**UNIVERSITA’ DEGLI STUDI DI NAPOLI**

**“PARTHENOPE”**



**SCUOLA INTERDIPARTIMENTALE**

**DELLE SCIENZE, DELL’INGEGNERIA**

**E DELLA SALUTE INFORMATICA**

**Dipartimento di Studi**

**Scienze e Tecnologie**

**Corso di Laurea in**

**Informatica**

*Tesi di Laurea in*

*Natural Language Processing e agenti conversazionali*

**Titolo tesi**

**Sviluppo di un Chatbot per l’orientamento universitario**

|  |  |
| --- | --- |
| TUTOR  Chiar.mo Prof.  Antonino Staiano | CANDIDATO  Dominick Ferraro  MATR. 0124002048 |

ANNO ACCADEMICO 2021/2022

**INDICE**

Sommario

[ABSTRACT 3](#_Toc126999912)

[CAPITOLO 1 - INTRODUZIONE 7](#_Toc126999913)

[1.1 – DIFFUSIONE DEI CHATBOT 8](#_Toc126999914)

[1.2 – CLASSIFICAZIONE DI UN CHATBOT 9](#_Toc126999915)

[1.3 – FUNZIONAMENTO DI UN CHATBOT 10](#_Toc126999916)

[**1.3.1 – FONDAMENTI DI NATURAL LANGUAGE PROCESSING PER LO SVILUPPO DI UN AGENTE CONVERSAZIONALE 12**](#_Toc126999917)

[CAPITOLO 2 – DEFINIZIONE DEL DOMINIO APPLICATIVO E REQUISITI DI ORIENTABOT 19](#_Toc126999918)

[2.1 – SCENARI 19](#_Toc126999919)

[2.2 – REQUISITI FUNZIONALI E NON FUNZIONALI 20](#_Toc126999920)

[2.3 – OBIETTIVI DI PROGETTAZIONE 22](#_Toc126999921)

[CAPITOLO 3 – STRUMENTI PER LO SVILUPPO 23](#_Toc126999922)

[3.1 – LIBRERIE E FRAMEWORK 24](#_Toc126999923)

[**3.1.1 – NATURAL LANGUAGE TOOLKIT 25**](#_Toc126999924)

[**3.1.2 – NUMPY 26**](#_Toc126999925)

[**3.1.3 – PYTORCH 27**](#_Toc126999926)

[**3.1.4 – TKINTER 28**](#_Toc126999927)

[CAPITOLO 4 – SVILUPPO DELL’APPLICAZIONE 30](#_Toc126999928)

[4.1 – ALGORITMO DI ORIENTABOT 33](#_Toc126999929)

[CAPITOLO 5 – INTERFACCIA E USABILITÀ 36](#_Toc126999930)

[5.1 – TEST DI USABILITÀ 40](#_Toc126999931)

[5.2 – VALUTAZIONE DI USABILITÀ 42](#_Toc126999932)

# 

**ELENCO DELLE FIGURE**

[**Figura 1 - Questionario agli studenti 6**](#_Toc127005383)

[**Figura 2 - Interfaccia di ELIZA 7**](#_Toc127005384)

[**Figura 3 - Esempio Chatbot Poste Italiane 9**](#_Toc127005385)

[**Figura 4 - Associazione Pattern → Response 11**](#_Toc127005386)

[**Figura 5 - Campi NLP 13**](#_Toc127005387)

[**Figura 6 - I sei significati di عقد 15**](#_Toc127005388)

[**Figura 7 - Stemming 16**](#_Toc127005389)

[**Figura 8 - Lowering 17**](#_Toc127005390)

[**Figura 9 - Lowering + Ignore Punctuation 17**](#_Toc127005391)

[**Figura 10 – Lemmatization 18**](#_Toc127005392)

[**Figura 11 – Sequence Diagram Chatbot 21**](#_Toc127005393)

[**Figura 12 - TIOBE Top 10 Programming Language 23**](#_Toc127005394)

[**Figura 13 - OrientaBot Package 24**](#_Toc127005395)

[**Figura 14 - Funzioni di NumPy 26**](#_Toc127005396)

[**Figura 15 – PyTorch 27**](#_Toc127005397)

[**Figura 16 - Tkinter Install 29**](#_Toc127005398)

[**Figura 17 - Creazione della bag of words 32**](#_Toc127005399)

[**Figura 18 - Passi dell'algoritmo NLP 33**](#_Toc127005400)

[**Figura 19 - Palette Interfaccia OrientaBot 37**](#_Toc127005401)

[**Figura 20 – ORIENTABOT GUI 37**](#_Toc127005401)

[**Figura 21 – focus sulla barra di testo e pulsante invio 37**](#_Toc127005401)

[**Figura 22 - Avatar OrientaBot 39**](#_Toc127005402)

[**Figura 23 - Valutazione Chatbot Università di Padova 42**](#_Toc127005403)

[**Figura 24 - Valutazione OrientaBot 43**](#_Toc127005404)

[**Figura 25 - Conversazione con OrientaBot n.1 44**](#_Toc127005405)

[**Figura 26 - Interazione con OrientaBot n.2 45**](#_Toc127005406)

# **ABSTRACT**

Spesso la ricerca di informazioni puó essere difficile quando sei in un ambiente nuovo, è il caso dell'università dove alcune informazioni possono essere complesse da ricercare oppure necessitano di un confronto con altri addetti come docenti, segreteria…

Per una matricola appena arrivata è difficile orientarsi, per non parlare di chi ancora non è iscritto.

Personalmente ho sempre cercato informazioni autonomamente, a volte impiegando tempo e facendo viaggi in treno solo per parlare per cinque minuti con la segreteria, sottraendo tempo prezioso ai miei studi.

Il carattere di una persona poi gioca la sua parte: quante volte non abbiamo chiesto un’informazione per timidezza o per paura di essere giudicati male?

Dopo aver fatto un sondaggio tra miei colleghi e coetanei ho deciso di sviluppare un Chatbot per l'orientamento degli studenti universitari, utile a chi alle nuove matricole, a chi si iscriverà ma anche a chi frequenta tutti i giorni questo fantastico ambiente ed è in cerca di un’informazione veloce.

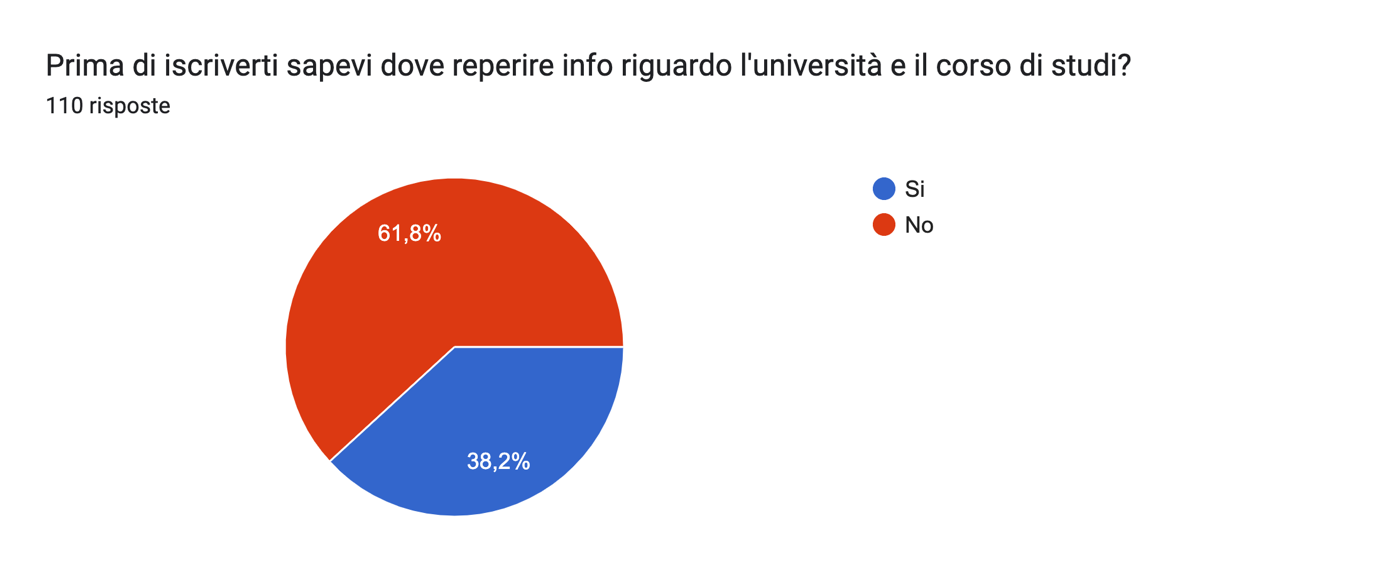
Secondo un sondaggio da me effettuato, su 110 studenti universitari:

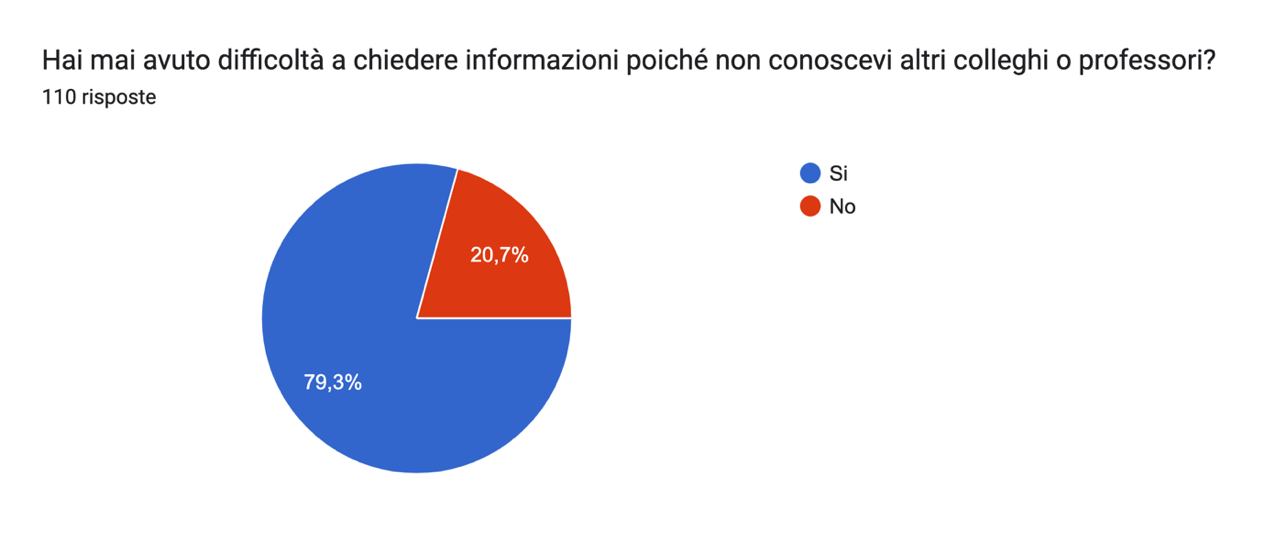
* **68** studenti non sapevano dove reperire informazioni prima di iscriversi all'universitá.
* **88** studenti hanno avuto difficoltà a chiedere informazioni poiché non conoscevano professori o altri colleghi.
* **104** studenti userebbero un Chatbot che risponde alle loro richieste riguardanti l'ambiente universitario.

Nasce così OrientaBot, un chatbot che voglio donare alla mia universitá, per aiutare i miei attuali e futuri colleghi.

Nell'elaborato verrà descritto a pieno cos’è un chatbot, quali tecnologie utilizza, come è stato ideato e sviluppato e come funziona.

Ovviamente questo lavoro è da considerarsi un supporto all’orientamento, senza voler sminuire l’ottimo lavoro che l’Università offre.





****

Figura 1 - Questionario agli studenti

# **CAPITOLO 1 - INTRODUZIONE**

Il boom tecnologico degli ultimi anni ha sottolineato una necessità mancante all’essere umano: l’utilizzo di assistenti digitali o agenti conversazionali.

Il primo assistente digitale di cui abbiamo notizie è ***ELIZA***, un’intelligenza artificiale sviluppata dalla MIT nel 1966. ELIZA simulava un terapista e rispondeva alle domande dei pazienti in base a modelli di conversazione prestabiliti.

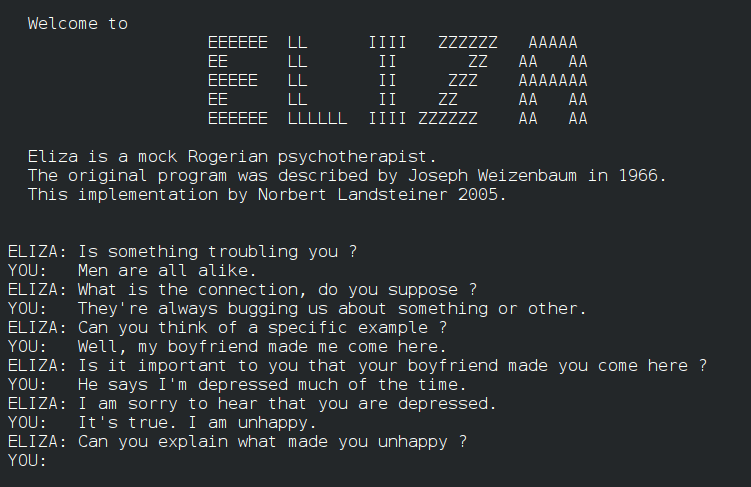


Figura 2 - Interfaccia di ELIZA

I primi esperimenti risalgono agli anni ’50 con il Test di Alan Turing che consisteva nel far parlare, con la scrittura, una persona con un umano ed un bot che si alternavano a sua insaputa.

L’esperimento aveva successo se l’utente non si accorgeva della differenza tra l’uomo e la macchina.

Lo sviluppo si arrestò negli anni ’90 a causa della tecnologia ancora di basso livello. Il vero problema era dare uno scopo specifico al bot.

In tempi più recenti, nell’ottobre 2011, Apple rilasciò la prima versione di Siri, un’intelligenza artificiale al supporto degli utenti. Siri permetteva di aggiungere numeri in rubrica, effettuare telefonate, scrivere dei messaggi e tanto altro.

Ovviamente questo fu solo l’inizio di una nuova era, per il mercato e per tutti i fruitori.

Dopo Apple, le grandi aziende Tech hanno sviluppato un proprio agente conversazionale:

* Google Assistant - Google
* Cortana - Microsoft
* Alexa – Amazon
* Bixby – Samsung

Il motivo della forte diffusione di questi assistenti deriva dal fatto che essi sono integrati nei dispositivi quali smartphone, tablet, auto, smart speaker e riescono a fare semplici e veloci task in breve tempo.

## **1.1 – DIFFUSIONE DEI CHATBOT**

Un agente conversazionale che sta spopolando nel 2023 è *Chat GPT,* sviluppato da OpenAI. L’utente può interagire attraverso una chat con l’assistente digitale e chiedergli ciò che necessità.

Definiamo quindi il concetto di **chatbot**:applicazione software usata per interagire con le conversazioni umane in modo naturale.

L’esistenza di un chatbot personale per un’azienda è diventata una necessità da adempiere: tutti i siti web di ogni servizio pubblico o privato hanno un chatbot personalizzato al supporto dell’utente (*Poste Italiane, INPS, Amazon, Unicredit).*

Avere un chatbot per un’azienda significa essere al passo coi tempi.

Secondo *BI Intelligence,* nel 2020 l’80% dei brand ha iniziato lo sviluppo di un chatbot personale per l’interazione e l’assistenza clienti.

Una stima dell’Istituto per la Competitività (I-Com) afferma che il mercato globale dei chatbot potrebbe raggiungere 1,25 miliardi di dollari entro il 2025, con una crescita nel prossimo biennio pari a un tasso medio annuo superiore al 37%.

## **1.2 – CLASSIFICAZIONE DI UN CHATBOT**

Un chatbot interpreta il testo scritto dall’utente e fornisce una risposta coerente al quesito, cercando di soddisfare al meglio la richiesta.

Quando usiamo un chatbot abbiamo davanti una semplice schermata dove possiamo interagire con l’assistente digitale, ecco un esempio:

Immagine che contiene testo

Descrizione generata automaticamente

Figura 3 - Esempio Chatbot Poste Italiane

In che modo il chatbot interpreta il linguaggio naturale e fornisce le risposte adeguate?

I chatbot elaborano in linguaggio naturale ( *Natural Language Processing),* sono guidati da *AI,* regole automatizzate e *Machine Learning.*

Abbiamo due tipi di chatbot:

1. Dichiarativi: sono programmi che svolgono un unico task. Sfruttando NLP, ML e regole interne riescono a generare risposte automatiche e colloquiali ai quesiti degli utenti.

Un chatbot dichiarativo viene creato ad hoc per un determinato contesto, riuscendo a gestire domande frequenti e comuni. Pur usando tecniche NLP la loro interazione risulta essere limitata poiché vi è un *dataset* alla base più o meno grande, gestito dagli sviluppatori.

Attualmente i chatbot dichiarativi sono quelli più utilizzati poiché sono semplici da utilizzare e forniscono un forte supporto agli utenti di un determinato servizio.

1. Predittivi: sono indicati come *virtual assistant* e sono più interattivi e sofisticati rispetto ai chatbot dichiarativi. Questi assistenti. Conoscono il contesto di riferimento e sfruttano la *NLU, NLP* e *ML* per apprendere.

Con l’applicazione di intelligenza predittiva e analisi di dati riescono a studiare l’utente, riuscendo a fornire risposte sempre più personali, in modo da risolvere problemi o addirittura prevenirli.

I chatbot predittivi sono quelli diffusi dalle Big Tech e molto spesso sono integrati in dispositivi smart.

## **1.3 – FUNZIONAMENTO DI UN CHATBOT**

Un chatbot prende in input delle frasi in linguaggio naturale, le elabora in linguaggio macchina e fornisce una risposta adeguata.

Per fare ciò ci sono alcune procedure da dover analizzare per rendere comprensibile il funzionamento di un agente conversazionale.

Per prima cosa, dobbiamo creare un *training data* personalizzato, a seconda del dominio applicativo.

Un training data è una collezione di dati processati dal chatbot per rispondere adeguatamente a un quesito.

Abbiamo diversi tipi di training data, quello che analizzeremo è quello usato in questo lavoro.

Un training data contiene tutte le possibili interazioni di un utente generico: domande frequenti, problemi tipici, interazioni generali e richiesta di servizi.

Il training data usato è una tripla ***tag – pattern – response*:**

* **Tag**: codice identificativo univoco dell’argomento trattato.
* **Pattern**: è l’insieme delle possibili interazioni dell’utente.

Un pattern non è altro che la stessa frase scritta in modi differenti, in modo tale che il chatbot possa apprendere più sfumature del linguaggio naturale e riconoscere di cosa stiamo parlando.

* **Response**: è l’insieme delle risposte che il chatbot fornisce all’utente in base a una specifica domanda.

Come per i pattern, possiamo fornire più risposte per lo stesso argomento per diversificare il chatbot il più possibile.

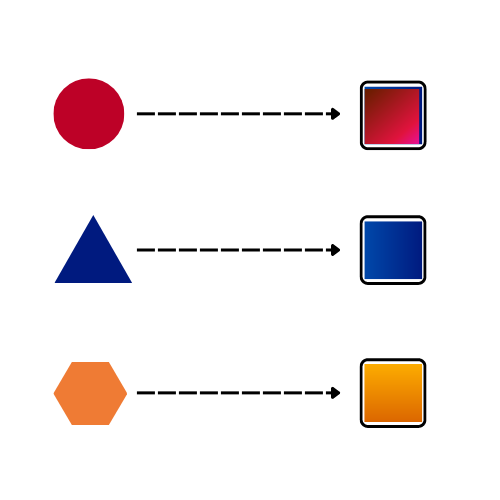


Figura 4 - Associazione Pattern → Response

Ecco un esempio: acquisto di uno smartphone.

* **Tag**: *acquistoSmartphone*
* **Pattern**:
  + “come posso acquistare uno smartphone?”
  + “dove posso comprare uno smartphone?”
  + “come effettuare l’acquisto di uno smartphone”
* **Response**:
  + “Puoi acquistare uno smartphone sul nostro sito”
  + “Puoi comprare uno smartphone sul sito o in un negozio fisico”
  + “Visita il nostro sito per acquistare uno smartphone”

Data una domanda dell’utente, questa viene analizzata e classificata in base al training data.

Ovviamente più il training data è ampio, più il chatbot è efficiente e riesce a interpretare il linguaggio naturale.

### **1.3.1 – FONDAMENTI DI NATURAL LANGUAGE PROCESSING PER LO SVILUPPO DI UN AGENTE CONVERSAZIONALE**

Il Natural Language Processing (NLP) è un ramo di linguistica, informatica e intelligenza artificiale che si occupa di fornire ai computer la capacità di comprendere parole pronunciate e testo, come l’essere umano.

Di seguito sono riportati i campi principali della NLP:

1. **Natural language generation( (*NLG*)**: È una tecnologia software che trasforma automaticamente i dati in inglese semplice.
2. **Natural language understanding (*NLU*)** : è una branca dell'elaborazione del linguaggio naturale (NLP) che aiuta i computer a comprendere e interpretare il linguaggio umano suddividendo il discorso nelle sue parti costitutive. Tuttavia, NLU va oltre il riconoscimento vocale per capire cosa l'utente sta cercando di comunicare con le sue parole.
3. **Natural language interaction (*NLI*)** : per interagire con qualsiasi dispositivo o servizio connesso in modo umano, NLI riunisce una variegata raccolta di principi del linguaggio naturale.

Immagine che contiene testo, elettronico

Descrizione generata automaticamente

Figura 5 - Campi NLP

Un algoritmo NLP per elaborare il linguaggio naturale comprende alcune tecniche particolari che fondono l’informatica, la grammatica e la logica: ***tokenization, stemming, lowering and ignore punctuation, lemmatization.***

#### **1.3.1.1 – TOKENIZATION**

Per fare in modo che una macchina apprenda il linguaggio naturale, non possiamo dare in input interi testi poiché risulterebbe davvero complesso addestrare il modello di apprendimento, ecco che entra in gioco la tokenization.

La tokenization trasforma un dato non strutturato, un testo, in un insieme di dati numerici discreti che viene computato dal computer.

La ***tokenization*** è una tecnica NLP, è la chiave per lavorare con dati testuali e consiste nella suddivisione di un testo in piú pezzi, detti *token*. Possiamo applicare la tokenization per frasi o per parole.

Per quanto riguarda la **tokenization per frasi**, dato un testo con *X* frasi, avremo *X* token, ognuno contenente una delle frasi del testo in input.

**La tokenization per parole** invece: dato un testo con *N* parole, avremo *N* token differenti.

È banale dire che *,* quindi una tokenization per parole risulta essere più precisa.

**ESEMPIO:**

***“I computer sono incredibilmente veloci, accurati e stupidi. Gli uomini sono incredibilmente lenti, inaccurati e intelligenti. L'insieme dei due costituisce una forza incalcolabile.”***

Tokenization per frasi:

1. I computer sono incredibilmente veloci, accurati e stupidi
2. Gli uomini sono incredibilmente lenti, inaccurati e intelligenti.
3. L'insieme dei due costituisce una forza incalcolabile.

Tokenization per parole:

1. I
2. Computer
3. Sono
4. Incredibilmente
5. Veloce
6. Accurati
7. E
8. Stupidi

Nonostante sia una tecnica ottimale, ha delle limitazioni: possiamo applicare la tokenization ad alfabeti che usano spazi e punteggiatura ma risulta essere molto difficile applicarla su lingua quali *cinese, coreano, giapponese, arabo.*

In particolare, la lingua araba ha una morfologia di linguaggio complessa: una singola parola potrebbe contenere piu token e significati differenti, come la parola “عقد”:



Figura 6 - I sei significati di عقد

#### **1.3.1.2 – STEMMING**

Lo **stemming** è una tecnica che riconduce un termine dalla sua forma flessa alla radice.

Ad esempio, le parole “*computer*”, “computare”, “computazionale”, “computabile”, sono ricondotte alla radice *“comp”.*

Lo stesso vale per le parole al singolare, plurale, maschile e femminile: “anziana”, “anziano”, “anziani”, “anziane” avranno la stessa radice “anzian”.

Come possiamo immaginare il risultato dello stemming è quasi sempre una parola senza significato.

Questa tecnica viene applicata per poter addestrare la macchina a riconoscere parole simili, con stesso significato e associabili a una radice comune.

La lingua italiana è composta da molte parole con radice comune, al contrario di quella inglese ad esempio: questo potrebbe rendere l’addestramento del nostro sistema più complicato ma ci sono *Stemmer* in grado di farlo.

Applicando lo stemming potremmo ricadere in delle incongruenze: date le parole *“universo”* e *“universitá”* la radice comune è *“univers”* ma hanno un significato completamente differente oppure *“pianta*” e *“pianto”*. Ciò comporta la perdita di significato di entrambi i termini.

Se paradossalmente avessimo un testo con molte parole simili all’esempio precedente, avremmo un output totalmente sconnesso.

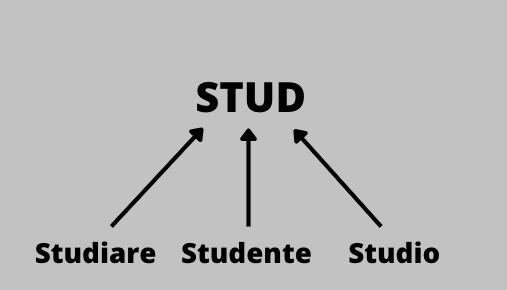


Figura 7 - Stemming

#### **1.3.1.3 – LOWERING AND IGNORE PUNCTUATION**

La tecnica più semplice ma anche molto efficace è il ***lowering***: consiste semplicemente nel trasformare tutti i caratteri scritti in maiuscolo in minuscolo.

Applicando questa tecnica si evita di memorizzare la stessa parola più volte, poiché per l’essere umano “*casa*”, “Casa”, “CASA” sono la stessa parola ma per un elaboratore sono 3 parole differenti.

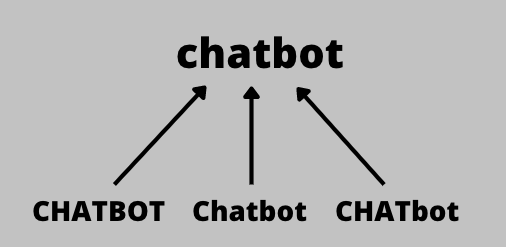


Figura 8 - Lowering

La tecnica di ***ignore punctuation*** consiste nell’escludere tutta la punteggiatura da un testo, in modo da analizzare solamente le parole. Questa tecnica viene combinata con il lowering per semplificare il training data.



Figura 9 - Lowering + Ignore Punctuation

#### **1.3.1.4 – LEMMATIZATION**

La **lemmatization** è una tecnica NLP simile allo stemming: riconduce diverse forme flesse alla stessa radice ma a differenza dello stemming cerca trovare una radice di senso compiuto.

Riprendendo l’esempio dello stemming, se abbiamo “*computer*”, “computare”, “computazionale”, “computabile”, sono ricondotte alla radice *“computer”.*

L’applicazione dello stemming non è errata ma come è intuibile usare la lemmatization potrebbe essere più corretto, ovviamente resta una scelta dello sviluppatore quale tecnica utilizzare.

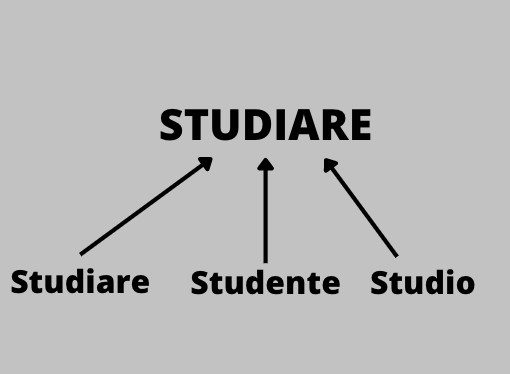


Figura 10 – Lemmatization

# **CAPITOLO 2 – DEFINIZIONE DEL DOMINIO APPLICATIVO E REQUISITI DI ORIENTABOT**

Come accennato nell’abstract, ci troviamo nell’ambito universitario.

L’idea è quella di sviluppare un chatbot che riesca a supportare gli studenti universitari, cercando di fornire risposte adeguate e risolvere i problemi di chi lo utilizza.

**OrientaBot** è esclusivo per l’Università Parthenope di Napoli, tutte le informazioni fornite riguardano quest’ultima. Le informazioni sono state estrapolate da esperienze personali, siti dell’università, moduli online e altro e sono accuratamente e completamente affidabili.

## **2.1 – SCENARI**

Di seguito si elencano alcuni scenari del software sviluppato:

* Peter, **studente** **universitario**, sta per iniziare la sua attività di tirocinio. Peter non ha molte informazioni e non sa come iniziare, dovrebbe affrontare 1 ora di treno per parlare con la segreteria e cercare di reperire qualche informazione utile. Peter decide di usare OrientaBot: scrive la sua richiesta e riceve ciò che voleva sapere in breve tempo.
* Alessia, **liceale**, vuole visitare la sede del Centro Direzionale ma non sa come arrivare. Prova a cercare in rete qualche info dettagliata ma si sente insicura.

Alessia chiede ad OrientaBot come arrivare alla sede del Centro Direzionale e ottiene il percorso e le modalitá di viaggio.

* Dario, **matricola** **universitaria**, vuole sapere a che ora inizia la lezione di Sistemi Operativi, prova chiedere a OrientaBot ma non riceve risposta: sfortunatamente il chatbot non è stato addestrato per rispondere a questa domanda.

Dario per contribuire allo sviluppo del chatbot può inviare una mail allo sviluppatore segnalando questo problema e ampliando quindi il set di domande e risposte del chatbot.

## **2.2 – REQUISITI FUNZIONALI E NON FUNZIONALI**

I seguenti requisiti funzionali (FR) e non funzionali (NFR) sono stati prefissati per lo sviluppo di OrientaBot:

**FR1:** Risposte adatte ai quesiti: il chatbot deve fornire risposte coerenti e consone all’argomento di cui si sta parlando.

**FR2:** Training data efficiente: il chatbot deve avere a disposizione un training data ampio, in modo da poter raccogliere più informazioni possibili e comportarsi di conseguenza.

**FR3:** Gestione delle incongruenze: quando l’utente immette una frase senza senso logico o parzialmente logica il chatbot deve fornire una risposta neutrale, evitando di rispondere con un altro argomento.

**FR4:** Correzione degli errori: se l’utente scrive una frase logica ma parzialmente errata grammaticalmente il chatbot deve riuscire a identificare di cosa si sta parlando e rispondere correttamente.

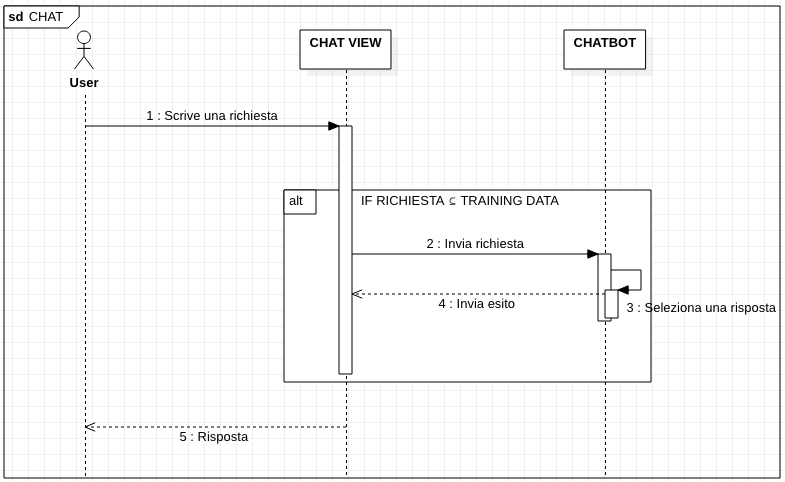
**NFR1:** Usabilità: il sistema dovrebbe essere intuitivo da usare e l’interfaccia utente dovrebbe essere semplice da capire.

**NFR2:** Conformità alle linee guida: la progettazione del sistema deve essere conforme alle linee guida sull’usabilità per il sistema operativo scelto.

**NFR3:** Il sistema deve garantire un corretto funzionamento e deve essere sempre aggiornato e attivo.

**NFR4:** Il sistema deve garantire il trattamento dei dati personali degli utenti ai sensi delle normative previste in merito

**NFR5:** Il linguaggio usato dal chatbot deve essere formale, sintatticamente e logicamente corretto.

****Figura 111 – Sequence Diagram Chatbot

Il seguente Sequence Diagram esplicita il funzionamento del Chatbot.

L’utente scrive una richiesta attraverso una chat, la richiesta viene inviata al chatbot che la analizza, verifica se coincide con qualche *tag* del training data, se così seleziona una risposta casuale tra le *responses* appartenenti al tag individuato nel dataset e la fornisce all’utente.

Se la richiesta non è inclusa nel training data, il Chatbot darà una risposta preimpostata e chiederà di porre un nuovo quesito.

L’interazione si conclude quando l’utente lascia la chat o chiude il programma.

## **2.3 – OBIETTIVI DI PROGETTAZIONE**

• **Response** **time**: OrientaBot deve fornire soluzioni in tempi brevi rispetto ad eventi straordinari che potrebbero verificarsi, ad esempio come nei sistemi real-time.

• **Maintenance** **cost**: OrientaBot dovrebbe garantire costi convenevoli di manutenibilità.

• **Security**: OrientaBot dovrebbe garantire la sicurezza dei dati nel rispetto della privacy di tutti gli enti utilizzatori rispettando le leggi dello stato in cui è attivo.

• **Portability**: OrientaBot dovrebbe essere un sistema multipiattaforma.

• **Extensibility**: OrientaBot dovrebbe essere tale da permettere l’aggiunta di nuove funzionalità e comportamenti in modo semplice, senza dover modificare l’intero sistema.

• **Availability**: OrientaBot dovrebbe garantire, per quanto possibile, il funzionamento del servizio.

• **Look** **and** **feel**: le interfacce di OrientaBot dovrebbero essere responsive e rispettare tutti i canoni di progettazione moderna.

• **Engagement**: OrientaBot dovrebbe invogliare l’utilizzo all’utente ogni volta che è in cerca di informazioni.

• **Accuracy:** OrientaBot dovrebbe fornire sempre risposte coerenti, corrette ed esplicative, evitando di creare confusione all’utente o diffondendo informazioni errate.

# **CAPITOLO 3 – STRUMENTI PER LO SVILUPPO**

OrientaBot è stato sviluppato interamente in **Python.**

Python è un linguaggio di programmazione dinamico, ***object oriented***, utilizzabile per molti tipi di sviluppo software.

Python offre un forte supporto all’integrazione con altri linguaggi di programmazione.

Il linguaggio Python ha una portabilità elevata: è utilizzabile su Windows, Linux/Unix, Mac OS X, OS/2, Amiga, palmari Palm e cellulari Nokia; è stato anche portato sulle macchine virtuali Java e .NET.

Python è distribuito con licenza Open-Source approvata dalla OSI: il suo utilizzo è gratuito e libero anche per prodotti commerciali.

Secondo uno studio di ***TPC INDEX,*** indicatore della popolarità dei linguaggi di programmazione, Python si posiziona al primo posto a partire dal 2020 ad oggi.

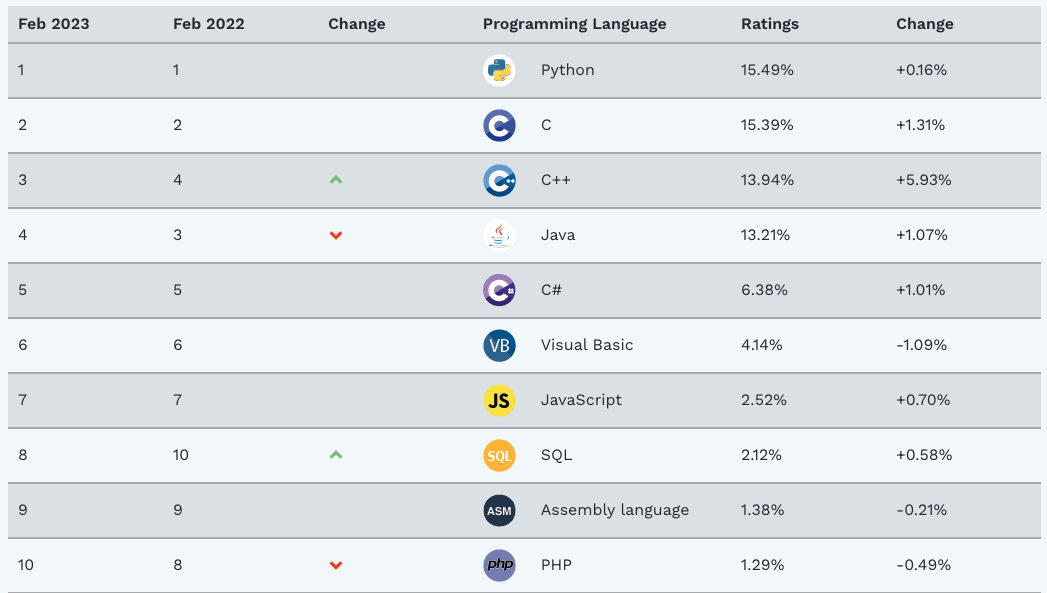


Figura 12 - TIOBE Top 10 Programming Language

## **3.1 – LIBRERIE E FRAMEWORK**

Oltre alla sua popolarità, Python è molto conveniente da utilizzare quando si ha a che fare con NLP, Machine Learning e modelli di addestramento.

Le librerie e i framework utilizzati per sviluppare OrientaBot sono molteplici:

1. **Natural Language ToolKit (NLTK)**
2. **NumPy**
3. **PyTorch**
4. **Tkinter**

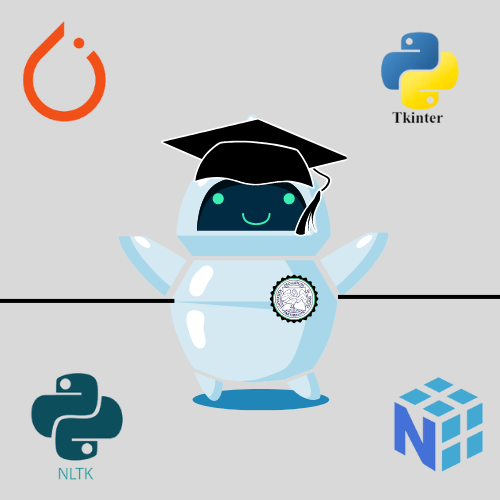


Figura 13 - OrientaBot Package

### **3.1.1 – NATURAL LANGUAGE TOOLKIT**

Natural Language ToolKit (NLTK) è un toolkit per il linguaggio naturale.

È un insieme di librerie per l’analisi simbolica e statistica nell’ambito dell’elaborazione del linguaggio naturale o NLP, scritte in Python.

Fu ideata e sviluppata da Steven Bird e Edward Loper al *Dipartimento di Informatica* dell’Università di Pennsylvania.

NLTK ha una vasta gamma di strumenti grafici e di analisi.

Questo toolkit supporta la ricerca del NLP, della linguistica, Machine Learning, Intelligenza Artificiale.

NLTK fornisce interfacce semplici da utilizzare e strumenti per l’elaborazione di testi, tokenization, stemming, tagging, ragionamento semantico e tutto ciò che è necessario per l’implementazione di OrientaBot.

NLTK richiede le versioni di Python 3.7, 3.8, 3.9, 3.10 o 3.11.

Per installare NLTK su un Sistema Operativo Unix Like basta eseguire questo comando da terminale:

1. Installazione NLTK: *run pip install --user -U nltk.*
2. Test installation: *run python* e digitare *import nltk.*

Per installare NLTK su un Sistema Operativo Windows:

1. Installazione NLTK: <https://pypi.python.org/pypi/nltk>
2. Test installation: *Start > Python38* [assumendo che la versione Python sia la 3.8] e digitare *import nltk.*

### **3.1.2 – NUMPY**

NumPy è una libreria Python Open Source per il supporto scientifico.

NumPy è una delle librerie più popolari e importanti per la scienza dei dati e il machine learning in Python.

Offre una suite di funzioni e strumenti avanzati per lavorare con array N-dimensionali come la creazione di array, manipolazione dei dati, operazioni puntuali e spaziali, selezione di elementi, aggregazione dei dati e tanto altro.

NumPy è molto performante rispetto alle alternative offerte nativamente da Python, questo lo rende indispensabile per le applicazioni che lavorano con una grande mole di dati e informazioni.

Inoltre, questa libreria è utilizzata in molti altri package scientifici, di Machine Learning e NLP come *Pandas, MatplotLib, TensorFlow, Tkinter, SciPy.*

NumPy è efficiente e ha una complessità computazionale minima poiché gli array NumPy sono oggetti e per il linguaggio Python significa elevato risparmio nell’allocazione e deallocazione.



Figura 14 - Funzioni di NumPy

### **3.1.3 – PYTORCH**

**PyTorch** è una libreria Open Source per il Machine Learning, Deep Learning e il Natural Language Processing in Python.

È stato sviluppato da *Meta AI* ed è diventato uno dei framework di deep learning più popolari e utilizzati nell'industria dello sviluppo software e nella ricerca accademica.

PyTorch offre molte caratteristiche che lo distinguono da altri framework di deep learning, tra cui:

1. **Integrazione con le operazioni sui tensori**: PyTorch è costruito sulle operazioni sui tensori e offre un'interfaccia user-friendly per le operazioni di deep learning. I tensori PyTorch sono simili agli array NumPy ma possono essere utilizzati anche su una GPU NVIDIA compatibile con CUDA.
2. **Dynamic Computational Graph:** PyTorch utilizza un grafo computazionale dinamico, che consente agli utenti di modificare il grafo durante l'esecuzione del modello. Questa caratteristica lo rende ideale per prototipazione rapida e sperimentazione.
3. **Supporto per la distribuzione**: PyTorch offre un supporto nativo per la distribuzione, che consente agli utenti di eseguire modelli su più GPU o nodi di calcolo.
4. **Facile da usare e integrarsi con altre librerie:** PyTorch è progettato per essere intuitivo e facile da usare, e si integra bene con altre librerie popolari come NumPy, Pandas e MatplotLib.

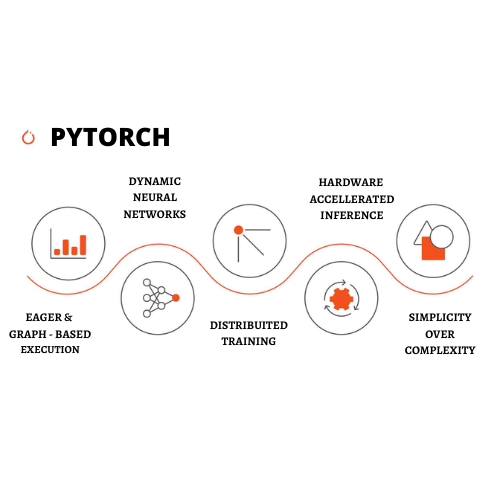


Figura 15 – PyTorch

### **3.1.4 – TKINTER**

Python ha molti framework per le *Graphic User Interface* ma Tkinter è l’unico framework built-in di Python.

Tkinter è un modulo per la creazione di interfacce grafiche utente (GUI) in Python, cross-platform, portabile e sempre aggiornato.

È stato sviluppato per la prima volta per la versione Python 1.5 ed è uno dei moduli GUI più antichi e diffusi per Python. Tkinter fornisce una serie di widget come pulsanti, caselle di testo, menú e altri elementi grafici che possono essere utilizzati per costruire interfacce utente complete, funzionali e semplici da utilizzare per tutti i tipi di utenti.

Gli elementi visivi sono renderizzati usando elementi nativi del Sistema Operativo, ciò significa che le applicazioni sviluppate con Tkinter potrebbero essere differenti a seconda del SO che si utilizza.

Tkinter è ideale per la creazione di applicazioni desktop e di prototipi, ma potrebbe essere anche una buona soluzione per progetti più complessi che richiedono una maggiore flessibilità e personalizzazione.

L’uso di Tkinter per OrientaBot è fondamentale: attraverso esso si è creata un’interfaccia utente smart, semplice e intuitiva, sostituendo l’interfaccia da linea di comando che non è usabile da tutti ed è molto più complessa e difficoltosa da assimilare per un utente medio che non ha dimestichezza con queste tecnologie.

La libreria è suddivisa nei seguenti moduli:

* Tkinter: modulo base che fornisce un insieme di widget sottoforma di classi.
* Tkinter.colorchooser: è il *color picker* nativo del sistema.
* Tkinter.commondialog: contiene la classe *Dialog* usata in altri moduli.
* Tkinter.filedialog: implementa procedure per selezione di file e cartelle.
* Tkinter.font: contiene una classe, *Font,* usata per gli stili testuali.
* Tkinter.messagebox: implementa procedure per creare pop-up window.
* Tkinter.scrolledtext: consente la visione di testi lunghi con ***scroll bar.***
* Tkinter.dnd: modulo sperimentale che implementa il Drag and Drop.
* Tkinter.ttk: modulo che effettua *l’override* dei widgets presenti nel modulo base, implementa anche nuove classi come *Style.* Ttk punta a rendere le applicazioni Tkinter più moderne e originali.

L’installazione di questo framework è immediata, basta aprire il terminale e digitare:

* ***pip install tk***

Questo comando inizierà il download e l’installazione dei package di Tkinter. Una volta terminato, verrà visualizzato un messaggio di conferma installazione.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 16 - Tkinter Install

# **CAPITOLO 4 – SVILUPPO DELL’APPLICAZIONE**

OrientaBot nasce da un’esigenza comune di studenti universitari nel reperire informazioni velocemente, evitando di fare ricerche estenuanti e senza successo.

Tutte le informazioni raccolte e fornite dal chatbot sono aggiornate all’anno *2022/2023,* ciò implica che OrientaBot necessita di una continua supervisione e aggiornamento.

Oltre alle tecnologie citate nel capitolo precedente, un elemento essenziale è sicuramente il *training data*, scritto in **JSON (JavaScript Object Notation)** che contiene tutte le info che il bot elabora.

Sul training data è applicata la tokenization, il lowering, lo stemming, e la ignore punctuation.

Il training data sviluppato è un ***dizionario*** con un unico array *intents* che contiene la tripla **tag – patterns – responses.**

L’idea è quella di indicizzare tutte le possibili frasi che l’utente immette e taggarle, in modo tale che il chatbot le possa riconoscere e dare una risposta.

Per creare il training data sono state ispezionate molte fonti: siti web Parthenope, gruppi universitari, social network, esigenze personali e problemi ricorrenti di cui ho sentito parlare o ho affrontato durante il mio percorso.

Immaginiamo che l’utente voglia chiedere al chatbot come funziona la seduta di laurea:

* I possibili *pattern* per questa richiesta potrebbero essere:
  + *"Come funziona la seduta di laurea?"*
  + *“Come si svolge la seduta di laurea?”*
  + *“Mi spieghi come è strutturata la seduta di laurea?”*

Attraverso questi pattern OrientaBot proverà a ricercare un tag che soddisfa il quesito e restituisce una *response*.

Vediamo meglio nel dettaglio come avviene ciò.

Dati i nostri *patterns*, vogliamo addestrare il nostro modello che sarà una ***rete feed forward neural con due hidden layers.***

Non possiamo far elaborare interamente tutti i pattern, in qualche modo dobbiamo convertire le nostre stringhe in un vettore che contiene numeri e per fare ciò useremo una ***bag of words.***

Per usare una bag of word dobbiamo raccogliere tutte le diverse parole dei patterns in un array.

Applico lo stemming a tutte le parole, cioè risalgo alla radice di ogni termine e inserisco tutte le stringhe in un array e le spezzo,

ad esempio la frase “*come funziona la seduta di laurea?”* diventa un array del genere: [“come”, “si”, “svolge”, “la”, “seduta”, “di”, “laurea”, “?”].

Questo array, che chiameremo “***all words***”, contiene tutte le parole del nostro training data e lo useremo per creare la bag of words.

Per ogni pattern creeremo un array della stessa dimensione dell’array *all words.*

Se le parole contenute nei pattern sono incluse nell’array di tutte le parole inseriremo un 1 nella posizione dell’array del pattern rispettivo alla posizione dell’array *all words*, se non esiste allora inseriremo uno 0 per indicare che quella parola non è inclusa nell’array all words.

Alla fine avremo un insieme di dati ***X*** per l’addestramento e per ogni istanza abbiamo un tag che viene indicizzato, ad esempio il tag “ciao” ha label 0, il tag “tesi” ha label 1 e così via.

Le label dei tag sarà il vettore Y per il nostro modello che addestrerà **la *feed forward neural net*.**

Il seguente esempio renderà chiara l’idea:

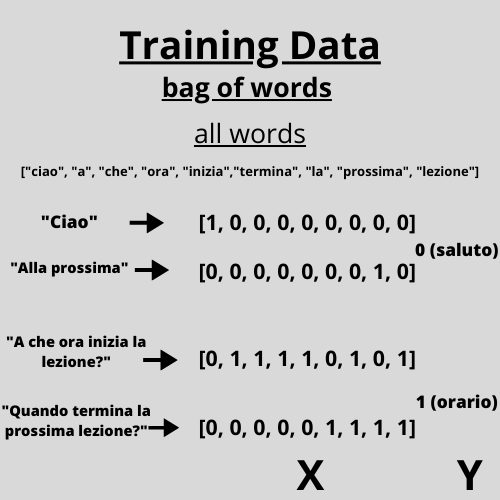
****

Figura 17 - Creazione della bag of words

X e Y saranno gli input per addestrare la nostra rete, alla fine avremo **probabilità** diverse per ogni label.

## **4.1 – ALGORITMO DI ORIENTABOT**

Il nostro algoritmo **NLP** è così articolato:

1. Trasformiamo l’input dell’utente in singole frasi.
2. Applichiamo la tokenization, splittiamo la frase in un array di token dove ogni termine di una frase è un elemento dell’array.
3. Applichiamo il lowering, trasformando tutte le lettere maiuscole in minuscole.
4. Applichiamo lo stemming, in modo da accumulare parole con la stessa radice in una sola istanza dell’array.
5. Applico la ignore punctuation per eliminare tutta la punteggiatura dell’input.
6. Dall’array risultante calcoliamo la bag of words ottenendo il vettore X.

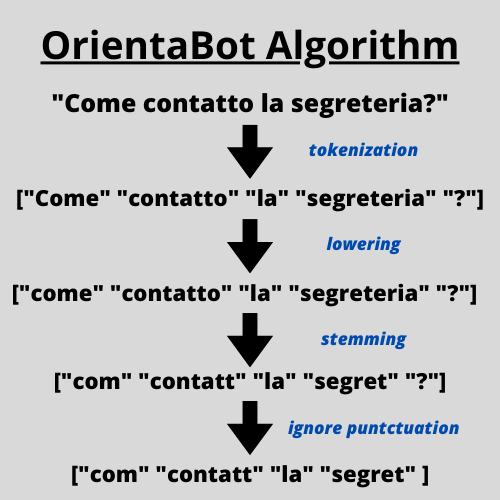


Figura 18 - Passi dell'algoritmo NLP

Con un ciclo for su *intents* [intents è un dizionario con un solo array con tutti i tag, patterns e responses] salviamo tutti i tag in un array.

Facciamo lo stesso su tutti i pattern e gli applichiamo la tokenization e li inseriamo nell’array *all words.*

Successivamente nella lista *xy* mettiamo i pattern tokenizzati e la label corrispondente.

Dopo la tokenization vogliamo applicare lowering, stemming e ignore punctuation.

Definiamo una *ignore\_words* ed eliminiamo tutta la punteggiatura.

All\_words è una lista dove applico lo stemming ed escludo la punteggiatura attraverso l’ignoring punctuation.

Ora ordiniamo *all words* ed eliminiamo le parole ripetute, in modo tale nell’array ci sia una sola occorrenza per parola.

Ora vogliamo creare la bag of words: creo una lista *X\_train* e una *Y\_train.*

In X mettiamo le bag of words e in Y le tag.

Con un ciclo sulla lista xy (pattern\_sentence, tag) vogliamo creare una bag of words chiamando la funzione bag of words (sviluppata in *ntlk utils*) che prende in input la pattern\_sentence e all\_words).

Aggiungiamo la bag in *X\_train.*

Ora indicizziamo tutte le tag e le inseriamo in un array chiamato label e le inseriamo in *Y\_train.*

Dopo convertiamo questi due in un np array.

Così abbiamo creato il nostro training data personalizzato e possiamo addestrare OrientaBot.

È possibile visionare, scaricare e testare il codice sorgente su GitHub visitando la pagina <https://github.com/dom0000D/Bot>.

## **4.1 – SVILUPPO DI UNA RETE NEURALE**

Una feed forward neural net (FNN) è una rete neurale artificiale dove le connessioni tra i nodi non creano un ciclo.

La FNN per questo motivo è diversa dal suo discendente: la ***recurrent neural network (RNN):*** una RNN è una classe di reti neurali artificiali dove le connessioni tra i nodi possono creare un ciclo, consentendo all’output di alcuni nodi di influenzare l’input successivo degli stessi nodi.

Le feed forward neural net sono state le prime reti neurali ideate: in questo modello le informazioni si muovono in una sola direzione, in **avanti**, dai nodi in input, attraverso nodi nascosti e verso nodi di output.

Ciò esplicita il motivo del loro nome, che in italiano significa *alimentare in avanti*.

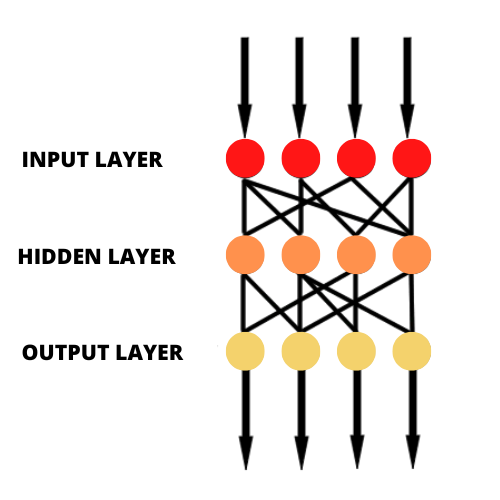


Figura 19 - FeedForward neural net con 1 hidden layer

La tipologia più semplice di rete neurale è un ***perceptron*** *net* con un singolo hidden layer che consiste in un singolo strato di nodi di output.

In Machine Learning, il *perceptron* o *percettrone* è un algoritmo per l’apprendimento supervisionato di classificatori binari.

Un classificatore binario è una funzione che può decidere se un input appartiene o meno a una classe specifica.

È un classificatore **lineare**, ovvero un algoritmo di classificazione che effettua previsioni sulla base di una *funzione predittiva lineare* che combina un insieme di pesi col vettore delle ***feature***: una feature è una proprietà individuale misurabile o una particolarità di un fenomeno.

Gli input vengono alimentati (*feed*) direttamente alle uscite tramite una serie di pesi.

La somma dei prodotti dei pesi e degli input viene calcolata per ogni singolo nodo, se il valore è al di sopra di una certa soglia fissata (*thresh)* il neurone si attiva e assume il valore attivato, tipicamente 1.

Se invece non è al di sopra della soglia, il neurone assume il valore di disattivato, tipicamente -1.

I percettroni possono essere addestrati da un semplice algoritmo di apprendimento chiamato ***delta rule:*** esso calcola gli errore tra l’output calcolato e i dati di output del campione e li usa per creare un miglioramento dei pesi, implementando una forma di ***gradient descent.***

Gradient descent (chiamata anche discesa più ripida ) è un algoritmo di **ottimizzazione iterativo** per trovare un minimo locale di una funzione differenziabile.

L'idea è di fare passi ripetuti nella direzione opposta al gradiente della funzione nel punto corrente, perché questa è la direzione della discesa più ripida.

Al contrario, un passo nella direzione del gradiente porterà a un massimo locale di quella funzione; la procedura è quindi nota come salita del gradiente.

La discesa del gradiente è generalmente attribuita ad Augustin-Louis Cauchy, che la suggerì per la prima volta nel 1847.

I percettroni a strato singolo sono in grado di apprendere solo schemi ***separabili linearmente.***

La separabilità lineare è una proprietà di due insieme di punti.

È facilmente evidenziabile in due dimensioni, usando un piano euclideo: immaginiamo un insieme di punti colorati di rosso e un altro insieme colorato di blu, questi due insiemi sono linearmente separabili se esiste almeno una retta nel piano con tutti i punti blu su un lato della retta e tutti i punti rossi dall’altro lato della retta.

Il problema di determinare se una coppia di insiemi è linearmente separabile è molto importante per la statistica e il machine learning: l’obiettivo è classificare certi tipi di dati presi da un insieme.

Nel 1969 in una famosa monografia intitolata *Perceptrons* , Marvin Minsky e Seymour Papert dimostrarono che era impossibile per una rete di perceptron a singolo strato apprendere una funzione **XOR.**

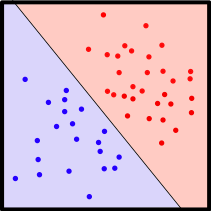


Figura 20 - Separabilità Lineare

Un esempio pratico potrebbe essere un classificatore di e-mail spam: una volta trovata la retta che divide l’insieme di mail ordinarie (punti rossi) e-mail spam (punti blu), alla ricezione di una nuova mail potremmo subito classificarla in spam o meno.

Il problema è quando le classi non sono linearmente separabili.

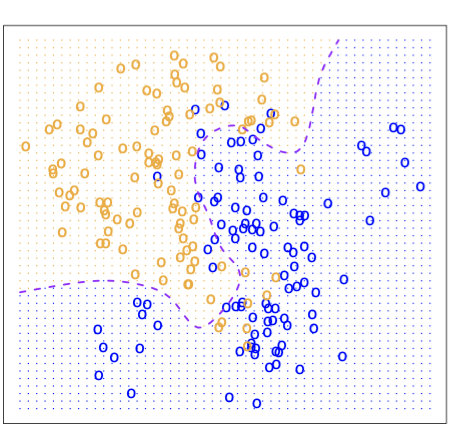


Figura 21 - Problema non separabile linearmente

Un esempio è il dataset dei fiori *Iris*.

Ci sono 150 fiori Iris che sono di 3 tipologie diverse: versicolor, setosa e virginica.

Ciascuno di questi fiori ha 4 input: lunghezza e larghezza del petalo, lunghezza e larghezza del sepalo.

Dati questi 150 fiori, l’obiettivo è classificare i diversi Iris. Per fare ciò si utilizza una *matrice di confusione* che ha tante righe e colonne quante sono le classi da predire, nel caso di Iris ho 3 classi, ottenendo quindi una matrice *3x3.*

Sulle righe indico il valore dell’output corretto da predire, sulle colonne ho i dati predetti dal modello. Sulla diagonale avremo le risposte corrette e per calcolare la percentuale si sommano i valori della diagonali e si dividono per il numero di dati in input, 150 nel caso di Iris.

Sebbene una singola unità di soglia sia piuttosto limitata nella sua potenza computazionale, è stato dimostrato che reti di unità di soglia parallele possono approssimare qualsiasi funzione continua da un intervallo compatto di numeri reali nell'intervallo *[-1,1].*

Questo risultato può essere trovato in Peter Auer, Harald Burgsteiner e Wolfgang Maass*: "Una regola di apprendimento per approssimatore universale molto semplice costituite da un singolo strato di percettroni".*

Una feed forward neural net con un singolo hidden layer può calcolare un output continuo invece di una funzione a gradino.

Una scelta comune è la cosiddetta funzione logistica:

Con questa scelta, la rete single-layer è **identica** al modello di regressione logistica, ampiamente utilizzato nella modellazione statistica.

Le feed forward multistrato invece sono classi di reti costituite da più strati di unità computazionali, interconnesse in maniera feed forward: ogni neurone in uno strato ha connessioni dirette ai neuroni dello stato successivo.

Il teorema di *approssimazione universale* per le reti neurali afferma che ogni funzione continua che associa intervalli di numeri reali a un intervallo di output di numeri reali può essere approssimata da un percettrone multistrato.

Le reti multistrato utilizzano una varietà di tecniche di apprendimento, la più popolare è la ***backpropagation***.

Con la backpropagation i valori di output vengono confrontati con la risposta corretta per calcolare il valore di una funzione di errore predefinita.

Con varie tecniche, l'errore viene poi riportato attraverso la rete.

Utilizzando queste informazioni, l'algoritmo regola i pesi di ciascuna connessione per ridurre il valore della funzione di errore di una piccola quantità.

Dopo aver ripetuto questo processo per un numero sufficientemente elevato di cicli di addestramento, la rete di solito converge in uno stato in cui l'errore dei calcoli è piccolo. In questo caso si direbbe che la rete abbia appreso una certa funzione target.

In generale, il problema di insegnare a una rete a funzionare bene, anche su campioni che non sono stati utilizzati come campioni di addestramento, è un problema abbastanza sottile che richiede tecniche aggiuntive.

Ciò è particolarmente importante per i casi in cui sono disponibili solo numeri molto limitati di campioni di addestramento.

Il pericolo è che la rete sovradimensioni i dati di addestramento( ***overfitting*** ) e non riesca a catturare il vero processo statistico che genera i dati.

La teoria dell'apprendimento computazionale si occupa dell'addestramento dei classificatori su una quantità limitata di dati.

Nel contesto delle reti neurali una semplice euristica, chiamata ***early stopping***, spesso garantisce che la rete si generalizzi bene agli esempi non presenti nel set di addestramento.

In una rete neurale feed-forward con due hidden layer, ogni strato è costituito da più neuroni, che sono completamente connessi con i neuroni degli strati precedente e successivo.

Ogni neurone artificiale riceve uno o più segnali di ingresso e invia un valore ai neuroni dello strato successivo.

L'uscita y è una somma ponderata non lineare dei segnali di ingresso.

La non linearità si ottiene passando la somma lineare attraverso funzioni non lineari note come funzioni di attivazione.

Funzioni popolari di attivazione dei neuroni: l'unità lineare rettificata (ReLU), Sigmoid (Sigm) e Tanh.

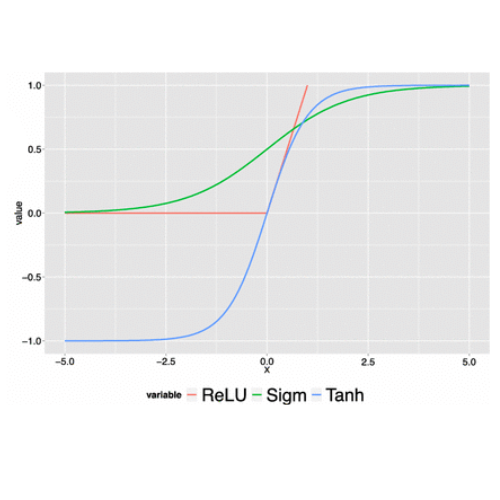


Figura 22 - Funzione di attivazione dei neuroni

In conclusione, qualsiasi grafo aciclico diretto può essere usato per una rete feed forward, con nodi designati come input e altri come output.

OrientaBot utilizza una rete neurale feed forward con 2 hidden layer: prende come input la *bag of word*, con un livello completamente connesso che ha il numero di ogni singolo pattern e poi 2 livelli nascosti.

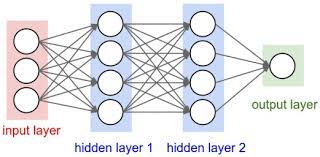


Figura 23 - FNN 2 hidden layers

A supporto del nostro modello abbiamo definito alcuni parametri che possono essere modificati a seconda delle necessità computative.

|  |  |
| --- | --- |
| **NUM\_EPOCHS** | 1000 |
| **BATCH\_SIZE** | 8 |
| **LEARNING\_RATE** | 0.001 |
| **INPUT\_SIZE** | Lunghezza bag of word |
| **HIDDEN\_SIZE** | 8 |
| **OUTPUT\_SIZE** | Lunghezza tag |

* **NUM\_EPOCHS**: quando il training set è sottoposto al modello si ha una epoch.
* **BATCH**: Il training set potrebbe essere troppo grande per essere elaborato tutto in una volta, lo dividiamo in sottogruppi chiamati batch.
* **BATCH\_SIZE**: Il numero di esempi contenuti in ogni batch è detto batch size.

Se ho 2000 esempi posso dividere il set in batch con 500 esempi ciascuno e quindi 1 epoch avrà 4 iterazioni.

Il numero di epoch ed il batch size influiscono sulla velocità di addestramento di un modello, ma anche sul suo modo di perfezionarsi.

* **HIDDEN\_SIZE**: La dimensione nascosta è il numero di caratteristiche dello stato nascosto.

Quindi aumentando la dimensione nascosta, calcoliamo la funzionalità più grande come output dello stato nascosto.

Questi parametri sono detti ***hyperparameter***: sono dei parametri i cui valori sono utilizzati per controllare il processo di apprendimento.

Il tempo necessario per addestrare e testare un modello può dipendere dalla scelta dei suoi iperparametri.

# **CAPITOLO 5 – INTERFACCIA E USABILITÀ**

OrientaBot ha come obiettivo quello di essere moderno, smart e alla portata di tutti, inoltre poiché è indirizzato a un’utenza molto tecnologica, deve essere all’altezza di ciò che si è già potuto vedere altrove.

Si è puntato a un design minimale, contemporaneo, semplice e immediato, con aspetti chiave già noti all’utente.

È stata scelta la seguente palette di colori così da garantire un aspetto moderno per l’utenza e consistenza alle varie schermate del sistema:



Figura 24 - Palette Interfaccia OrientaBot

Rispettivamente da sinistra verso destra:

*#000000, #FFC107, #CEC3AD, #F4E4CA, F7ECDA.*

I colori utilizzati sono stati scelti dopo una lunga selezione e tante considerazioni, raccogliendo il parere di più utenti e cercando di soddisfare le loro preferenze, in modo tale da non rendere OrientaBot un lavoro personale ma trasformarlo in un prodotto di tutti gli utenti che lo utilizzeranno.

Tra gli obiettivi futuri vi è proprio la possibilità di far scegliere all’utente quali colori utilizzare, in modo da personalizzare completamente l’esperienza.

Il font utilizzato è *Helvetica,* con dimensione di caratteri che va da un range di 18 a 40.

È stato scelto Helvetica poiché è un carattere familiare, non invasivo e approvato da grandi aziende come *Motorola, Panasonic, Toyota, Fifa, Fendi, Intel…*

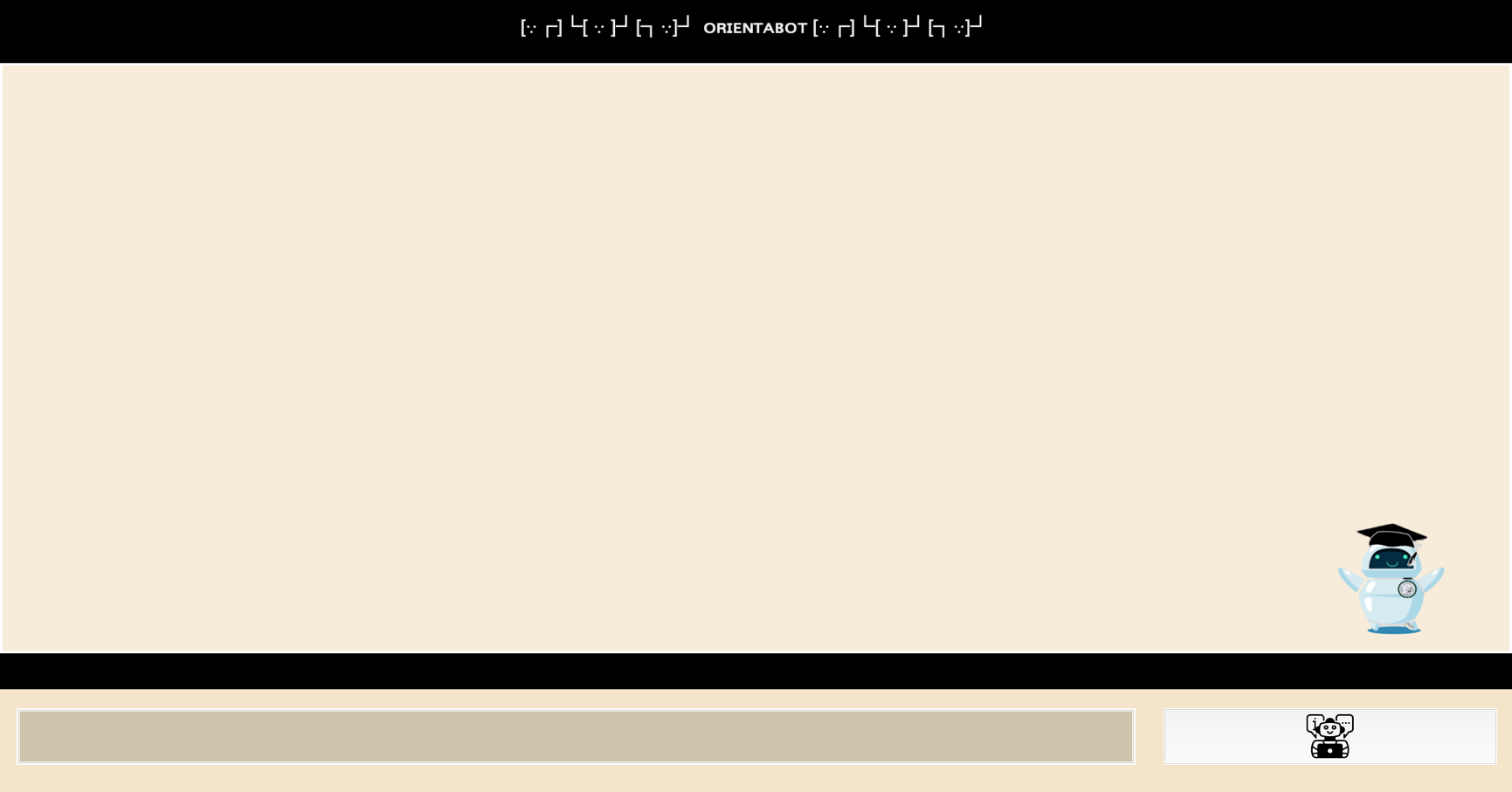
Vediamo nell’interezza come si presenta OrientaBot:

Figura 25 - OrientaBot GUI

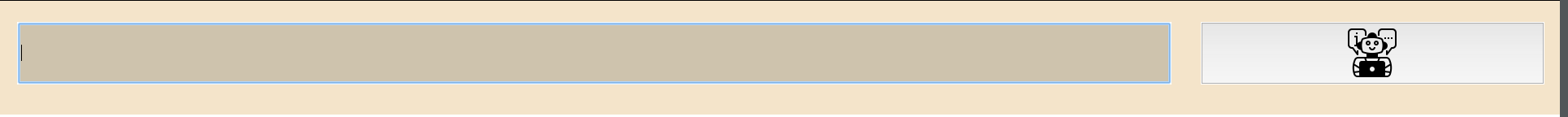




Figura 26 ­- Focus sulla barra di testo e pulsante di invio

Come possiamo osservare, OrientaBot presenta una classica interfaccia grafica di un chatbot, con una finestra di testo per l’interazione e un pulsante per inviare il testo.

È possibile inviare il testo anche col tasto Invio della tastiera per avere un’interazione ancora più rapida e semplice.

Per rendere l’interfaccia più moderna si è scelto di utilizzare un bottone classico ma con un’icona al posto della solita scritta “*invia”* e in basso a destra vi è l’avatar di OrientaBot: un robot con il classico cappello da laureato, il tocco, e lo stemma dell’Università degli Studi di Napoli Parthenope.

Tutte queste piccolezze sono funzionali per rendere l’interfaccia più riconoscibile, cercando di impattare nella mente dell’utente, in modo tale che alla visione dell’interfaccia di OrientaBot saprà subito identificarlo e per rafforzare l’identità e l’appartenenza all’Università.

OrientaBot è sviluppato interamente e unicamente per l’Università Parthenope.

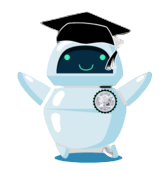


Figura 27 - Avatar OrientaBot

## **5.1 – TEST DI USABILITÀ**

Il test di usabilità è stato condotto dallo sviluppatore tesista, dal docente relatore e da altre persone che frequentano il mondo universitario o che lo frequenteranno presto.

In fasi successive sarà ampliato anche verso membri esterni, così da aver maggiori feedback e migliorare l’esperienza dell’utente.

Gli obiettivi del test riguardano il capire il livello di usabilità dell’interfaccia mostrata precedentemente in base ai seguenti sette princìpi dell’usabilità:

* **Adeguatezza al compito (suitability for the task).**
* **Auto-descrizione (self-descriptiveness).**
* **Conformità alle aspettative dell’utente (conformity with user expectations).**
* **Adeguatezza all’apprendimento (suitability for learning).**

* **Controllabilità (controllability).**
* **Tolleranza verso gli errori (error-tolerance).**
* **Adeguatezza all’individualizzazione (suitability for individualization).**

Per quanto riguarda il test effettuato dal sottoscritto, ho eseguito l’applicazione molte volte e testato la sua affidabilità, inserendo molteplici richieste, frasi correttamente scritte ma senza logica, frasi grammaticamente errate e verificando il comportamento del chatbot.

Ogni pattern è stato testato a fondo e man mano sono state aggiunte più frasi per dare la possibilità al chatbot di avere un apprendimento ottimale.

Per quanto riguarda i test effettuati dagli altri utenti, che non conoscevano il contenuto del training data e il funzionamento del chatbot, semplicemente si è utilizzato OrientaBot, provando a chiedere determinate informazioni, scrivendo particolari termini e frasi.

Da sottolineare che i test hanno avuto un’importanza fondamentale al fine della realizzazione del chatbot.

Senza un’accurata fase di testing è davvero difficile sviluppare un’applicazione di questo tipo poiché si rischia di lasciare incompleti molti punti senza esserne a conoscenza, per poi scoprirlo solo a run time.

Non escludo infatti che ci saranno pattern mancanti, sarà cura dello sviluppatore in primis e degli utenti in seguito, segnalare eventuali mancanze e necessità.

Il testing degli utenti è risultato molto utile, sia per verificare l’effettivo funzionamento di OrientaBot, sia per ampliare il data training con nuove domande e risposte da inserire.

Inoltre sono stati testati i tempi di risposta del Chatbot, ottenendo perfetti risultati nell’ordine di millisecondi per ogni risposta ed elaborazione.

Si rendono evidenti i seguenti interventi migliorativi indispensabili:

* Ulteriore rifinitura e miglioramento di UI e UX.
* Aggiornare e ampliare il data training frequentemente.
* Creare un box utenti per segnalare gli errori o per arricchire il training data.

Si rendono evidenti i seguenti interventi migliorativi possibili:

* Creare un effetto *on hover* sul bottone di invio.
* Impostare una conversazione predefinita attraverso delle scelte con il mouse.
* Rendere l’icona di OrientaBot interattiva.
* Aggiungere suoni personalizzati

## 

## **5.2 – VALUTAZIONE DI USABILITÀ**

La valutazione dell’usabilità è stata realizzata mediante un foglio Excel all’interno del quale ogni utente ha inserito il proprio voto relativo all’**esperienza** con OrientaBot e quello relativo a un sistema software che offre un servizio simile: il chatbot dell’Università degli Studi di Padova.

Con la seguente valutazione non si intende screditare minimamente il lavoro altrui, è solo un metodo per confrontare il lavoro svolto con uno in uso e già rilasciato al pubblico.

I voti sono stati tanti quanti gli aspetti di ognuno dei princìpi elencati nel paragrafo precedente, con un range che va da 0 a 4.

Successivamente è stata effettuata la media per ogni tester e per tutti gli aspetti di ogni principio per ottenere il voto medio per quel determinato principio di usabilità e infine è stato realizzato un grafico radar di confronto tra i due sistemi software.

Di seguito si allegano i risultati dei grafici radar di tale valutazione di usabilità:



Figura 28 - Valutazione Chatbot Università di Padova



Figura 29 - Valutazione OrientaBot

Come possiamo osservare dai grafici, complessivamente OrientaBot si presenta più completo, con una buona adeguatezza all’apprendimento e al compito.

OrientaBot ha dimostrato di essere conforme alle aspettative degli utenti, grazie alla sua interfaccia moderna e alla scelta di un linguaggio giovane e adatto alla fascia di utenti medi che useranno l’applicazione.

Per quanto riguarda la tolleranza verso gli errori, nel paragrafo successivo verrà analizzato il comportamento di OrientaBot in situazioni limite e particolari.

Questi risultati sono incoraggianti per il lavoro effettuato ma in futuro si auspica di migliorare quanto piú possibile OrientaBot, in modo tale da diventare un servizio effettivo e personale dell’Università degli Studi di Napoli Parthenope.

## **5.3 – ESECUZIONE DI ORIENTABOT**

In questa sezione analizzeremo l’esecuzione di OrientaBot, come risponde ai quesiti fatti dall’utente e come si comporta in situazioni particolari.

L’obiettivo di OrientaBot è quello di essere affidabile, sicuro e veloce. Non possiamo permettere che il chatbot dia informazioni errate dato che siamo in un contesto delicato come quello universitario.

Il training data realizzato presenta un numero elevato di quesiti che un utente generale potrebbe chiedere, vediamo qualche esempio.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 30 - Conversazione con OrientaBot n.1

Possiamo osservare una simulazione di conversazione tra un utente con OrientaBot.

Tutte le interazioni dell’user sono scritte in un linguaggio informale, evitando termini complessi, non appartenenti a un linguaggio quotidiano.

OrientaBot risponde esaustivamente alla richiesta dell’utente.

Osserviamo una particolarità: il chatbot cerca di portare l’utente sulla sua strada, servendo delle domande implicite, basta guardare la seconda interazione dove il sito della scuola non viene fornito esplicitamente ma se viene richiesto OrientaBot immediatamente fornisce tutte le info del caso.

Questa interazione è voluta appositamente dallo sviluppatore, per rendere ancora più realistica l’interazione con il chatbot, evitando di dare risposte lunghe e noiose e incuriosendo l’utente a fare più domande per scoprire se il chatbot è in grado di rispondere oppure no.

Osserviamo adesso questa conversazione:

Immagine che contiene testo

Descrizione generata automaticamente

Figura 31 - Interazione con OrientaBot n.2

Questa conversazione non è altro che una versione alternativa a quella precedente, con frasi differenti e risposte del chatbot diverse.

Da sottolineaare che non tutte le *responses* hanno più versioni e quindi alcune potrebbero avere la stessa risposta o l’algoritmo potrebbe selezionare in modo pseudo-casuale la stessa risposta piú volte consecutivamente.

Vi è la possibilità di selezionare il testo, copiare, incollare ed eseguire tutti gli shortcut da tastiera come CTRL + C, CTRL + V e tanto altro.

Inoltre possiamo scrollare il testo con la *scroll bar* posta alla destra del chatbot oppure usare la rotellina del mouse o utilizzare la feature *two finger scrolling* dei computer portatili.

Immagine che contiene testo

Descrizione generata automaticamenteVediamo altri esempi di conversazione:

Figura 32 - Interazione con OrientaBot n.3

Ecco un caso di diversificazione delle risposte da parte di OrientaBot: alla stessa richiesta vi è una risposta diversa.

Da notare che il primo quesito non è grammaticalmente una domanda a causa dell’assenza del punto interrogativo ma grazie all’ignore punctuation il chatbot riesce comunque a interpretare ciò che è stato chiesto, a testimonianza della potenza delle funzioni dell’elaborazione del linguaggio naturale.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 33 - Interazione con OrientaBot n.4

Nel caso in cui un utente dovesse scrivere qualche parola errata, il chatbot dovrebbe interpretare comunque il quesito e cercare di dare una risposta.

Ovviamente il quesito dell’utente deve avere una logica, risulterebbe quasi impossibile per il chatbot interpretare un discorso senza senso.

Negli esempi successivi è illustrato quanto descritto:

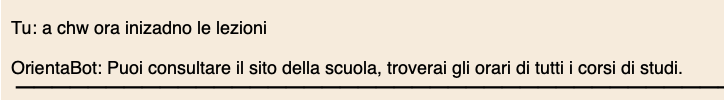


Figura 34 - Interazione con OrientaBot n.5

Anche se erroneamente si inseriscono parole con una doppia sillaba in meno o in più, OrientaBot riesce a interpretare il quesito:

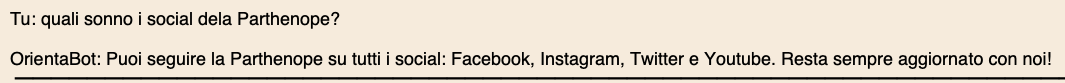


Figura 35 - Interazione con OrientaBot n.6

Unendo parole errate e parole grammaticalmente scorrette, ecco il risultato:

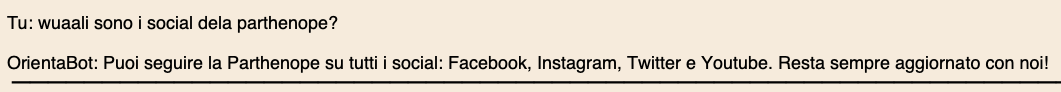


Figura 36 - Interazione con OrientaBot n.7

OrientaBot è stato correttamente addestrato per situazioni di questo tipo che possono capitare frequentemente.

Ovviamente ci sono casi in cui o il quesito dell’utente non è presente nel training data o la frase risulta essere incomprensibile.

Quando ciò accade il chatbot risponde con un messaggio preimpostato.

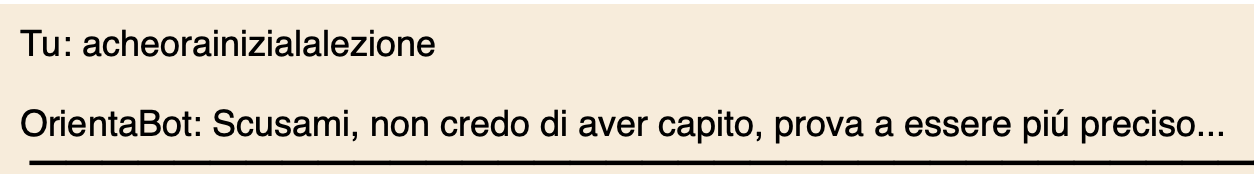


Figura 37 - Interazione con OrientaBot n.8

Immagine che contiene testo

Descrizione generata automaticamente

Figura 38 - Interazione con OrientaBot n.9

L’obiettivo è che il modello capisca di aver ricevuto in input qualcosa a lui sconosciuto ed evitare che risponda in maniera casuale, magari associando la frase in input dell’utente con qualche tag del training data.

Certamente non è escluso che ciò accada, vediamo un esempio:

Immagine che contiene testo

Descrizione generata automaticamente

Figura 39 - Interazione con OrientaBot n.10

In questo caso il pattern “*a che ora”* è simile ad alcuni presenti nel training data ed erroneamente è stato associato al tag “segreteria”. Questo è un errore che potrebbe capitare ma c’è da sottolineare che l’obiettivo di OrientaBot è quello di aiutare gli studenti universitari in cerca di informazioni e non ricevere quesiti senza un senso logico.

Ciò non toglie che uno dei punti futuri è evitare questi scenari, in modo da distinguere al 100% che una frase non è presente nel training data o è priva di senso logico.

## **5.4 – SVILUPPI FUTURI DI ORIENTABOT**

OrientaBot ha come scopo quello di durare nel tempo e di diventare un elemento essenziale nella vita degli studenti della Parthenope.

Per fare ciò bisogna puntare sempre più in alto, in modo da rendere il chatbot competitivo e invogliare gli utenti ad utilizzarlo.

Tra gli sviluppi futuri vi è sicuramente la versione mobile di OrientaBot: questa risulterebbe più smart, alla portata di tutti e sempre presente quando vi è necessità.

OrientaBot Mobile punterà ad essere rilasciata in tutti gli store e per tutti i sistemi operativi mobile più famosi, in modo da essere più diffusa possibile.



Figura 40 - OrientaBot Mobile

OrientaBot Mobile sarà un servizio gratuito, senza pubblicità e utilizzabile anche senza una connessione internet. Richiederà aggiornamenti periodici per ampliare il training data ma non avrà una dimensione elevata e non impatterà negativamente sulle prestazioni generali del sistema.

Per implementare OrientaBot Mobile verrà creato un gruppo di lavoro, in modo da diversificare i compiti e rendere lo sviluppo più semplice e veloce.

Un’altra feature molto ricercata è lo *speech to text*: OrientaBot riuscirà ad analizzare le nostre parole e trasformarle in testo, in modo che l’utente non dovrà più scrivere il proprio quesito ma gli basterà enunciarlo a voce.

Lo speech to text è un software di riconoscimento vocale che permette il riconoscimento e la traduzione del parlato in testo attraverso la linguistica computazionale.

È conosciuta con il nome di riconoscimento vocale o riconoscimento vocale computerizzato.

Applicazioni, strumenti e dispositivi specifici possono trascrivere flussi audio in tempo reale per visualizzare un testo e utilizzarlo.

Lo speech to text avviene tramite un complesso modello di Machine Learning che prevede diversi passi:

1. Quando la bocca emette un suono per pronunciare una parola, emette anche una serie di **vibrazioni**. La tecnologia di trascrizione vocale agisce rilevando queste vibrazioni e traducendole in linguaggio digitale tramite un convertitore da analogico a digitale.
2. Il convertitore da analogico a digitale riceve i suoni da un file audio, misura le onde con grande precisione e le filtra per distinguerne i relativi suoni.
3. I suoni vengono quindi **segmentati** in centesimi o millesimi di secondo e in seguito abbinati a fonemi. Un fonema è un'unità di suono che distingue una parola da un'altra in qualsiasi lingua. Ad esempio, nella lingua inglese ci sono circa 40 fonemi.
4. I fonemi vengono quindi fatti passare attraverso una rete tramite un modello matematico che li confronta con proposizioni, parole e frasi conosciute.
5. Il risultato viene quindi presentato sotto forma di testo o come domanda basata sul computer a seconda della versione più probabile dell'audio.

Utilizzare una tecnica del genere renderebbe OrientaBot più efficiente, facendo risparmiare tempo agli utenti meno avvezzi alla tecnologia, migliorando l’esperienza generale degli utenti.

Tuttavia, come ogni tecnica avanzata, lo speech to text non è affidabile al 100 %: sebbene la tecnologia di dettatura sia uno strumento potente, è ancora in fase di miglioramento e questo significa che ci sono ancora delle lacune nelle sue prestazioni generali.

Poiché produce solo testo parola per parola, si potrebbe ottenere una trascrizione imprecisa o in cui mancano alcune parti specifiche.

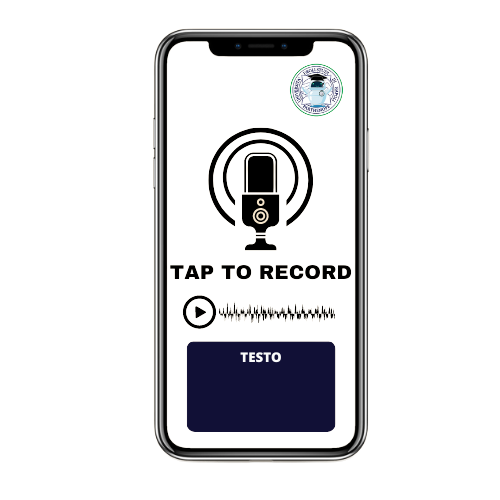


Figura 41 - OrientaBot Speech to Text

Un’altra funzionalità a cui si punta è l’inversa di quella appena descritta: il ***text to speech (TTS)***.

Questa funzione è molto interessante, renderebbe OrientaBot simile agli assistenti digitali con cui interagiamo ogni giorno, come Alexa, Google Home o Siri.

Il TTS aumenterebbe il piacere di utilizzo del chatbot, rendendo l’esperienza ancora più moderna e affascinante.

L’idea è quella di fornire OrientaBot con un set di voci reali, in modo da far familiarizzare l’utente con il chatbot.

OrientaBot nasce per supportare gli studenti universitari, immaginiamo OrientaBot in un sistema *embedded* nei corridoi dell’Università: sarebbe accessibile a tutti in un qualsiasi momento, evitando di utilizzare dispositivi personali e scaricare l’applicazione.

Questo sviluppo renderebbe il chatbot unico, qualcosa che puoi trovare solamente se sei in Università, rafforzando il branding dell’Ateneo e migliorando l’esperienza universitaria di tutti.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 42 - OrientaBot UNI