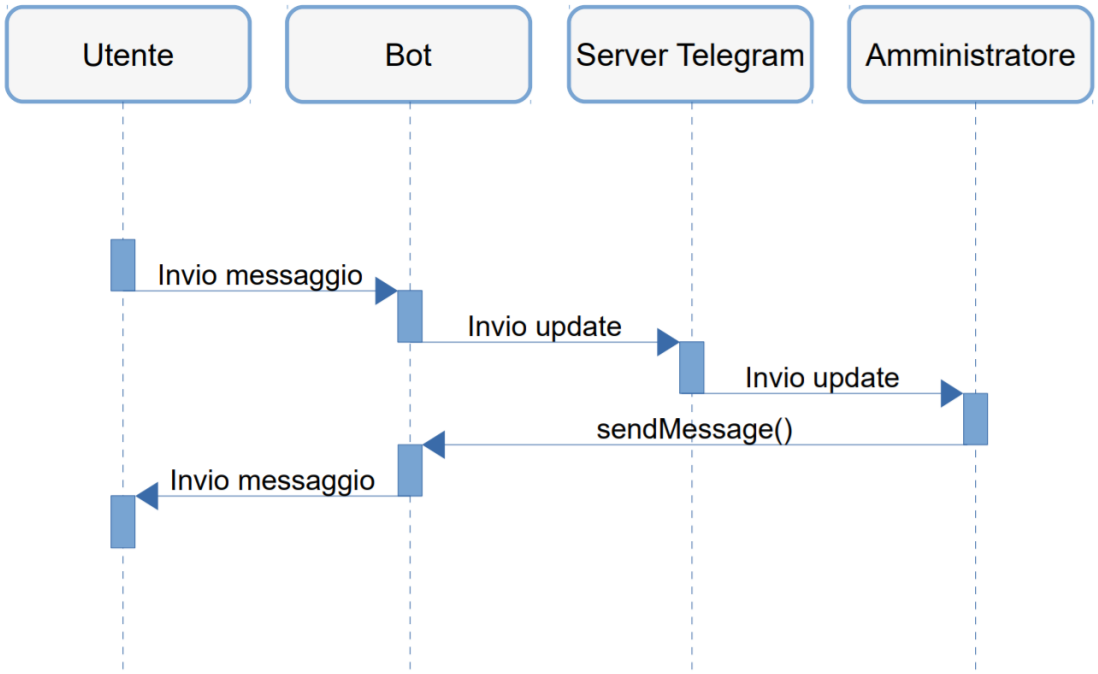

Figura 7: Funzionamento del metodo setWebhook

Analogamente al metodo precedente, nel momento in cui un utente invia un messaggio al bot, questo lo inoltra al server Telegram tramite una update. Stavolta, però, la update viene immediatamente inviata al programma che si occupa di analizzarla e di inviare la risposta all'utente con lo stesso metodo sendMessage().
Per sapere a chi inviare le update ricevute, Telegram utilizza il meccanismo del Webhook. Uno Webhook è appunto un indirizzo, indicato nella forma di una URL, che specifica a chi devono essere destinati i messaggi inviati dagli utenti al bot. Nel nostro caso, lo Webhook sarà indicato come un programma risiedente sul server HMR.
Per impostare un certo Webhook relativo a un bot, si utilizza il metodo setWebhook.

Questo metodo appare da subito più conveniente, in quanto non c'è bisogno di dover richiedere nuove update periodicamente, ma queste saranno inviate in automatico soltanto al momento di un nuovo messaggio di un utente. Tuttavia, per poter instaurare questo tipo di comunicazione tra Telegram e il server sul quale risiede il programma indicato come Webhook, sono richieste una serie di caratteristiche da parte del server e in particolare è necessario un server che:

- Supporti IPv4
- Accetti richieste POST da 149.154.167.197-233 sulle porte 443,80,88 o 8443
- Sia in grado di gestire traffico TLS1.0+HTTPS
- Sia provvisto di un certificato autofirmato o verificato
- Usi un CN o SAN che corrisponda al dominio indicato durante il setup
- Fornisca tutti i certificati intermedi per completare i processi di verifica

2. Natural Language Processing

Il Natural Language Processing (NLP), o Trattamento Automatico del Linguaggio (TAL) è un campo della Linguistica Computazionale che riguarda l'interazione tra il computer e i linguaggi naturali.

Tradizionalmente i computer richiedono che si "parli" con loro tramite un linguaggio di programmazione, quindi un modo di comunicare che è preciso, non ambiguo e altamente strutturato, utilizzando un numero finito di comandi noti. Il linguaggio umano, al contrario, non è preciso, è spesso ambiguo, e la struttura linguistica può dipendere da molte diverse variabili, come i dialetti e i vari contesti sociali. (Natural Language Processing, sito web)

Per questo motivo, NLP è un settore estremamente importante, in quanto studia e cerca di risolvere tutte le difficoltà che incontra il computer nel momento in cui deve interpretare, o analizzare il linguaggio umano.

L'argomento è particolarmente interessante all'interno di questa tesi, in quanto nel nostro caso è il bot che dovrà interpretare dei testi in italiano per costruire la sua conoscenza della storia dell'informatica con cui conversare con gli utenti.

In questo capitolo verrà esposta brevemente la storia del NLP e le sue caratteristiche (par. 2.1), per poi analizzarne strumenti (par. 2.2) e limiti (par. 2.3) e infine valutare in che modo può essere utile agli obiettivi di tesi (par. 2.4).

2.1 Breve storia del NLP e caratteristiche generali

Anche la nascita del NLP, nella sua accezione generale, può essere fatta risalire al 1950 con la pubblicazione dell'articolo "Computing Machinery and Intelligence" (Turing, 1950) di Alan Turing.

Il settore del NLP trova una sua identità compiuta alla fine degli anni '80, con l'avvento degli algoritmi di machine learning. Il sistema di interpretazione che venne introdotto era basato su modelli statistici e decisioni probabilistiche, che tenevano conto del diverso peso assegnato ai dati di input.

Molti dei primi successi, dovuti soprattutto al lavoro di IBM Research (Della Pietra, 1994), si conseguirono nel campo della traduzione automatica, grazie alla disponibilità di numerosi corpora già esistenti. La maggior parte degli altri sistemi, tuttavia, eseguivano i task richiesti utilizzando corpora creati ad hoc, ottenendo, così, risultati applicabili solo a contesti specifici.

Le varie tecniche di apprendimento automatico si sono man mano divise in due categorie principali:

- apprendimento supervisionato, o *supervised learning*;
- apprendimento non supervisionato, o *unsupervised learning*.

Il funzionamento generale degli algoritmi di apprendimento supervisionato è il seguente:

1. vengono definiti dei dati di input;
2. per ogni input viene indicato manualmente il corrispettivo output;
3. l'insieme di input/output (*training set*) viene fornito all'algoritmo;
4. l'algoritmo analizza i criteri per cui ad ogni input viene fatto corrispondere un output e crea un *modello della lingua*;
5. utilizzando tale modello, è in grado di applicare a nuovi input delle interpretazioni che permettono di restituire degli output.

Questa tecnica si basa sull'idea generale che a ingressi simili corrispondano uscite simili, il che non è sempre vero nel mondo reale. In ogni caso, l'efficacia di tale algoritmo è strettamente dipendente dal numero di dati forniti come *training set*: un numero di dati molto ristretto renderebbe l'algoritmo poco efficiente in quanto non possiederebbe una sufficiente esperienza, mentre un numero troppo elevato di dati finirebbe per rendere l'algoritmo eccessivamente complicato e quindi lento.

Attualmente la ricerca si sta concentrando sempre più su algoritmi di apprendimento non supervisionato, ovvero in grado di imparare da dati non annotati o corretti manualmente.
Il funzionamento generale degli algoritmi di apprendimento non supervisionato è il seguente:

1. vengono forniti all'algoritmo dei dati di input;
2. l'algoritmo analizza i dati;
3. i dati vengono classificati sulla base di caratteristiche comuni.

Nell'apprendimento non supervisionato, al contrario di quello supervisionato, vengono forniti soltanto *training set* non annotati, e viene lasciata all'algoritmo la scelta di classificare i dati secondo vari criteri che si basano su similarità e differenze.
L'utilizzo di questo tipo di tecniche ha portato, negli ultimi anni, a notevoli miglioramenti e preziosi risultati (Vinyals, 2015) (Goldberg, 2016) in diversi ambiti del NLP.

L'applicazione del NLP ai bot, da Eliza a Jabberwacky, fa sempre parte di questa storia ed è stata già discussa al par. 1.1.

2.2 Strumenti del NLP

NLP è utilizzato per la risoluzione di numerosi problemi legati all'elaborazione del linguaggio naturale e per riuscirci fa uso di diversi metodi e strumenti. Tra i principali troviamo (Lenci, Montemagni e Pirrelli, 2005):

- part of speech (POS) tagging – individuazione della parte del discorso a cui appartiene ogni parola;
- machine translation – traduzione automatica di un testo da un certo linguaggio naturale ad un altro;
- named entity recognition – classificazione dei nomi propri in categorie (persona, luogo, organizzazione, ecc.);
- question answering – individuazione di risposte a specifiche domande poste in linguaggio naturale;
- sentiment analysis – estrazione di informazione soggettiva da un testo per individuarne la polarità, ossia la posizione in cui l'autore del messaggio si pone rispetto all'argomento di cui sta parlando;
- topic recognition – individuazione dell'argomento principale di cui tratta un certo testo.

Alcuni degli strumenti elencati – come il machine translation – sono legati direttamente a situazioni del mondo reale, mentre altri – come il POS tagging – servono per risolvere sotto-problemi relativi a task più grandi.
In ogni caso, tutti i metodi e gli strumenti di sono tutti strettamente collegati tra loro.

2.3 Limiti del NLP

L'elaborazione automatica del linguaggio naturale è resa particolarmente complicata da alcune caratteristiche di quest'ultimo.
La difficoltà maggiore sta nel fatto che spesso le espressioni linguistiche possono avere più di un'interpretazione, creando così delle *ambiguità*.
Le ambiguità esistono a tutti i livelli di analisi:

- ambiguità fonologiche:
  una sequenza di suoni può corrispondere a parole diverse, per esempio il suono "/lama/" può corrispondere a
  - "lama", o
  - "l'ama";

- ambiguità morfologiche e morfosintattiche:
  le parole possono essere diversamente scomposte in morfemi, per esempio la parola "portale" può corrispondere a
  - "porta-le" (verbo + pronome clitico), o
  - "portal-e" (sostantivo);

- ambiguità sintattiche:
  a un'espressione linguistica possono corrispondere diverse analisi sintattiche, per esempio la frase "ha chiamato Gianni" può essere analizzata come
  - [[ha chiamato] [$_{sogg}$ Gianni]], o
  - [[ha chiamato] [$_{ogg}$ Gianni]];

- ambiguità semantiche:
  un'espressione linguistica può avere più di un significato, per esempio la parola "navigare" può voler dire
  - andare per mare, o
  - visitare pagine web su Internet;

- ambiguità pragmatiche:
  un'espressione linguistica può essere usata per diversi scopi comunicativi, per esempio la frase "Hai la macchina?" può essere intesa come
  - domanda, o
  - richiesta di un passaggio.

Oltre alle ambiguità, il linguaggio umano presenta varie altre complessità, è privo di una forte regolarità e ricco invece di sottintesi, modi di dire, metafore, e in generale informazioni che richiedono interpretazioni extra-testuali.

Per via soprattutto di queste difficoltà, non esistono ancora software in grado di riconoscere e interpretare un qualunque testo con la stessa media di successo che potrebbe avere una persona, tuttavia nel campo del NLP si stanno raggiungendo risultati sempre più soddisfacenti, per esempio per quanto riguarda il *language modeling* (Jozefowicz, Vinyals, Schuster, Shazeer, Wu, 2016), o il *parsing* (Choe, Charniak, 2016).

Un altro limite, sentito in particolare dal nostro progetto, è la lingua italiana: la maggior parte della ricerca nell'elaborazione del linguaggio naturale si concentra sulla lingua inglese. Per l'italiano, se pure siano stati ottenuti successi e siano in sviluppo diversi progetti, gli strumenti non sono ancora al livello di quelli disponibili per l'inglese. Alcuni tra i principali progetti attualmente attivi sono *Tint* e *Tanl*, che sono stati discussi nel par. 1.2.6.

2.4 Utilizzo del NLP nella tesi

All'interno del progetto di tesi l'NLP serve come strumento per analizzare testi da cui estrarre determinate informazioni, che verranno poi riproposte agli utenti durante le conversazioni con il bot.
Questi testi sono disponibili in un database e, grazie agli strumenti di NLP, saranno processati per estrarre e strutturare le informazioni contenute al loro interno, in modo che possano essere utilizzate convenientemente dal programma del bot.

I modi in cui il chatbot utilizzerà le informazioni ottenute con le analisi NLP possono essere diversi e in particolare possiamo distinguere 3 scenari d'uso delle informazioni ordinati per un livello di sofisticazione crescente:

- strutturazione in una serie di eventi o fatti in modo che siano restituibili tramite un messaggio di risposta; questi fatti saranno selezionabili dall'utente tramite scelte preimpostate, o saranno scelti in automatico dal bot;
- strutturazione delle informazioni in una serie di domande e risposte, in modo che il bot possa porre le domande agli utenti sotto forma di quiz e verificare la correttezza della risposta, oppure possa dare una certa risposta all'utente nel caso in cui ritenga che il messaggio da esso inviato corrisponda con una delle domande;
- strutturazione delle informazioni in una serie di soggetti della storia dell'informatica collegati a eventi che li coinvolgono, in modo che se l'utente chiede informazioni su un certo soggetto, il bot sia in grado di fornirgli il corrispettivo evento.

Nello studio di fattibilità (cap. 3) saranno analizzate queste possibilità, per valutare la modalità in cui andranno poi messe in pratica nel corso della realizzazione.

3. Studio di fattibilità

Dopo aver esposto le varie tecniche di Intelligenza Artificiale usate dai chatbot (cap. 1.2) e aver scelto la piattaforma più adatta (cap. 1.3), è il momento di porre degli obiettivi che andranno poi raggiunti nella fase di realizzazione.
Lo scopo di questo capitolo è quello stabilire ciò che andrà sviluppato, restando in linea con l'idea generale del progetto e rientrando nelle risorse tecniche e temporali disponibili.

3.1 L'idea

Il progetto consiste nella realizzazione di un chatbot con il quale gli utenti possano intrattenere una conversazione, testuale o tramite opzioni preimpostate, per soddisfare le loro curiosità sulla Storia dell'Informatica e sul progetto HMR.
Tra le funzionalità aggiuntive, il chatbot potrebbe avere una modalità *quiz* nella quale porre delle domande agli utenti che potrebbero scegliere tra una serie di risposte preimpostate, oppure sostenere una conversazione del tipo di Eliza in cui potrebbe cercare di portare l'argomento su fatti noti della Storia dell'Informatica.

3.2 Metodi di realizzazione

Per quanto riguarda la piattaforma sulla quale il bot sarà attivo, la scelta è già stata presa nel cap. 1.3 ed è ricaduta su Telegram.
Nel cap. 1.2 invece, sono state esposte le varie tecniche utilizzate per dotare i chatbot di intelligenza e adesso deve essere trovato il modo di usarle opportunamente ai nostri scopi.

È doveroso differenziare due diversi tipi di lavoro in cui si dividerà il progetto: un lavoro di analisi degli input degli utenti per formulare una risposta (par. 3.2.1) e uno di trattamento delle informazioni che dovranno costituire la conoscenza del bot e successiva strutturazione in modo che possano essere usate al meglio (par. 3.2.2).

3.2.1 Analisi dei messaggi di input

Innanzitutto è necessario ripulire il testo di input dell'utente da alcuni tipi di rumore e per far questo si utilizzeranno le tecniche di *preparazione del testo* (par. 1.2.1). In particolare andranno eliminati eventuali spazi a inizio e fine messaggio e andranno normalizzate le maiuscole.

Dopodiché dovremo individuare le informazioni contenute all'interno dell'input per le quali si intende restituire una certa risposta. A questo scopo può tornare utile la tecnica del *pattern matching* (par. 1.2.2): con le espressioni regolari sarà possibile

trovare uno specifico insieme di parole (o radici) se riconosciuto all'interno del messaggio.

La *trasposizione di parole* (par. 1.2.3) ci è utile solo marginalmente, ma può essere usata per aumentare la parvenza di comprensione dei messaggi da parte del bot, alla maniera di Eliza e bot simili.

Sempre per migliorare il livello della conversazione di base, ma anche per poter parlare di un certo argomento senza che venga dimenticato quale esso fosse useremo la *memorizzazione del contesto* (par. 1.2.4). Per fare questo avremo bisogno di inserire informazioni in un database, in modo da poterla sfruttare se fosse richiesto.

Una volta che saremo in grado di riconoscere il concetto espresso dall'input, si pone il problema del caso in cui sia più di un concetto ad essere riconosciuto. Grazie al *ranking di possibili risposte* (par. 1.2.5) potremo decidere tra i vari riconoscimenti, quale sia più rilevante e rispondere solo a questo.

Infine, le tecniche di Natural Language Processing possono essere molto utili ai fini di dare un'intelligenza al bot. Sapere qual è la struttura grammaticale e logica di una frase può aiutare a trovare una risposta accettabile anche se il contenuto del messaggio non è riconosciuto da nessun pattern.

3.2.2 Trattamento dei dati, "pre-masticazione" di testi esistenti

Le informazioni che andranno a formare la competenza del bot sono interamente contenute, in forma di testi, in due distinti database:

- oggiSTI - contiene gli eventi della storia dell'Informatica accaduti in specifiche date nel passato, completi di titolo in italiano e in inglese, immagine, una descrizione completa e una riassuntiva (non ripetitive).
- EPICAC - contiene informazioni relative al progetto HMR, dalla sua storia a eventi recenti, con titolo, descrizione e data.

L'uso di NLP sarà fondamentale per poter analizzare questi database dai quali ricavare un nuovo database "pre-masticando" le informazioni, ossia strutturandole nel modo migliore perché il chatbot le possa utilizzare.

Tra le opzioni elencate nel par. 2.4 relative al modo migliore in cui strutturare i dati, è stata scelta quella in cui dai testi viene estratta una serie di eventi o fatti e per ognuno di questi viene indicata la data, il soggetto, il verbo e i vari complementi. In questo modo il bot può utilizzare queste informazioni sia per restituire informazioni su un certo evento quando richiesto dall'utente durante la conversazione, sia per funzioni specifiche come per esempio il quiz.

Tra i due database oggiSTI e EPICAC, per sfruttare al meglio l'impegno ragionevole per un lavoro di tesi, è stato scelto di concentrare questo lavoro sul primo, mentre l'implementazione del secondo viene lasciata aperta a sviluppi futuri.

3.3 Utilizzo di risorse

3.3.1 Risorse necessarie

Per poter essere funzionante, un chatbot Telegram ha bisogno di un server sul quale deve risiedere il programma che elabora i messaggi. Su tale server deve essere installato un certificato SSL in modo da poter avere uno scambio di informazioni sicuro.
È inoltre necessario avere a disposizione una qualche base di dati, fondamentale per poter dotare il bot di memoria, per avere una lista degli utenti da poter usare per diversi scopi (per esempio l'invio di messaggi di broadcast), oltre che ovviamente per poter immagazzinare tutta l'informazione che il bot deve avere a disposizione per elaborare le risposte.

3.3.2 Risorse a disposizione

Con il progetto HMR abbiamo a disposizione un server Web Apache che ospita il dominio http://progettohmr.it.
Il server offre l'utilizzo di database MySQL ed ha installati i linguaggi di programmazione PHP v5.6, Python v2.6.6, Ruby v1.8.7 e Perl v5.10.1[2].
C'è inoltre la possibilità di installare librerie aggiuntive, tra le quali potrebbe esserci utile NLTK. Si tratta di un insieme di librerie e programmi scritti in Python, creati per supportare il trattamento del linguaggio naturale (Natural Language Processing, sito web).
È inoltre presente un certificato SSL installato sul dominio, con rinnovo automatico.

3.3.3 Vincoli sulle risorse disponibili, sicurezza

Un vincolo importante sta nel fatto che il nostro server Web possiede un modulo chiamato ModSecurity che filtra le informazioni scambiate limitando alcuni tipi di comunicazione (ModSecurity, sito web). Tra queste c'è anche la comunicazione con Telegram e per questo motivo non è possibile rendere funzionante il bot con ModSecurity attivato.
Per risolvere questo problema abbiamo creato un sottodominio http://bot.progettohmr.it nel quale saranno collocate solamente le risorse relative al bot, disattivando ModSecurity solo su questo sottodominio; in questo modo tutto il resto delle risorse non sarà soggetto a rischi di sicurezza.

2 Versioni dei linguaggi verificate il giorno 25/07/2017.

Per lo stesso motivo non è conveniente che il bot acceda al database contenente i testi ed è quindi necessario copiare i contenuti di questi in un altro database al quale accederà il bot.
In questo modo si evita che le credenziali di accesso ai database principali siano presenti nel codice del bot e che siano in questo modo soggette ai rischi dovuti alla mancanza della protezione ModSecurity.
Questa architettura, inoltre, si sposa bene con la fase di pre-masticazione di cui è stato detto nel par. 3.2.2.

4. Requisiti

4.1 Introduzione al capitolo

4.1.1 Scopo del capitolo

Lo scopo di questo capitolo è quello di descrivere, in linea generale, le caratteristiche che dovrà avere il chatbot a progetto concluso, ai fini di creare una linea guida da seguire in fase di realizzazione. Il capitolo è stato redatto seguendo le indicazioni del documento “IEEE Recommended Practice for Software Requirements Specifications” (IEEE Std 830-1993).

4.1.2 Obiettivi del progetto

L’obiettivo da raggiungere è quello di realizzare un chatbot con il quale sia possibile conversare tramite la piattaforma di messaggistica istantanea Telegram (Telegram Messenger, sito web) e che sia in grado di rispondere a curiosità sulla storia dell’Informatica e, marginalmente, sul progetto HMR[3].

4.1.3 Parole chiave

Di seguito viene indicata una lista di parole usate spesso nel documento con relativa descrizione, per chiarificare a cosa viene fatto riferimento quando vengono usate.

Termine	**Descrizione**
update	È un oggetto che rappresenta un messaggio inviato da un utente a un bot Telegram.
getUpdates	Metodo utilizzato per ricevere le update di un bot Telegram.
sendMessage	Metodo utilizzato per inviare un messaggio da parte di un bot Telegram verso un certo utente.
Webhook	Indirizzo web a cui Telegram invia le update ricevute da un bot.
setWebhook	Metodo Telegram utilizzato per impostare un certo Webhook.
pull	Un processo in cui l’attività a monte spinge quella a valle
push	Un processo in cui l’attività a valle trascina quella a monte

3 Hackerando la Macchina Ridotta è un progetto di ricerca in storia dell'informatica, il suo obiettivo è recuperare e raccontare le storie e le tecnologie dei primi calcolatori – italiani in particolare, ma non solo. http://hmr.di.unipi.it/

4.1.4 Organizzazione del capitolo

La restante parte di questo capitolo contiene una descrizione generale del contesto in cui dobbiamo lavorare per la realizzazione del bot (cap. 4.2) e una parte dedicata alle funzionalità specifiche che dovrà avere (cap. 4.3).

4.2 Descrizione generale

Questa sezione contiene la descrizione del contesto in cui si deve inserire il nostro chatbot, quindi il modo in cui il chatbot dovrà interagire con il resto dell'infrastruttura del progetto HMR. Verranno discusse, inoltre, le altre risorse da realizzare che faranno da contorno al prodotto principale, come per esempio il pannello di controllo con le relative pagine e quella di informazione, contenente la descrizione del bot.

4.2.1 Collocazione del bot all'interno della struttura di HMR

Il programma che gestisce ricezione e invio dei messaggi del bot, risiederà sul server del progetto HMR, dalle caratteristiche descritte al par. 3.3.2. Le fonti di conoscenza che avrà il bot saranno costituite dai due database di cui è già stato accennato al par. 3.2.2, uno contenente informazioni sulla storia dell'Informatica, l'altro con informazioni sul progetto HMR.
Il bot, tuttavia, non accederà direttamente a questi database, per i motivi di sicurezza discussi al par. 3.3.3, ed è quindi necessario spostare le informazioni contenute nei due suddetti database in un terzo al quale il bot potrà accedere senza creare problemi di sicurezza. Queste informazioni non dovranno essere semplicemente copiate, ma andranno analizzate e ristrutturate in database pre-masticati nel modo più conveniente perché il bot le possa usare per i propri scopi.
La situazione generale sarà quella illustrata in Figura 8.

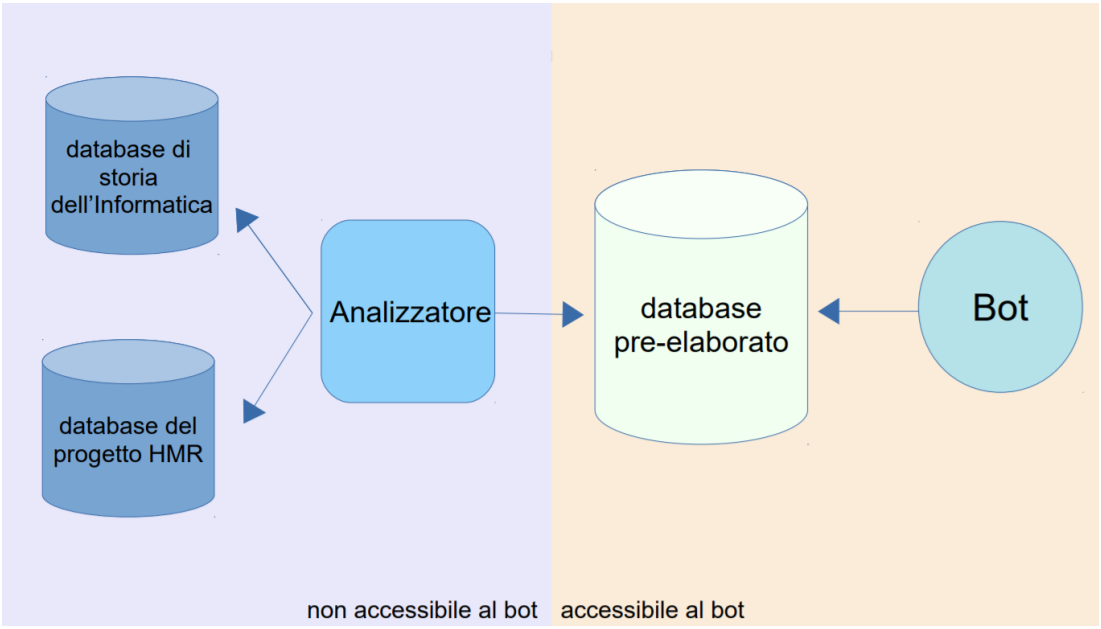

Figura 8: Preparazione dei dati che verrano usati dal bot

Le informazioni contenute nei due database originari saranno quindi analizzate e strutturate all'interno del database pre-elaborato (o pre-masticato), al quale il bot potrà accedere per ottenere le informazioni necessarie per rispondere ai messaggi degli utenti.

4.2.2 Componenti

4.2.2.1 Programma di analisi e pre-masticazione

Il componente che nella Figura 8 abbiamo indicato come "Analizzatore" avrà la funzione di analizzare le informazioni dei database sulla storia dell'Informatica e sul progetto HMR e, in seguito, di inserirle nel database di destionazione seguendo uno schema logico opportuno. Queste operazioni andranno eseguite in locale ed in modalità asincrona rispetto al funzionamento del bot, in modo che l'analisi degli input degli utenti e la costruzione delle successive risposte possa sempre contare su una bae di dati già coerente ed ottimizzata.
L'obiettivo di questa funzione è quella di facilitare le operazioni del bot nel momento in cui deve ottenere una certa informazione contenuta nei database. Avendo le informazioni strutturate sarà sufficiente cercare, per esempio, una data tra un elenco di date, oppure un nome tra un elenco di nomi per trovare l'informazione richiesta dall'utente, invece che cercarla all'interno di un insieme di testi.

In particolare, le informazioni che ci interessa strutturare saranno di tre tipi: personaggi, date e eventi.
La strutturazione finale che si vuole ottenere sarà quella in cui abbiamo una lista di personaggi, ognuno dei quali sarà associato a una serie di eventi con le relative date nelle quali gli eventi sono avvenuti.

4.2.2.2 Pannello di amministrazione del bot

Il pannello di amministrazione è contenuto all'interno del pannello di controllo di HMR al quale possono accedere solamente gli utenti registrati e, in particolare, solo quelli che possiedono i permessi di amministrazione del bot.
Tramite il pannello sarà possibile eseguire alcune operazioni sul bot, come per esempio la visualizzazione del log delle conversazioni tra il bot e gli utenti, oppure l'invio di un messaggio di broadcast, ossia un messaggio inviato a tutti gli utenti che abbiano iniziato una conversazione con il bot.

4.2.2.3 Pagina web del bot

La pagina web del bot sarà raggiungibile all'indirizzo http://bot.progettohmr.it che è il sottodominio nel quale sono presenti i file relativi al bot.
In questa pagina sarà inserito un collegamento con il sito principale del progetto HMR e una semplice guida su come iniziare una conversazione con il bot.

4.3 Requisiti specifici

4.3.1 Requisiti di interfaccia esterna

4.3.1.1 Interfaccia utente

Il bot deve essere accessibile da qualunque dispositivo che supporti la piattaforma Telegram, quindi ogni smartphone o tablet con sistema iOS, Android, o Windows Phone e ogni computer con sistema Linux, Windows, o OSX.

Deve possedere un nickname intuitivo in modo da poter essere facilmente individuabile.

Una volta iniziata la chat deve saper riconoscere il linguaggio naturale italiano per poter intrattenere conversazioni con ogni tipo di utente (vedi Figura 9).

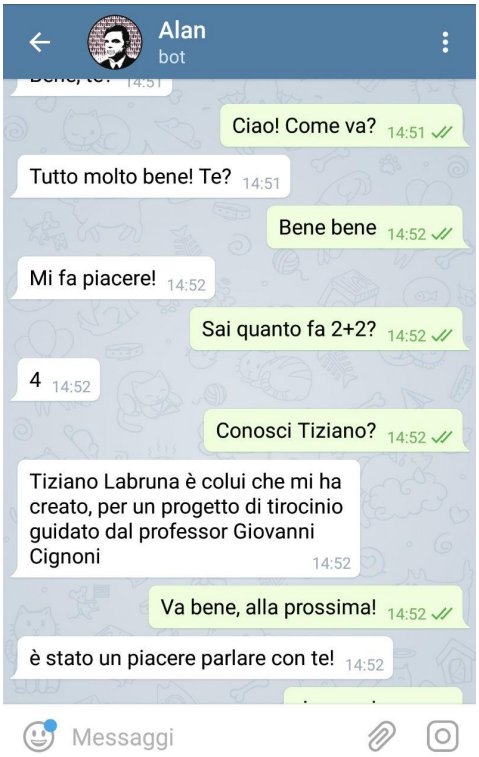

Figura 9: Esempio di chat con il bot

Il pannello di amministrazione deve avere un'interfaccia intuitiva, in modo da poter usare tutti gli strumenti disponibili nella maniera più semplice possibile.

La pagina web del bot deve essere semplice e intuitiva e deve permettere agli utenti di riuscire facilmente a comunicare con il bot.

4.3.1.2 Interfaccia hardware

Il progetto non ha bisogno di gestire le interazioni con nessun tipo di hardware.

4.3.1.3 Interfaccia software

Il bot dovrà possedere la conoscenza contenuta nei database del progetto HMR e per questo dovrà poter accedere a questi database.
Il pannello di amministrazione è presente all'interno del pannello di controllo HMR e condivide la gestione degli utenti con il sistema generale.
Al pannello di amministrazione del bot accede un solo utente per volta per evitare che ci siano collisioni sui dati. La multiutenza non è necessaria.

Il pannello di amministrazione del bot deve poter essere utilizzato solamente dagli utenti registrati e che possiedano i permessi richiesti. Le password degli utenti devono essere criptate nel database.

4.3.1.4 Interfaccia di comunicazione

Il bot dovrà condividere il protocollo di comunicazione di Telegram per poter ricevere le update e inviare i messaggi.

4.3.2 Requisiti funzionali

4.3.2.1 Funzionalità lato utente

id: RF1
Titolo: Rispondere a domande sulla storia dell'informatica
Descrizione: Il bot deve riuscire a dare delle risposte a richieste degli utenti relative a informazioni sulla storia dell'Informatica e sul progetto HMR. Questa funzionalità generale è suddivisa in due sotto-funzioni elencate di seguito.

id: RF1.1
Titolo: Soggetti dell'Informatica, personaggi, aziende, prodotti
Descrizione: L'utente deve poter chiedere informazioni su un certo evento della storia dell'Informatica o su un certo fatto relativo al progetto HMR e il bot deve potergli fornire l'informazione richiesta se ritiene che l'input dell'utente corrisponda a un qualche elemento presente nel database.

id: RF1.1.1
Titolo: Soggetti dell'Informatica tramite risposte preimpostate
Descrizione: L'utente deve poter scegliere tra alcune risposte preimpostate per ottenere risultati su un certo soggetto della storia dell'Informatica e il bot deve saper fornire una risposta.

id: RF1.2
Titolo: Date e eventi

Descrizione: L'utente deve poter chiedere al bot in che data è successo un certo evento della storia dell'Informatica, oppure cosa è successo in una certa data; in entrambi i casi, se l'evento o la data sono presenti nel database del bot, quest'ultimo deve sapergli fornire l'informazione richiesta.

id: RF1.2.1
Titolo: Date e eventi tramite risposte preimpostate
Descrizione: L'utente deve poter scegliere tra alcune risposte preimpostate per ottenere risultati su eventi successi in specifiche date.

id: RF1.2.2
Titolo: Date e eventi: cos'è successo oggi
Descrizione: L'utente deve poter ottenere informazioni su eventi accaduti nel passato nello stesso giorno dell'anno rispetto alla data odierna.

id: RF2
Titolo: Sottoporre delle domande sulla storia dell'Informatica in stile quiz
Descrizione: Il bot deve avere una modalità quiz, nella quale sottoporrà all'utente delle domande sulla storia dell'Informatica, relativamente alle quali l'utente potrà scegliere tra una serie di risposte preimpostate.

id: RF2.1
Titolo: quiz sai chi
Descrizione: Il bot sottoporrà all'utente delle domande sulla storia dell'Informatica, relativamente alle quali l'utente potrà scegliere tra 4 risposte preimpostate di cui una sola giusta. Questa funzione deve essere accessibile tramite i comandi.

id: RF2.2
Titolo: quiz sai quando
Descrizione: Il bot presenterà all'utente dei fatti sulla storia dell'informatica, domandandogli in quale anno è accaduto lo specifico fatto. L'utente dovrà rispondere indicando una data e il bot indicherà se è la data corretta, se è anteriore o se è posteriore. Questa funzione deve essere accessibile tramite i comandi.

id: RF3
Titolo: Sostenere una conversazione base
Descrizione: Il bot deve essere in grado di sostenere una conversazione di base nella quale l'utente sarà spinto ad entrare nel campo di competenza primaria del bot.

id: RF3.1
Titolo: Evitare riferimenti offensivi e atteggiamenti negativi
Descrizione: Durante le conversazioni con gli utenti il bot non deve rispondere in modo che possa risultare offensivo per gli utenti e deve cercare di mantenere un atteggiamento positivo.

id: RF3.2
Titolo: Ricordarsi informazioni dette in precedenza
Descrizione: Il bot deve potersi ricordare informazioni già dette dall'utente, per rendere la conversazione meno automatizzata e deve saper rispondere cose diverse a seconda del contesto conversativo.

id: RF3.3
Titolo: Indirizzare la conversazione sul campo di competenza
Descrizione: Nel momento in cui un certo messaggio non viene riconosciuto, il bot deve cercare di indirizzare l'utente su argomenti di sicura conoscenza, consigliandogli l'invio di possibili messaggi per i quali sa dare una risposta.

id: RF4
Titolo: Inserire comandi Telegram facilmente accessibili
Descrizione: Devono essere presenti dei comandi che l'utente possa richiamare con facilità e che attivino specifiche funzioni del bot.

4.3.2.2 Funzionalità lato amministratore

id: RF5
Titolo: Aggiornarsi automaticamente alla modifica del database
Descrizione: Il bot deve essere in grado di modificare automaticamente il modo in cui interagisce con l'utente nel momento in cui viene aggiornato il database, senza bisogno che l'amministratore modifichi il programma del bot.

id: RF6
Titolo: Visualizzare il log delle conversazioni
Descrizione: L'amministratore deve essere in grado di visualizzare la cronologia delle conversazioni tramite un'interfaccia intuitiva.

id: RF7
Titolo: Inviare un messaggio in broadcast
Descrizione: L'amministratore deve poter inviare un messaggio a tutti gli utenti del bot in modo semplice e intuitivo.

id: RF8
Titolo: Ottenere giudizi dagli utenti sulle risposte date
Descrizione: Chiedere dei giudizi di attinenza delle risposte date agli utenti da poter usare per analizzarne l'efficienza.

4.3.3 Requisiti di sicurezza

Al Pannello di Controllo del bot potrà accedere soltanto l'utente con permessi di amministrazione.
Il login al Pannello di Controllo dell'amministratore del bot è collegato al sistema di autenticazione unica di HMR, condivisa con i progetti di oggiSTI (Pratelli, 2017) e EPICAC (Lenzi, 2017).

Un'accortezza importante da dover prendere legata alla sicurezza riguarda il fatto che, per i motivi spiegati al par. 3.3.3, il sotto-dominio bot.progettohmr.it è più vulnerabile rispetto al resto dei file presenti sul server del progetto ed è conveniente che acceda ad un database specifico per le sue funzioni, separato da quelli di oggiSTI ed EPICAC.
Lo scambio di informazioni tra il bot e i database dovrà seguire lo schema descritto in Figura 10.

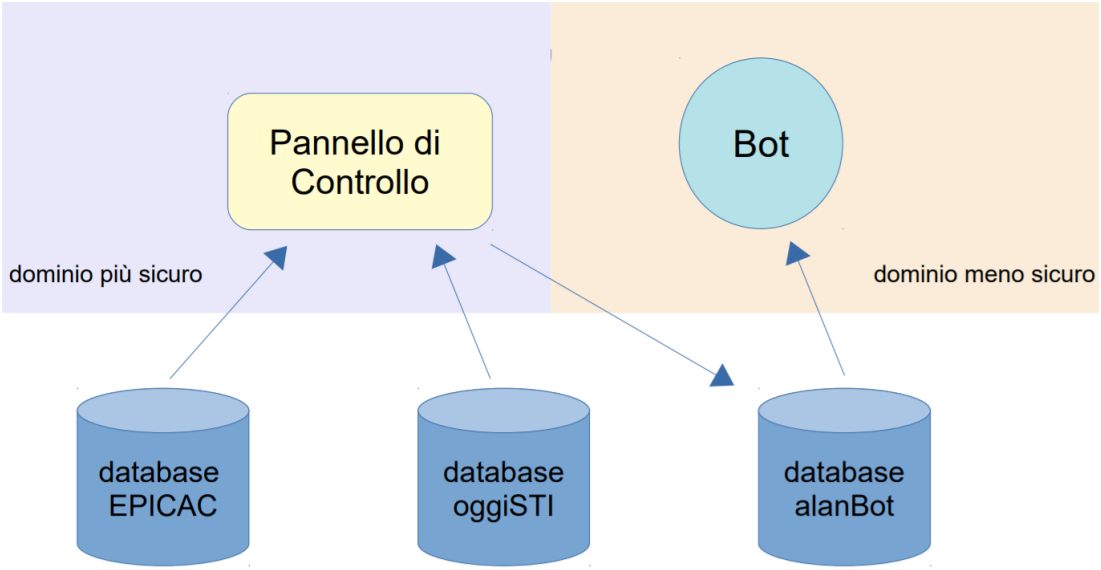

Figura 10: Comunicazione tra bot e database

Le informazioni presenti nei database *oggiSTI* e *EPICAC* verranno, innanzitutto, copiate e rese accessibili nel Pannello di Controllo.
Sempre dal Pannelo di Controllo, nella sezione dell'amministrazione del bot, sarà possibile avviare la funzione di pre-masticazione dei dati di *oggiSTI* per inserirli nel database del bot.
A questo punto il bot avrà la possibilità di accedere a queste informazioni senza intaccare la sicurezza degli altri database.

4.3.4 Requisiti di prestazione

Il progetto non richiede nessun particolare requisito di prestazione.
I tempi di risposta del bot sono comunque più che soddisfacenti, anche nel caso in cui deve ricercare informazioni nel database.

Le attività di NLP sul database *oggiSTI* sono fatte fuori linea e dipendono da un'applicazione esterna (vedi par. 1.2.6.4).
In ogni caso, sulle prove fatte, i tempi sono più che accettabili, intorno a una decina di secondi.

4.3.5 Vincoli di progetto

Il bot deve basare la sua conoscenza sulle informazioni già presenti nel contesto del progetto HMR e quindi in particolare i due database di cui abbiamo parlato.
L’aspetto del pannello di amministrazione e della pagina web del bot devono essere coerenti con quello del sito web di HMR.

5. Realizzazione

5.1 Creazione del bot

Per creare un bot Telegram è necessario interfacciarsi proprio con uno specifico bot Telegram: BotFather[4]. Durante la creazione viene richiesto di fornire il nome del bot e il nickname, mentre in seguito è possibile aggiungere, sempre tramite BotFather, altre informazioni come ad esempio i comandi, la descrizione, l'immagine del profilo e la bio.

5.1.1 Nome

Come nome per il bot è stato scelto Alan.
La scelta di questo nome è ricollegabile alla figura di Alan Turing, personaggio che nell'immaginario collettivo ha il ruolo del padre dell'informatica, nonché inventore del primo computer; nonostante siano informazioni non del tutto veritiere storicamente, resta comunque un soggetto di importanza indiscussa nella storia dell'informatica, in particolare perché propositore del test di Turing, che si basa su una conversazione in linguaggio naturale con un programma.

5.1.2 Nickname

Come nickname per il bot è stato scelto @ProgettoHMRBot.
A differenza del nome che deve essere familiare e mnemonico per poter essere facilmente trovato tra le chat attive, il nickname, nella terminologia Telegram, viene usato una volta soltanto, per poter iniziare la prima sessione di chat.
Per questo motivo la scelta del nickname è ricaduta su un nome che sia un riferimento diretto al Progetto HMR, che è il più probabile punto di accesso al bot.

5.1.3 Comandi

I comandi sono dei messaggi (che hanno come carattere iniziale il simbolo "/") che l'utente può inviare al bot, il quale risponderà nella maniera indicata dalla corrispondente descrizione.
Questa è la lista dei comandi che, ad oggi[5], possiede il bot:

- trova_evento - Trova un evento in base alla data
- evento_random - Restituisce un evento casuale
- trova_soggetto - Trova un evento in base a un argomento
- accadde_oggi – Scopri cos'è successo oggi nella Storia dell'Informatica
- quiz_sai_chi - Prova la tua conoscenza dei protagonisti della Storia dell'Informatica!
- quiz_sai_quando - Una sfida sulla cronologia dell'Informatica

4 È possibile iniziare una chat con BotFather seguendo il seguente link: https://telegram.me/BotFather.
5 Relativamente alla data 30/10/2017

La scelta dei nomi dei comandi è stata dettata da alcuni vincoli sintattici che pone Telegram, tra cui in particolare l'impossibilità di inserire spazi.

5.1.4 Descrizione

La descrizione è il messaggio che appare all'inizio della prima conversazione con il bot. È stata scelta una descrizione che indichi il collegamento tra il bot e il Progetto HMR e che spieghi cosa è possibile fare con il bot.
Inoltre sono stati inseriti alcuni esempi di azioni che può compiere un utente e l'indirizzazione al comando "/help" nel caso in cui l'utente desideri avere la lista completa delle operazioni eseguibili dal bot.

5.1.5 Immagine del profilo

L'immagine è una rappresentazione della più famosa foto di Alan Turing a cui sono stati applicati elementi tipici di un robot, per dare l'idea del ruolo di assistente automatizzato del bot. Sullo sfondo è stato aggiunto il logo di HMR ripetuto in modo da coprire l'intera area superiore.

5.1.6 Bio

La bio (o "about" nella versione in inglese) è una breve sezione visibile accedendo alle informazioni del profilo del bot. È stato inserito un breve messaggio che spiega agli utenti qual è lo scopo del bot e a quale progetto è collegato.

5.2 Inizializzazione del bot

Come è stato esposto nel par. 1.3.6, esistono due metodi in cui vengono gestiti i messaggi nei bot Telegram: getUpdates e setWebhook.
Per questo progetto è stato scelto il metodo setWebhook, in quanto garantisce una gestione automatica dei messaggi, senza necessità di analizzare manualmente gli input per successivamente inviare gli output.
Il metodo setWebhook richiede che venga indicato uno Webhook, ossia un file (presente su un server che deve essere in grado di gestire il traffico HTTPS) al quale vengono inviati tutti i messaggi ricevuti dal bot, in modo da poter mandare le relative risposte.
Come Webhook è stato impostato un file PHP, presente sul server del Progetto HMR.
Per riuscire a gestire i messaggi ed a effettuare tutte le altre operazioni relative al comportamento del bot, è stata seguita la documentazione fornita da Telegram (Telegram Bot API, sito web).

5.3 Aggiunta della funzionalità di conversazione base

Pur non essendo lo scopo principale del progetto, è importante che il bot dia all'utente una parvenza di intelligenza e che si avvicini, per quanto possibile, a simulare una conversazione umana.
Per fare questo, sono state messe in pratica alcune delle tecniche descritte al par. 1.2.

1. All'invio di un messaggio da parte dell'utente, questo viene ripulito da eventuali rumori e i caratteri vengono normalizzati al minuscolo.
2. Collegandosi a un database, diventano disponibili le variabili di sessione utili a far ricordare al bot cose dette in precedenza (per esempio il nome dell'utente).
3. Il testo viene confrontato con una serie di espressioni regolari, in modo da poter verificare se corrisponde ad un certo schema di messaggio (o a più di uno).
4. Per ogni corrispondenza che viene trovata, viene salvata una potenziale risposta a cui si assegna un certo ranking di pertinenza.
5. La risposta con ranking più alto viene inviata come output all'utente.
6. Se non viene trovata nessuna corrispondenza, viene consigliato all'utente di spostare il discorso sul campo delle competenze del bot, proponendo alcuni esempi di messaggi riconosciuti.

Il punto 1 viene gestito all'interno del file *operazioniPreliminari.php*.
Il punto 2 viene gestito all'interno del file *connessioneDB.php*.
I punti 3-4 vengono gestiti all'interno del file *conversazioneBase.php*.
I punti 5-6 vengono gestiti all'interno del file *invioRisposta.php*.

La capacità del bot di fare conversazione non può coprire qualunque tipo di argomento o stile di messaggio, ma osservando le conversazioni degli utenti durante i periodi di prove, è stato possibile implementare delle risposte per buona parte dei messaggi più comuni che un utente medio si trova ad inviare al bot, con risultati accettabili. La soluzione adottata, benché non particolarmente sofisticata, è comunque migliorabile, senza stravolgerne l'impianto, aumentando l'insieme delle frasi base; il codice è strutturato per facilitare questo tipo di operazioni.

5.4 Creazione del database pre-masticato

Come specificato nei requisiti, al par. 4.2.2.1, avremo bisogno di dare una struttura ai dati contenuti in *oggiSTI*, così che il bot possa reperire le informazioni utili a fornire risposte agli utenti in modo veloce e ordinato.

5.4.1 Strumenti e strutture utilizzati

Per estrarre le parti del discorso dalle frasi, per poi strutturarle, useremo alcuni strumenti di NLP.
In particolare ci serviremo di Tanl (par. 1.2.6.4), un servizio Web di analisi NLP sviluppato dal professor G. Attardi.
Per poter utilizzare i servizi di Tanl è necessario fornire, al momento della richiesta, un token Google, a scadenza periodica, che deve essere reperito manualmente. Per questo motivo, non è possibile eseguire la pre-strutturazione in maniera automatica, ma è necessario avviarla tramite Pannello di Controllo, fornendo il token al momento dell'avvio.

La struttura delle tabelle all'interno del database coinvolte nel processo di pre-masticazione dei dati è la seguente:

- alanbot – *tabelle relative al bot*
  - accadimenti – *tabella copiata da hmrepicac*
  - eventi – *tabella copiata da oggisti*
  - eventipremast – *dati pre-strutturati relativi alla tabella eventi*
  - personepremast – *dati pre-strutturati contenenti Named Entity*
- hmrepicac – *tabelle relative a EPICAC*
  - accadimenti
- oggisti – *tabelle relative a oggiSTI*
  - eventi

Le informazioni di *hmrepicac* e *oggisti* vengono copiate in *alanbot*, come indicato nei requisiti di sicurezza al par. 4.3.3. Al momento in cui le informazioni vengono copiate, l'accesso viene fatto dal dominio principale.

5.4.2 Analisi del processo di pre-masticazione

I passaggi che vengono eseguiti per effettuare la strutturazione dei dati sono i seguenti:

1. la tabella *accadimenti* di *hmrepicac* e la tabella *eventi* di *oggisti* vengono confrontate rispettivamente con *accadimenti* e *eventi* di *alanbot*;
   1.1. nel caso in cui le tabelle di *alanbot* fossero vuote, tutti i contenuti delle tabelle originali vengono copiati;
   1.2. nel caso in cui le tabelle di *alanbot* contengano le stesse informazioni delle tabelle originali, non viene eseguita alcuna azione;
   1.3. le informazioni contenute nelle tabelle originali non presenti nelle tabelle di *alanbot*, vengono aggiunte a quest'ultime;
   1.4. le informazioni non contenute nelle tabelle originali, ma presenti invece nelle tabelle di *alanbot*, vengono eliminate da quest'ultime;
2. viene utilizzato il servizio Tanl per estrarre soggetto, verbo, complemento oggetto, predicato e altri complementi da ogni frase presente nella tabella *eventi* di *oggisti* e viene popolata la tabella *eventipremast* con questi dati;

3. viene utilizzato il servizio Tanl per estrarre le Named Entity dalla tabella *eventi* di *oggisti* e viene popolata la tabella *personepremast* con questi dati.

Nei punti da 1.1 a 1.4 il database *alanbot* viene aggiornato modificando soltanto le informazioni differenti rispetto ai database *oggisti* e *hmrepicac* e lasciando invariate quelle uguali.
I risultati ottenuti alla fine di questo processo sono la tabella *eventipremast* che contiene eventi della storia dell'Informatica e la tabella *personepremast*, che contiene soggetti (persone, aziende, luoghi, ecc.) della storia dell'Informatica. Queste due tabelle verranno poi utilizzate da varie funzioni di cui disporrà il bot e la cui realizzazione verrà discussa nel successivo paragrafo.

5.5 Funzionalità specifiche

Le tabelle con le informazioni pre-masticate ottenute alla fine dei processi discussi nel par. 5.4 devono adesso essere sfruttate per eseguire le funzioni del bot che l'utente andrà a utilizzare e che saranno descritte in questo paragrafo.

5.5.1 Ricerca di un evento in base alla data

Questa funzione permette all'utente di sapere quale evento è accaduto in una certa data. Per accedere a questa funzione, l'utente ha due possibilità:

- inviare la data manualmente durante la conversazione chat;
- richiamare il comando /trova_evento.

Nei prossimi due paragrafi verranno discussi entrambi i metodi.

5.5.1.1 Inserimento manuale della data

All'invio di un messaggio del tipo "cosa è successo il 15/07/1990?", o "parlami dell'evento di maggio 2015", o un altro qualunque messaggio che contenga una data al suo interno, il bot si comporta nella seguente maniera:

1. estrae la data dal messaggio, distinguendo il giorno (se indicato), il mese (se indicato) e l'anno;
2. nel caso in cui sia stato indicato un giorno, un mese e un anno, controlla nella tabella *eventipremast* se esiste un evento in tale data;
   2.1. se esiste va al punto 5;
   2.2. se non esiste va al punto 3;
3. nel caso in cui un sia stato indicato un mese e un anno, controlla nella tabella *eventipremast* se esiste un evento in tale data;
   3.1. se esiste va al punto 5;
   3.2. se non esiste va al punto 4;
4. nel caso in cui sia stato indicato l'anno, controlla nella tabella *eventipremast* se esiste un evento in tale data;
   4.1. se esiste va al punto 5;
   4.2. se non esiste informa l'utente che non ha informazioni su quel certo anno;

5. sceglie casualmente tra gli eventi accaduti in quella data;
6. invia l'evento scelto all'utente.

Dell'evento sarà inizialmente presente soltanto il titolo e sarà possibile visualizzarlo interamente tramite il tap sul pulsante "Vuoi saperne di più?".

5.5.1.2 Chiamata del comando /trova_evento

Inviando "/trova_evento", o selezionandolo nella lista dei comandi, è possibile eseguire il suddetto comando. Una volta richiamato il comando, l'utente sarà tenuto a scegliere una data selezionandola tramite alcune risposte preimpostate.
Le operazioni precise che esegue il bot sono le seguenti:

1. chiede all'utente di scegliere un secolo, proponendogli tutti i secoli per i quali è presente almeno un evento, controllando nella tabella *eventipremast*;
2. scelto il secolo, chiede all'utente di scegliere una decade, proponendogli tutte le decadi del secolo scelto per le quali è presente almeno un evento;
3. scelta la decade, chiede all'utente di scegliere un anno, proponendogli tutti gli anni della decade scelta per i quali è presente almeno un evento;
4. scelto l'anno, controlla quanti eventi sono accaduti in quell'anno;
    1. se è accaduto un solo evento invia l'evento all'utente;
    2. se è accaduto più di un evento, chiede all'utente di scegliere una data, proponendogli tutte le date dell'anno scelto per le quali è presente un evento e invia all'utente l'evento della data scelta.

Dell'evento sarà inizialmente presente soltanto il titolo e sarà possibile visualizzarlo interamente tramite il tap sul pulsante "Vuoi saperne di più?".

5.5.2 Visualizzazione di un evento a caso

Tramite questa funzione l'utente può visualizzare un evento scelto casualmente tra tutti quelli presenti nel database.

La funzione si attiva inviando al bot messaggi del tipo "Parlami di un evento a caso", oppure "Dimmi un evento a caso", o altrimenti inviando il comando "/evento_random" o selezionandolo tra i comandi.

All'attivazione del comando, il bot seleziona un evento, in maniera casuale, tra tutti quelli presenti nella tabella *eventipremast*, e lo invia come risposta all'utente.
Dell'evento sarà inizialmente presente soltanto il titolo e sarà possibile visualizzarlo interamente tramite il tap sul pulsante "Vuoi saperne di più?".

5.5.3 Ricerca di un evento in base al soggetto

Questa funzione permette all'utente di visualizzare un evento indicando il nome di un soggetto della storia dell'informatica: una persona, un'azienda, una macchina, e via dicendo (vedi par. 5.4.2).
Per avviare questa funzione esistono due alternative:

- inviare il soggetto manualmente tramite la conversazione chat;
- richiamare il comando /trova_soggetto.

Nei prossimi due paragrafi verranno discussi entrambi i metodi.

5.5.3.1 Invio manuale del soggetto

All'invio di un messaggio del tipo "parlami di Alan Turing", o "cos'era UNIVAC?", o un qualunque altro messaggio che contenga un soggetto della storia dell'informatica presente nel database, il bot si comporta nella seguente maniera:

1. confronta ogni soggetto della tabella *personepremast* con il messaggio dell'utente cercando delle corrispondenze;
    1. se viene trovata più di una corrispondenza e quindi esiste più di un evento associato al soggetto, indica all'utente tutte le date degli eventi che ha trovato e l'utente sarà tenuto a scegliere quella desiderata;
    2. se viene trovata una sola corrispondenza, capisce che esiste un solo evento associato al soggetto;
2. invia all'utente l'evento selezionato.

Dell'evento sarà inizialmente presente soltanto il titolo e sarà possibile visualizzarlo interamente tramite il tap sul pulsante "Vuoi saperne di più?".

5.5.3.2 Chiamata del comando /trova_soggetto

Inviando "/trova_soggetto", o selezionandolo nella lista dei comandi, è possibile eseguire il suddetto comando. Una volta richiamato il comando, l'utente sarà tenuto a scegliere uno dei soggetti selezionandolo tramite alcune risposte preimpostate.
Le operazioni precise che esegue il bot sono le seguenti:

1. chiede all'utente di selezionare un soggetto della storia dell'informatica;
2. rende disponibili, come opzioni preimpostate, tutti i soggetti presenti nella tabella *personepremast*, eliminando i duplicati;
3. dopo che l'utente sceglie una tra le opzioni, invia l'evento relativo al soggetto scelto.

Dell'evento sarà inizialmente presente soltanto il titolo e sarà possibile visualizzarlo interamente tramite il tap sul pulsante "Vuoi saperne di più?".

5.5.4 Accadde oggi

Questa funzione permette di visualizzare, se disponibile, un evento della storia dell'informatica accaduto nel giorno odierno, o nella stessa settimana, in un qualche anno del passato.
Per avviare questa funzione è possibile inviare un messaggio del tipo "Cos'è successo oggi nella storia dell'informatica?" o "Parlami di un evento che accadde oggi", oppure inviando il comando "/accadde_oggi", o selezionandolo nella lista dei comandi.
Le operazioni che esegue il bot all'avvio della funzione sono le seguenti:

1. individua la data odierna e la relativa settimana;

2. per ogni evento presente nella tabella *eventi* controlla se è accaduto nello stesso giorno o nella stessa settimana;
3. sceglie un evento casuale tra quelli accaduti nello stesso giorno;
4. se non è presente nessun evento del giorno, sceglie un evento casuale tra quelli accaduti nella stessa settimana;
5. invia all'utente l'evento scelto;
6. se non è presente nessun evento del giorno o della settimana, lo comunica all'utente.

Dell'evento sarà inizialmente presente soltanto il titolo e sarà possibile visualizzarlo interamente tramite il tap sul pulsante "Vuoi saperne di più?".

5.5.5 Modalità quiz

Tramite questa funzione è possibile mettere alla prova la propria conoscenza sulla storia dell'informatica. Verranno infatti poste delle domande all'utente e sarà valutata la correttezza delle risposte.
Sono state realizzate due diverse tipologie di quiz: un quiz sul Sai Quando e uno sul Sai Chi.
I metodi si basano sulle informazioni presenti nel database *eventipremast*, che grazie alla sua struttura nella quale distingue per ogni evento il soggetto, il verbo e i vari complementi (vedi cap. 5.4) permette di poter avere uno standard secondo il quale formulare le domande dei quiz.
Entrambe le funzioni saranno discusse nello specifico nei paragrafi seguenti.

5.5.5.1 Quiz Sai Quando

In questa modalità viene presentato un evento all'utente, che sarà tenuto a indicare l'anno in cui questo evento è accaduto.
È possibile accedere alla funzione inviando il comando "/quiz_sai_quando", o selezionandolo nella lista dei comandi.
Una volta che il comando è stato chiamato, il bot esegue le seguenti operazioni:

1. invia un testo di descrizione della funzione e chiede all'utente se vuole iniziare il quiz;
2. in caso affermativo, sceglie a caso un evento dalla tabella *eventipremast* e lo presenta all'utente chiedendogli in che anno è avvenuto;
3. se l'utente indica un anno anteriore alla data esatta, gli risponde "No, dopo!";
4. se l'utente indica un anno posteriore alla data esatta, gli risponde "No, prima!";
5. se l'utente indovina l'anno, gli risponde che ha indovinato, aggiungendo un commento sulla sua bravura, dipendentemente dal numero di tentativi fatti per indovinare;
6. se prima di indovinare, l'utente invia un qualunque messaggio che non contenga un anno, il bot interpreta il messaggio come una resa e invia la risposta corretta;

7. sia che l'utente abbia indovinato, sia che si sia arreso, il bot invia un messaggio "Vuoi saperne di più", che dà la possibilità all'utente di visualizzare l'evento;
8. subito di seguito, invia un messaggio di richiesta di feedback sulla qualità della domanda;
9. subito di seguito, invia un messaggio che chiede all'utente se vuole passare alla prossima domanda, o se vuole uscire dalla modalità quiz.

Per quanto riguarda il punto 8, il bot registrerà tutti i feedback degli utenti, e quando per una domanda il netto tra i i feedback negativi e positivi sarà maggiore di 5 negativi, questa domanda non sarà più proposta agli utenti.
Se un utente dà un feedback a una domanda che aveva già valutato, questo non sarà registrato.

5.5.5.2 Quiz Sai Chi

In questa modalità viene presentato un evento all'utente, di cui vengono specificate la data e le azioni compiute, ma viene omesso il soggetto. L'utente sarà tenuto a selezionare il soggetto che ritiene essere corretto, scegliendolo tra alcune opzioni preimpostate.
È possibile accedere alla funzione inviando il comando "/quiz_sai_chi", o scegliendolo nella lista dei comandi.
Le operazioni precise che effettua il bot, all'avvio della funzione, sono le seguenti:

1. invia un testo di descrizione della funzione e chiede all'utente se vuole iniziare il quiz;
2. in caso affermativo, sceglie a caso un evento dalla tabella *eventipremast* e lo presenta all'utente omettendone il soggetto;
3. il soggetto corretto e altri 3 soggetti di 3 eventi scelti a caso dalla tabella *eventipremast*, vengono visualizzati, in ordine casuale, come risposte preimpostate, tra le quali viene chiesto all'utente di sceglierne una;
4. se l'utente sceglie la risposta corretta, il bot invia i suoi complimenti;
5. se l'utente sceglie una riposta sbagliata, il bot glielo comunica e gli indica quella che era la risposta corretta;
6. sia che l'utente abbia o non abbia indovinato, il bot invia un messaggio "Vuoi saperne di più?", che dà la possibilità all'utente di visualizzare l'evento;
7. subito di seguito, invia un messaggio di richiesta di feedback sulla qualità della domanda;
8. subito di seguito, invia un messaggio che chiede all'utente se vuole passare alla prossima domanda, o se vuole uscire dalla modalità quiz.

Per quanto riguarda il punto 7, il bot registrerà tutti i feedback degli utenti, e quando per una domanda il netto tra i i feedback negativi e positivi sarà maggiore di 5 negativi, questa domanda non sarà più proposta agli utenti.
Se un utente dà un feedback a una domanda che aveva già valutato, questo non sarà registrato.

5.6 Realizzazione del pannello di controllo del bot

Per poter gestire in modo rapido le opzioni e le funzionalità del bot è stato creato un Pannello di Controllo del bot, inserito all'interno del pannello di amministrazione del progetto HMR.
Nel Pannello di Controllo, sono presenti varie sezioni, ad ognuna delle quali sarà dedicato un paragrafo qui di seguito.

5.6.1 Visualizzazione del log

In questa sezione è possibile visualizzare le conversazioni tra il bot e gli utenti. Per ogni messaggio vengono specificati:

- la data;
- l'ora;
- il codice dell'utente;
- l'username dell'utente, se presente.

È inoltre possibile svuotare il log, azzerando tutti i dati relativi alle conversazioni, o aggiornarlo, visualizzando i nuovi messaggi scambiati, se presenti.

5.6.2 Invio di messaggi di broadcast

In questa sezione è possibile inviare uno stesso messaggio a tutti gli utenti che hanno iniziato una conversazione con il bot.
Si lascia aperto a futuri sviluppi la possibilità di creare una lista di utenti a cui inviare i messaggi di broadcast e lasciare libera la possibilità di farne o meno parte.

5.6.3 Aggiornamento del database pre-masticato

In questa sezione è possibile avviare l'esecuzione delle operazioni descritte al par. 5.4, che restituiscono come risultato il database con le informazioni strutturate in maniera che il bot possa facilmente usarle per le sue funzioni.

Per eseguire la pre-masticazione viene sfruttato il servizio Tanl (vedi par. 1.2.6.4), che per funzionare ha bisogno di un token Google che ne autentichi la sessione. Da questa pagina è possibile inserire il token, relativo all'account hmrtanlusr@gmail.com, creato ad hoc.

5.6.4 Gestione delle Named Entity

In questa sezione è possibile visualizzare le Named Entity, o entità nominate, che sono state riconosciute all'interno del database nel corso del processo di pre-masticazione descritto al par. 5.4.
È inoltre possibile modificare ognuna di queste Named Entity, nel caso in cui siano state estratte in modo errato, oppure eliminarle.

Queste modifiche restano tali anche nel momento in cui il database viene aggiornato.

5.6.5 Visualizzazione dei feedback

In questa sezione è possibile visualizzare le valutazioni date dagli utenti alle domande dei quiz sul “Sai Chi” e sul “Sai Quando”, descritti nel par. 5.5.5.

Per ognuna di queste domande viene inizializzato un valore pari a 5, che può crescere o decrescere di una unità in base rispettivamente al giudizio positivo o negativo che gli utenti danno a ogni domanda. Se il feedback arriva a 0 la domanda non viene più visualizzata.

5.6.6 Visualizzazione delle statistiche

In questa sezione è possibile visualizzare le statistiche relative alle conversazioni che il bot ha con gli utenti.
Oltre a mostrare il numero totale di messaggi inviati, sono disponibili due grafici diversi:

- comprensione degli input;
- messaggi inviati da ogni utente.

Il primo grafico è relativo alla capacità di conversazione del bot e mostra la percentuale di messaggi inviati dagli utenti per cui il bot ha saputo dare un’interpretazione (non necessariamente corretta) e ha fornito una risposta.

Il secondo grafico rappresenta una fotografia dell’utenza, mostrando la percentuale di messaggi che ogni utente ha inviato, in relazione col numero totale di messaggi inviati dagli utenti.
È possibile scaricare questi dati in formato CSV, oppure azzerare le statistiche. In quest’ultimo caso, il numero totale di messaggi inviati viene reimpostato a zero.

5.7 Struttura del codice sul server

Di seguito viene descritta la struttura principale dei file che risiedono sul server di HMR.

- bot.progettohmr.it
  Sotto-dominio contenente tutti i file che gestiscono i messaggi degli utenti al bot e successiva risposta
  - session/ *contiene i file che permettono la gestione delle sessioni degli utenti*
    - class.DB.php
    - class.session.php
    - debug.php
  - callbackData.php – *la gestione del metodo callback di Telegram*
  - connessioneDB.php – *la gestione dell'accesso ai database e successive interrogazioni*
  - conversazioneBase.php – *le possibili risposte a un messaggio di conversazione base*

    - dailyCron.php – *le azioni da eseguire giornalmente*
    - functions.php – *contiene le funzioni utilizzate nei vari file*
    - invioRisposta.php – *le operazioni da eseguire per inviare la risposta all'utente*
    - operazioniOggiSTI.php – *contiene le operazioni legate al database oggiSTI*
    - operazioniPreliminari.php – *contiene le operazioni che hanno priorità di esecuzione*
    - quiz.php – *la gestione della modalità quiz*
- public_html/Bot
  Cartella contenente i file relativi alla pre-strutturazione del database
    - premast.php – *la gestione dell'accesso al database e successiva scrittura dei dati pre-strutturati*
    - script.sh – *la gestione della richiesta a Tanl per ottenere l'analisi NLP dei testi*
- public_html/amministrazione/asset
  Cartella contenente i file relativi alle pagine del Pannello di Controllo
    - css/
        - botStyle.css – *il foglio di stile delle pagine di amministrazione del bot*
    - html/
        - ammBot.php – *la pagina principale del Pannello di Controllo del bot*
        - broadcastBot.php – *la pagina relativa all'invio di messaggi di broadcast*
        - feedbackBot.php – *la pagina relativa alla visualizzazione dei feedback*
        - logBot.php – *la pagina relativa alla visualizzazione del log*
        - nerBot.php – *la pagina relativa alla gestione delle Named Entity*
        - premastBot.php – *la pagina relativa all'aggiornamento del database*
        - statsBot.php – *la pagina relativa alla visualizzazione delle statistiche*

La parte dei file relativi al funzionamento del bot contiene un totale di 2184 righe (di cui 1038 nel file *conversazioneBase.php*).
La parte dei file relativi alla pre-strutturazione dei dati contiene un totale di 678 righe (di cui 657 nel file *premast.php*).
La parte dei file relativi al Pannello di Controllo contiene un totale di 1455 righe.
Il totale assoluto di righe di codice è di 4317.

5.8 Verifica e validazione

Verifica e validazione sono due termini precisi dell'ingegneria del software (B. W. Boehm, 1984; in Cignoni e De Risi, 1998).
Con il termine verifica si intende il controllo fatto sul prodotto software atto a valutare quanto esso rispetti o meno i requisiti posti a monte.
Con il termine validazione si intende il controllo fatto, in genere, sul prodotto finito, atto a valutare quanto i requisiti posti inizialmente fossero corretti e quindi a definire la qualità del prodotto rispetto alle esigenze di utilizzo.

5.8.1 Verifica

Per quanto riguarda la verifica del nostro prodotto, il bot, allo stato in cui è stato completato per questo progetto di tesi, dispone di tutte le funzioni che erano state indicate nei requisiti (vedi par. 4.3.2) e in particolare:

- è in grado di fornire informazioni sui soggetti dell'informatica, sia se richiesto durante la conversazione base, sia tramite risposte preimpostate (RF1.1);
- è in grado di dire cosa è successo in una determinata data, sia se richiesto durante la conversazione base, sia tramite risposte preimpostate (RF1.2.1);
- è in grado di dire cos'è successo oggi nella storia dell'informatica (RF1.2.2).
- dispone della modalità quiz, sia sul "sai chi" che sul "sai quando" (RF2);
- è in grado di intrattenere una conversazione base con risultati accettabili[6], evitando linguaggi offensivi, utilizzando informazioni dette in momenti passati della conversazione e indirizzando l'utente sul campo di competenza nel caso in cui non sappia rispondere (RF3);
- dispone di una lista di comandi Telegram tra cui poter facilmente scegliere (RF4);
- il Pannello di Controllo del bot contiene tutte le funzioni specificate nei requisiti e altre aggiuntive (da RF5 a RF8).

Riguardo la gestione delle Named Entity, è stato sviluppato un sistema più sofisticato rispetto a quanto era stato inizialmente ipotizzato. Come spiegato nel par. 5.6.4, dopo che le Named Entity vengono estratte dal testo, è possibile modificarle o eliminarle manualmente, attraverso il pannello di amministrazione. In questo modo i risultati finali che si ottengono sono molto più precisi e privi di imperfezioni.
Questo sistema attualmente[7] utilizza la tabella *eventi* del database *oggiSTI*, che contiene solo 13 eventi, e restituisce un totale di 49 Named Entity, di cui 12 sono state modificate e 9 eliminate.

5.8.2 Validazione

Per quanto riguarda la validazione, disponiamo di dati relativi solo al breve periodo in cui il prodotto è stato utilizzabile.
Dai primi sviluppi – e quindi intorno a giugno 2017 – fino alla metà di ottobre 2017, è stato possibile testare il funzionamento del bot tramite l'osservazione delle chat con diversi utenti coinvolti come beta-tester.
Nel periodo tra il 04/10/2017 e il 24/10/2017 il bot ha scambiato con gli utenti 554 messaggi ottenendo la percentuale di comprensione mostrata in Figura 11.

6 Nella prima metà del mese di ottobre 2017, la percentuale di comprensione si è mantenuta intorno al 86-87%.
7 Relativemente alla data di oggi, 24/10/2017.

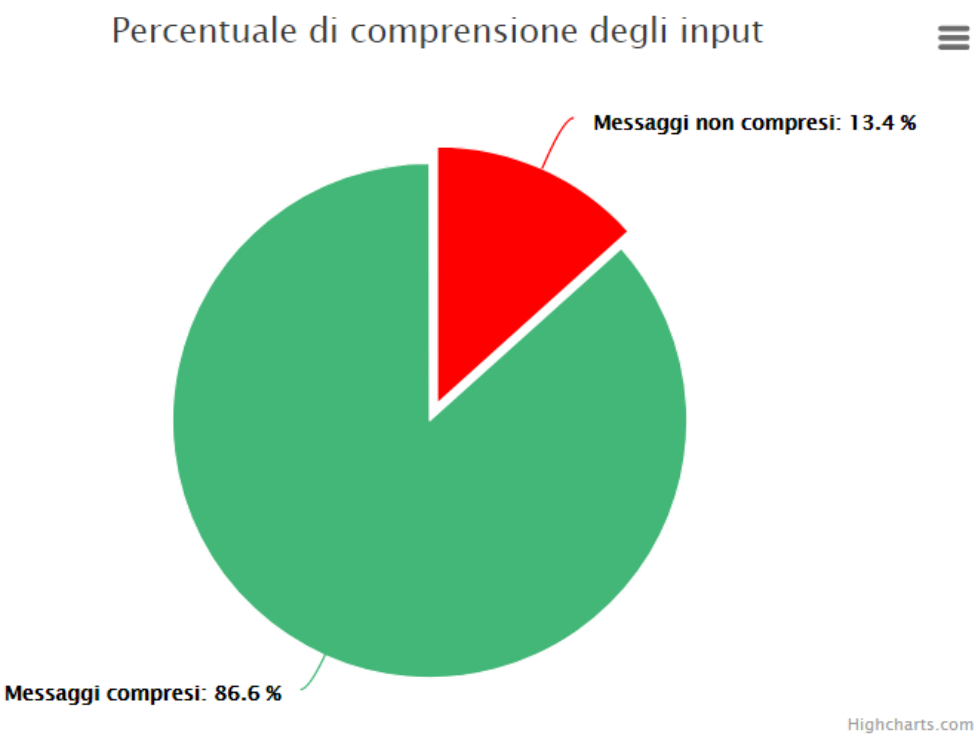

Figura 11: Messaggi compresi dal bot

È stato possibile osservare che quasi in nessun caso la conversazione si è limitata a uno scambio di messaggi, ma che molto spesso è stata richiesta l'esecuzione di funzionalità specifiche, legate alla capacità di reperire e fornire informazioni sulla storia dell'informatica, anche se al momento, essendo il database da cui il bot trae la sua conoscenza un progetto appena partito, i contenuti non sono moltissimi.
I giudizi dei beta-tester coinvolti sono stati in generale positivi, sia per quanto riguarda le funzioni specifiche, ottenendo informazioni utili sulla storia dell'informatica, sia per quanto riguarda la conversazione base, che è risultata essere piacevole e divertente.
Ovviamente l'utilizzo del bot e il suo confronto con il pubblico è ancora all'esordio, ma le premesse sono buone e c'è un ampio margine di miglioramento.

Conclusioni e sviluppi futuri

Come analizzato dettagliatamente nel capitolo 5.8, gli obiettivi posti nei requisiti sono stati raggiunti, ottenendo risultati molto soddisfacenti.
Il Natural Language Processing è stato molto utile ai fini della realizzazione del progetto, in particolare per riuscire ad analizzare, grazie all'integrazione del servizio Tanl, le informazioni presenti nel database di *oggiSTI* e poterle pre-masticare per un più conveniente utilizzo.

Il bot, durante il suo periodo di progettazione, è passato dal saper rispondere a un insieme molto ristretto di domande a possedere tutte le funzionalità che ci eravamo prefissi nell'analisi dei requisiti e al saper fornire una risposta a quasi il 90% degli input, come misurato dalle statistiche (vedi par. 5.6.6)

Sui risultati raggiunti si può ancora lavorare e sono molteplici le possibilità di miglioramento.
Fra gli sviluppi già individuati e fattibili nelle condizioni attuali ci sono:

- aggiunta di funzionalità collegate al database di *EPICAC* per fornire informazioni riguardo alle attività del progetto HMR*;*
- miglioramento del riconoscimento dei soggetti durante la conversazione base, introducendo una gestione dei sinonimi;
- possibilità di amministrazione della tabella *eventipremast* nel pannello di controllo con funzionalità analoghe a quelle utilizzate per le Named Entity;
- sfruttare campi aggiunti nel frattempo al database di *oggiSTI*, come ad esempio le parole chiave;
- gestione della lista di utenti a cui inviare messaggi di broadcast.

Essendo le tecnologie di comprensione automatica del linguaggio naturale ancora un campo in via di sviluppo, è ovvio che anche la capacità di fare conversazione di Alan (così come è stato chiamato il bot) non è perfetta. È però possibile adottare tecniche di comprensione degli input più raffinate ispirandosi a quelle già in uso nel variopinto mondo dei chatbot.

Per quanto riguarda le funzionalità principali del bot, risultano essere tutte correttamente operanti con i dati presenti attualmente. Sarà interessante verificare il il comportamento del bot con i nuovi dati che verranno inseriti in futuro.

L'attuale versione stabile di Alan è disponibile sulla piattaforma Telegram cercando il tag @ProgettoHMRBot e il sito di supporto del bot è online all'indirizzo www.bot.progettohmr.it.
Si invitano i lettori di questa relazione a iniziare una conversazione con il bot per poter testare, nella pratica, il prodotto finale che è stato realizzato.

Bibliografia

B. W. Boehm. 1984. *Verifying and Validating Software Requirements and Design Specifications*, Los Alamitos Vol. 1, Iss. 1.

Choe Do Kook, Charniak Eugene. 2016. *Parsing as a Language Modeling*. EMNLP.

Cignoni G.A. e P. De Risi. 1998. *Il test e la qualità del software,* Ed. Il Sole 24 Ore, Milano.

IEEE Std 830-1993 — IEEE Recommended Practice for Software Requirements Specifications. 1993.

Goldberg, Yoan. 2016. *A Primer on Neural Network Models for Natural Language Processing*. “Journal of Artificial Intelligence Research”, 57, pp. 345-420.

Jozefowicz Rafal, Vinyals Oriol, Schuster Mike, Shazeer Noam, Wu Yonghui. 2016. *Exploring the Limits of Language Modeling*. Cornell University Library.

Lenci Alessandro, Simonetta Montemagni, e Vito Pirrelli. 2005. *Testo e Computer: Elementi di linguistica computazionale*. Carocci.

Lenzi Emanuele, 2017. *Studio per la realizzazione di un CMS per HMR.* Tirocinio in Informatica Umanistica, Università di Pisa.

DellaPietra Stephen e Vincent. 1994. *The Candide System for Machine Translation.* “94 Proceedings of the workshop on Human Language Technology”, pp. 157-162.

Pratelli Nicolò, 2017. *Un’applicazione web: Oggi nella storia dell’informatica.* Relazione di Laurea in Informatica Umanistica, Università di Pisa.

Turing, Alan M. 1950. *Computing machinery and intelligence*. “Mind”, 59, pp. 433-460.

Vinyals, Oriol. 2015. *Grammar as a Foreign Language,* “15 Proceedings of the 28th International Conference on Neural Information Processing Systems”, pp. 2773-2781.

Sitografia

A.L.I.C.E. Artificial Intelligence Foundation, http://alice.pandorabots.com/ (visitato il 28 giugno 2017).

Bots: An introduction for developers, https://core.telegram.org/bots (visitato il 3 luglio 2017).

Chatbot Tutorial, https://www.codeproject.com/Articles/36106/Chatbot-Tutorial (visitato il 9 luglio 2017).

Dr. Sbaitso, https://classicreload.com/dr-sbaitso.html (visitato il 28 giugno 2017).

Eliza, computer therapist, http://manifestation.com/neurotoys/eliza.php3/ (visitato il 28 giugno 2017).

How to build Eliza Chatterbot, http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=5369&lngWId=3 (visitato il 9 luglio 2017).

Il cinguettio dei «bot»: su Twitter il 15% degli account è un software, http://www.ilsole24ore.com/art/impresa-e-territori/2017-03-18/il-cinguettio-bot-twitter-15percento-account-e-software--183532.shtml?uuid=AEjYqIp&refresh_ce=1 (visitato il 6 luglio 2017).

Jabberwacky, http://www.jabberwacky.com/ (visitato il 28 giugno 2017).
Loebner Prize, http://www.loebner.net/Prizef/loebner-prize.html (visitato il 26 giugno 2017).

Marvin's Marvellous Guide to All Things Webhook, https://core.telegram.org/bots/webhooks (visitato il 30 luglio 2017).

Meaning Cloud, https://www.meaningcloud.com/ (visitato il 01/08/2017).

Messenger Platform, https://developers.facebook.com/docs/messenger-platform (visitato il 4 luglio 2017).

ModSecurity, https://documentation.cpanel.net/display/EA/Apache+Module%3A+ModSecurity (visitato il 26 luglio 2017).

Natural Language Processing (NLP), http://searchbusinessanalytics.techtarget.com/definition/natural-language-processing-NLP (visitato il 3 novembre 2017).

Natural Language Toolkit, http://www.nltk.org/ (visitato il 26 luglio 2017).
Parry: The A.I. chatterbot from 1972, https://phrasee.co/parry-the-a-i-chatterbot-from-1972/ (visitato il 27 giugno 2017).

Say hello to Skype chat bots, https://www.skype.com/en/features/bots/ (visitato il 3 luglio 2017).

SimilarWeb, http://www.similarweb.com (visitato il 6 luglio 2017).

Smarterchild, https://www.chatbots.org/chatterbot/smarterchild/ (visitato il 28 giugno 2017).

Software tricks people into thinking it is human, https://www.newscientist.com/article/dn20865-software-tricks-people-into-thinking-it-is-human/ (visitato il 28 giugno 2017).

Statista, http://www.statista.com (visitato il 6 luglio 2017).

Tanl Italian Pipeline, http://tanl.di.unipi.it/it/ (visitato il 01/08/2017).

Telegram Bot API, https://core.telegram.org/bots/api (visitato il 29/07/2017).

Telegram Messenger, https://telegram.org/ (visitato il 12 luglio 2017).

Tint, http://tint.fbk.eu/ (visitato il 01/08/2017).

Tree Tagger, http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/ (visitato il 01/08/2017).

Twitter Platform, https://dev.twitter.com/ (visitato il 4 luglio 2017).

Wikipedia, voce *Chinese Room*, https://en.wikipedia.org/wiki/Chinese_room (visitato il 26 giugno 2017).

Wikipedia, voce *Logica Pull*, https://it.wikipedia.org/wiki/Logica_pull (visitato il 28 luglio 2017).

Wikipedia, voce *Logica Push*, https://it.wikipedia.org/wiki/Logica_push (visitato il 28 luglio 2017).

Wikipedia, voce *Pattern Matching*, https://en.wikipedia.org/wiki/Pattern_matching (visitato il 9 luglio 2017).