

**UNIVERSITA' DEGLI STUDI DI NAPOLI
“PARTHENOPE”**



**SCUOLA INTERDIPARTIMENTALE
DELLE SCIENZE, DELL'INGEGNERIA
E DELLA SALUTE INFORMATICA**

**Dipartimento di
Scienze e Tecnologie**

**Corso di Laurea in
Informatica**

Titolo tesi

**Sviluppo di un Chatbot per l'orientamento degli
studenti universitari**

TUTOR
Chiar.mo Prof.
Antonino Staiano

CANDIDATO
Dominick Ferraro
MATR. 0124002048

ANNO ACCADEMICO 2021/2022

INDICE

<u>ABSTRACT</u>	<u>5</u>
<u>CAPITOLO 1 - INTRODUZIONE.....</u>	<u>7</u>
1.1 – DIFFUSIONE DEI CHATBOT	11
1.2 – CLASSIFICAZIONE DI UN CHATBOT.....	12
1.3 – FUNZIONAMENTO DI UN CHATBOT	13
1.3.1 – FONDAMENTI DI NATURAL LANGUAGE PROCESSING PER LO SVILUPPO DI UN AGENTE CONVERSAZIONALE	15
<u>CAPITOLO 2 - DEFINIZIONE DEL DOMINIO APPLICATIVO E REQUISITI DI ORIENTABOT</u>	<u>25</u>
2.1 – SCENARI.....	28
2.2 – REQUISITI FUNZIONALI E NON FUNZIONALI.....	29
2.3 – OBIETTIVI DI PROGETTAZIONE.....	31
<u>CAPITOLO 3 - STRUMENTI PER LO SVILUPPO</u>	<u>32</u>
3.1 – LIBRERIE E FRAMEWORK	33
3.1.1 – NATURAL LANGUAGE TOOLKIT	34
3.1.2 – NUMPY	35
3.1.3 – PYTORCH.....	36
3.1.4 – TKINTER	37
<u>CAPITOLO 4 - SVILUPPO DELL'APPLICAZIONE</u>	<u>39</u>
4.1 – ALGORITMO DI ORIENTABOT.....	42
4.2 – SVILUPPO DI UNA RETE NEURALE.....	44
<u>CAPITOLO 5 - INTERFACCIA E USABILITÀ.....</u>	<u>57</u>

5.1 – TEST DI USABILITÀ	60
5.2 – VALUTAZIONE DI USABILITÀ.....	63
5.3 – MANUALE UTENTE PER L'ESECUZIONE	65
5.4 – ESECUZIONE DI ORIENTABOT	67
 <u>CAPITOLO 6 – CONCLUSIONI E SVILUPPI FUTURI.....</u>	 73
 <u>SITOGRAFIA.....</u>	 79
 <u>BIBLIOGRAFIA</u>	 80
 <u>RINGRAZIAMENTI</u>	 81

ELENCO DELLE FIGURE

FIGURA 1 - QUESTIONARIO AGLI STUDENTI	6
FIGURA 2 - INTERFACCIA DI ELIZA.....	8
FIGURA 3 - TEST DI TURING	9
FIGURA 4 - SPOT SIRI 2011	10
FIGURA 5 - CHATBOT E MERCATO.....	11
FIGURA 6 - ESEMPIO CHATBOT POSTE ITALIANE [6]	12
FIGURA 7 - ASSOCIAZIONE PATTERN → RESPONSE	14
FIGURA 8 - CAMPI NLP	16
FIGURA 9 - NLP APPLICAZIONI E TECNOLOGIE	17
FIGURA 10 - PAROLE & MORFOLOGIA.....	17
FIGURA 11 - LISTA DELLE PARTI DEL DISCORSO	18
FIGURA 12 - PARTI DEL DISCORSO	18
FIGURA 13 - SINTASSI	19
FIGURA 14 - EMOTICON, SLANG, ABBREVIAZIONI	19
FIGURA 15 - I SEI SIGNIFICATI DI 	21
FIGURA 16 - STEMMING	22
FIGURA 17 - LOWERING	23
FIGURA 18 - LOWERING + IGNORE PUNCTUATION	23
FIGURA 19 – LEMMATIZATION.....	24
FIGURA 20 - USE CASE DIAGRAM ORIENTABOT	25
FIGURA 21 – SEQUENCE DIAGRAM CHATBOT	30
FIGURA 22 - TIOBE TOP 10 PROGRAMMING LANGUAGE.....	32
FIGURA 23 - ORIENTABOT PACKAGE.....	33
FIGURA 24 - FUNZIONI DI NUMPY	35
FIGURA 25 – PYTORCH	36
FIGURA 26 - TKINTER INSTALL	38
FIGURA 27 - CREAZIONE DELLA BAG OF WORDS	41
FIGURA 28 - PASSI DELL'ALGORITMO NLP	42
FIGURA 29 - SCALA DI GRIGIO	44
FIGURA 30 - PESI E BIAS CAMBIANO OUTPUT DI UN NEURONE.....	46
FIGURA 31 - STEP FUNCTION.....	47
FIGURA 32 – SOFTMAX.....	48
FIGURA 33 – RELU.....	48
FIGURA 34 - FEEDFORWARD NEURAL NET CON 1 HIDDEN LAYER ..	49
FIGURA 35 - SEPARABILITÀ LINEARE	51

FIGURA 36 - PROBLEMA NON SEPARABILE LINEARMENTE.....	52
FIGURA 37 - FUNZIONE DI ATTIVAZIONE DEI NEURONI.....	55
FIGURA 38 - FNN 2 HIDDEN LAYERS	55
FIGURA 39 - PALETTE INTERFACCIA ORIENTABOT	57
FIGURA 40 - ORIENTABOT GUI	58
FIGURA 41 - FOCUS SULLA BARRA DI TESTO E PULSANTE DI INVIO	58
FIGURA 42 - AVATAR ORIENTABOT	59
FIGURA 43 - AFFORDANCE.....	62
FIGURA 44 - VALUTAZIONE CHATBOT UNIVERSITÀ DI PADOVA....	63
FIGURA 45 - VALUTAZIONE ORIENTABOT.....	64
FIGURA 46 - ESECUZIONE DI TRAIN.PY	65
FIGURA 47 - ESTRATTO DEI TAG - PATTERNS.....	65
FIGURA 48 - RAPPORTO EPOCH – LOSS	66
FIGURA 49 - ESECUZIONE DI ORIENTABOT.....	66
FIGURA 50 - CONVERSAZIONE CON ORIENTABOT N.1	67
FIGURA 51 - INTERAZIONE CON ORIENTABOT N.2.....	68
FIGURA 52 - INTERAZIONE CON ORIENTABOT N.3.....	69
FIGURA 53 - INTERAZIONE CON ORIENTABOT N.4.....	69
FIGURA 54 - INTERAZIONE CON ORIENTABOT N.5.....	70
FIGURA 55 - INTERAZIONE CON ORIENTABOT N.6.....	70
FIGURA 56 - INTERAZIONE CON ORIENTABOT N.7.....	70
FIGURA 57 - INTERAZIONE CON ORIENTABOT N.8.....	71
FIGURA 58 - INTERAZIONE CON ORIENTABOT N.9.....	71
FIGURA 59 - INTERAZIONE CON ORIENTABOT N.10.....	71
FIGURA 60 - ORIENTABOT SPLASH SCREEN	72
FIGURA 61 - ORIENTABOT MOBILE	73
FIGURA 62 - ORIENTABOT SPEECH TO TEXT	75
FIGURA 63 - ORIENTABOT UNI	76
FIGURA 64 - REPORT ORIENTABOT	77
FIGURA 65 - ORIENTABOT X UNIVERSITÀ PARTHENOPE	78

ABSTRACT

Spesso la ricerca di informazioni può 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'università.
- **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.

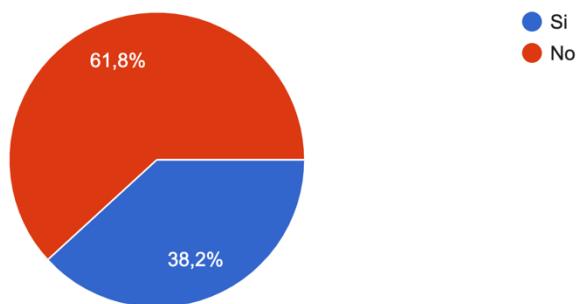
Nasce così OrientaBot, un chatbot che voglio donare alla mia università, per aiutare i miei attuali e futuri colleghi.

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.

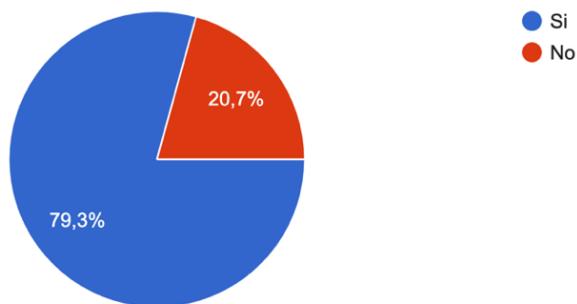
Prima di iscriverti sapevi dove reperire info riguardo l'università e il corso di studi?

110 risposte



Hai mai avuto difficoltà a chiedere informazioni poiché non conoscevi altri colleghi o professori?

110 risposte



Useresti un ChatBot che risponde a tutte le tue domande riguardo la tua università? [Tasse, tirocinio, servizi, aule studio, WiFi, esse3...]

110 risposte

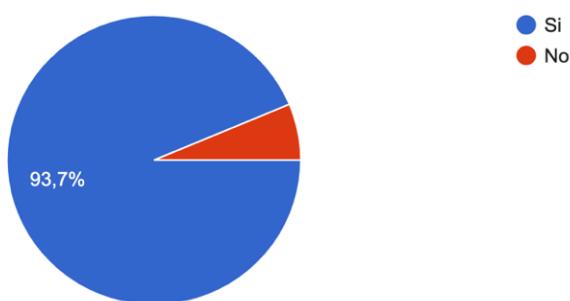


Figura 1 - Questionario agli studenti

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.

Al giorno d'oggi interagire con un chatbot è normalità, siamo abituati a comunicare con queste tecnologie per ogni genere di esigenza: lavorativa, accademica, per servizi o semplicemente per passare un po' di tempo e divertirsi.

Con la larga diffusione di questi agenti conversazionali la vita delle persone è migliorata: i task più noiosi e automatici che ogni giorno facciamo vengono sempre più affidati ai chatbot, velocizzando e automatizzando processi quotidiani: scrivere note, appunti, fare una ricerca, leggere una notizia, chiedere il meteo...

Non mancano questioni legali sui chatbot: molti sono stati sospesi a causa di violazione di privacy, è il caso di *Replika*, chatbot che simula un amico virtuale, che è stato messo in stop dal Garante della Privacy poiché metteva a rischio i minori a causa di risposte da parte del chatbot assolutamente non idonee e pericolose [1].

L'*amico virtuale*, presentato come in grado di migliorare il benessere emotivo dell'utente, aiuterebbe l'utente a comprendere i propri pensieri e calmare l'ansia, attraverso la gestione dello stress, la socializzazione e la ricerca dell'amore.

Ha caratteristiche che, intervenendo sull'umore della persona, possono accrescere i rischi per i soggetti ancora in una fase di sviluppo o in stato di fragilità emotiva.

Nonostante questi problemi, le autorità giuridiche, insieme a sviluppatori e investitori, lavorano ogni giorno per rendere l'interazione coi chatbot sicura e legale.

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 [2].

```

Welcome to
      EEEEEE LL      IIII    ZZZZZZ  AAAA
      EE     LL      II     ZZ   AA   AA
      EEEEEE LL      II     ZZZ  AAAAAAA
      EE     LL      II     ZZ   AA   AA
      EEEEEE LLLLLL IIII ZZZZZZ  AA   AA

Eliza is a mock Rogerian psychotherapist .
The original program was described by Joseph Weizenbaum in 1966 .
This implementation by Norbert Landsteiner 2005 .

ELIZA: Is something troubling you ?
YOU: Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU: They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU: Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU: He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU: It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:

```

Figura 2 - Interfaccia di ELIZA

ELIZA non è capace di pensare autonomamente, funziona solo attraverso un dataset prestabilito, usato per dare risposte consone all'utente utilizzatore.

In ogni caso, ELIZA è stato implementato per il supporto di persone con problemi psicologici, con l'obiettivo di migliorare la vita degli utenti.

Nonostante sembri essere un argomento dell'ultimo momento, già negli anni '50 **Alan Turing** in un articolo “*Computing Machinery and Intelligence*” stabiliva dei criteri e delle linee guida per le macchine.

Turing fu un importante matematico e informatico, considerato il padre dell'Intelligenza Artificiale.

Turing si domandò se una macchina fosse stata in grado di partecipare al *gioco delle imitazioni*: il gioco originale prevede tre giocatori, una donna, un uomo e un interrogatore, a prescindere dal sesso.

L'interrogatore non conosce il sesso degli altri due partecipanti e deve indovinare chi è l'uomo e chi la donna ponendo dei quesiti a questi ultimi.

La versione di Turing consisteva nel sostituire con una macchina uno dei due partecipanti, in modo da testare se l'interrogatore fosse stato in grado di riconoscere un essere umano da un robot, nacque quindi il *Test di Turing*.

Nel Test di Turing, la macchina non è altro che un chatbot che cerca di ingannare l'interrogatore fingendo di essere umano.

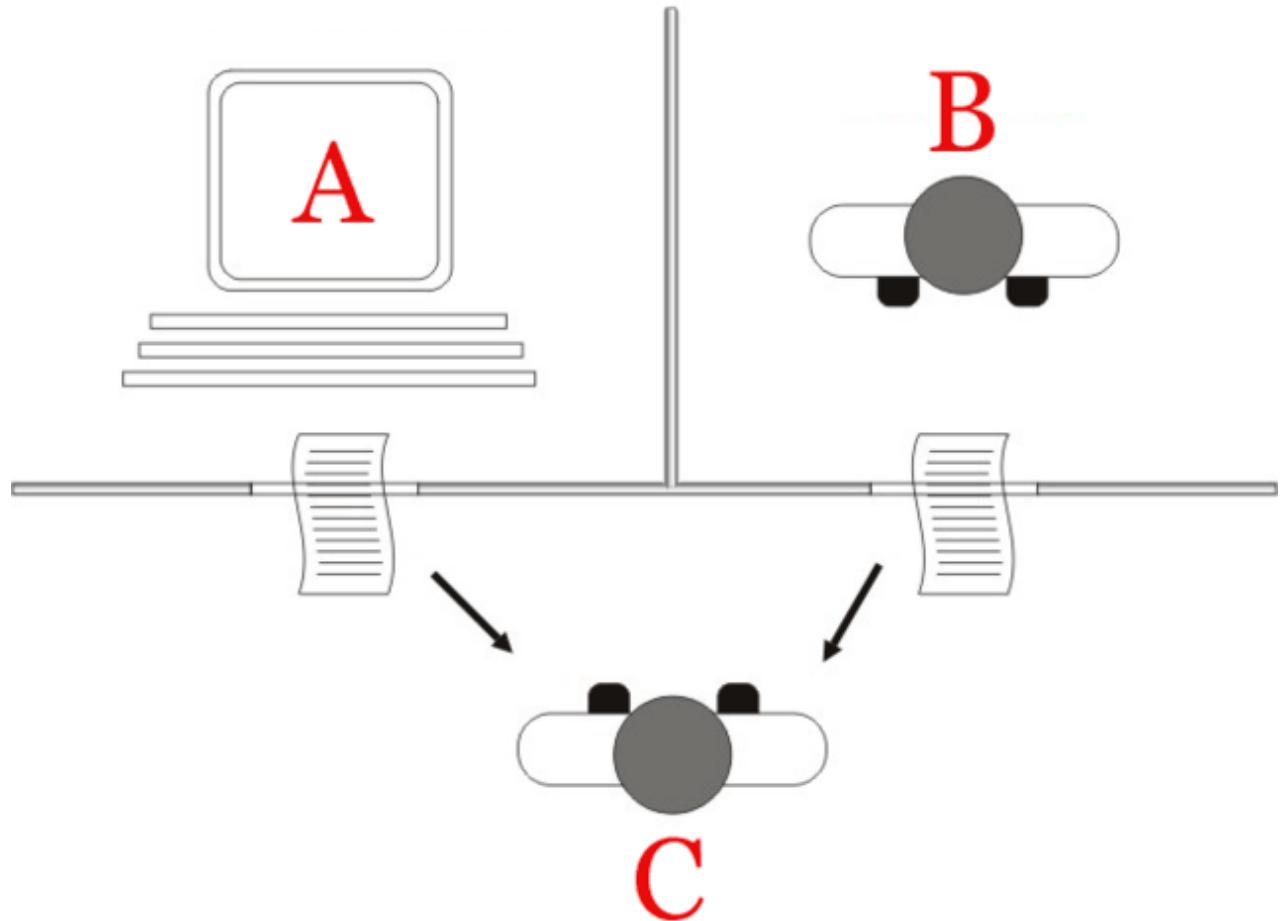


Figura 3 - Test di Turing

Il Test di Turing ha ricevuto molte critiche poiché è un test che imita l'intelligenza umana ma non dimostra la capacità della macchina.

La verità è che non esistono criteri assolutamente affidabili per distinguere l'intelligenza dell'uomo dall'intelligenza artificiale, inoltre non è detto che esista solamente l'intelligenza umana.

Qualsiasi criterio di valutazione, per quanto complesso o articolato, utilizzato per discriminare l'intelligenza umana da quella di una macchina, una volta definito e reso pubblico, potrebbe facilmente essere aggirato da un buon sviluppatore, fornendo l'agente conversazionale di adeguate tipologie di risposte in grado di trarre in inganno gli esaminatori e risultare sempre più simile all'essere umano.

Secondo il vicepresidente e capo scienziato di Alexa presso Amazon, Rohit Prasad, il Test di Turing non è più attuale per diversi motivi: un'intelligenza artificiale conosce le risposte istantaneamente ma per simulare l'intelligenza umana dovrebbe prendere alcuni secondi per riflettere ed elaborare, inoltre il Turing Test non prende in considerazioni le crescenti abilità delle IA nell'uso dei sensori per ascoltare, vedere e sentire il mondo circostante.

Secondo l'inventore e informatico **Raymond Kurzweil**, per intrattenere conversazioni con chatbot che risultino come chiacchiere con un essere umano in carne ed ossa bisognerà attendere il 2029 [3].

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, ricercare informazioni, scrivere delle note, impostare riunioni e tanto altro.



Figura 4 - Spot Siri 2011

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

Dopo Apple, tutte le grandi aziende Tech hanno sviluppato un proprio agente conversazionale a supporto degli utenti, ciò ha portato un forte interesse pubblico verso queste nuove tecnologie facendo crescere esponenzialmente la ricerca e lo sviluppo di nuove tecnologie per la creazione di agenti conversazionali sempre più efficienti e simili all'essere umano.

Tra gli assistenti personali delle aziende Tech ricordiamo:

- 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 necessita.

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 [4].

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% [5].



Figura 5 - Chatbot e mercato

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:

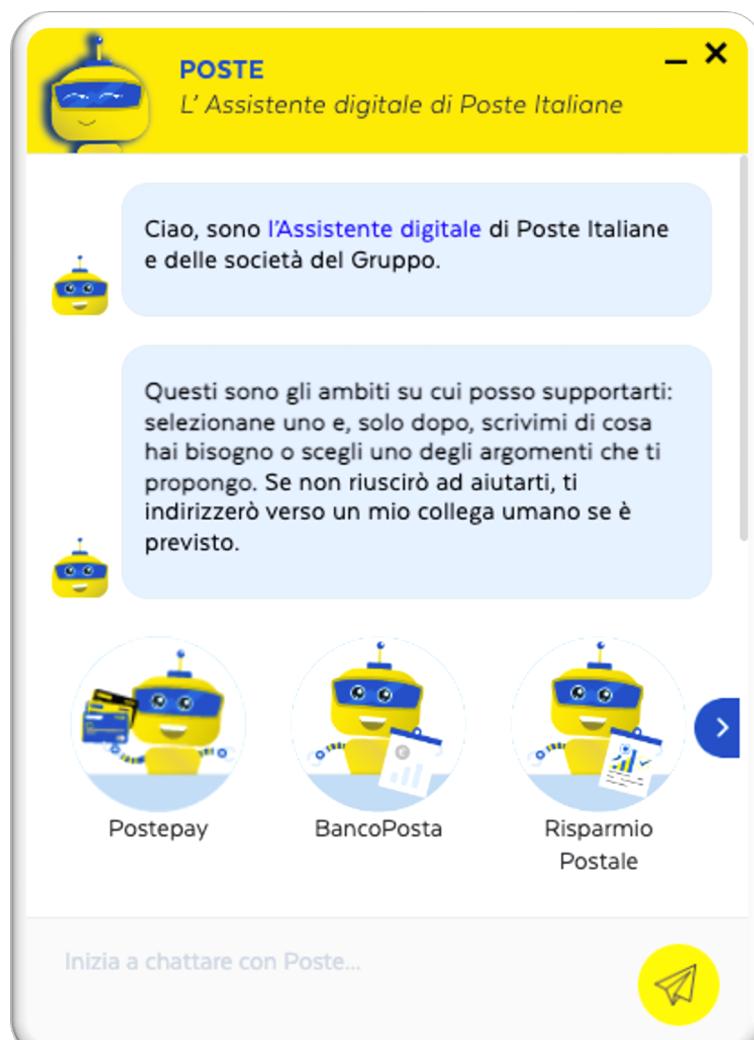


Figura 6 - Esempio Chatbot Poste Italiane [6]

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.

2. **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.

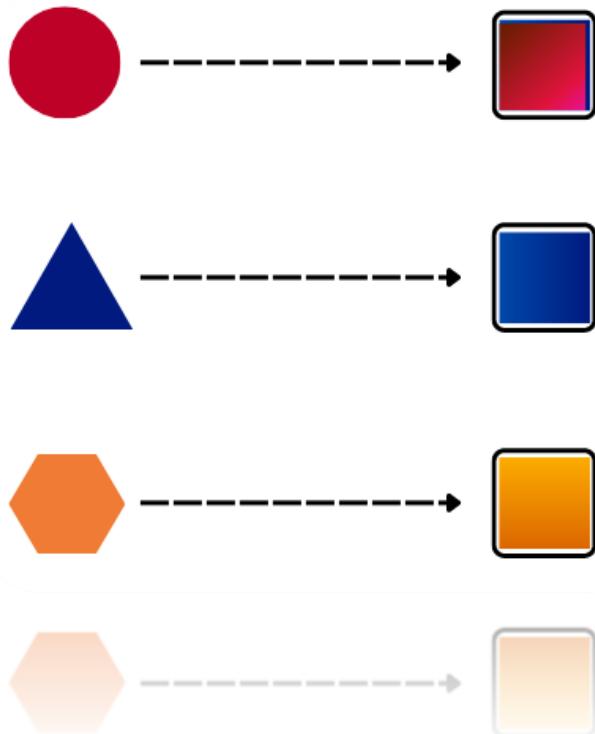


Figura 7 - Associazione Pattern → Response

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 [17] [18].

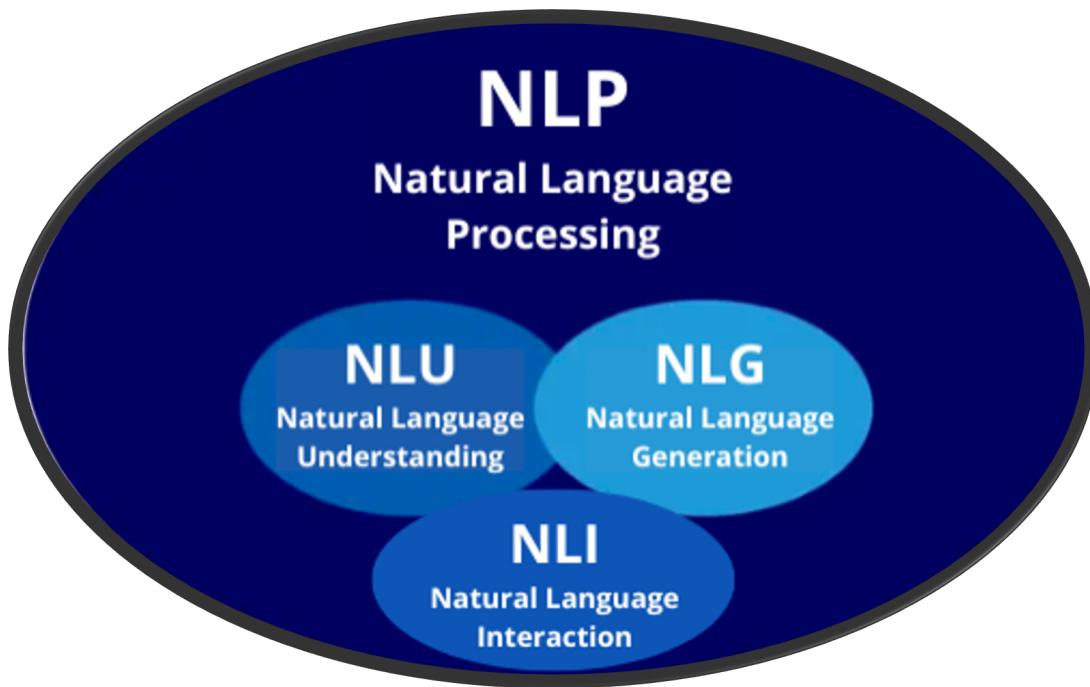


Figura 8 - Campi NLP

L'obiettivo del Natural Language Processing è creare algoritmi che facciano apprendere il linguaggio naturale ai computer per eseguire determinati task interessanti:

Applications

- Machine Translation
 - Google translate, DeepL
- Information retrieval
- Question Answering
 - IBM Watson
- Dialogue Systems (Chatbot & virtual assistant)
 - Siri, Alexa, Google Home
- Information extraction
- Summarization
- Sentiment Analysis
- Document classification & fake news detection

Core technologies

- Language Modelling
- Part-of-speech tagging
- Syntactic parsing
- Named-entity recognition
- Coreference resolution
- Word sense disambiguation
- Semantic Role Labelling
- ...

Figura 9 - NLP Applicazioni e Tecnologie

Un linguaggio è composto da molti livelli di strutture e l'essere umano integra tutti questi processi per elaborare e utilizzare un linguaggio, idealmente dovrebbe fare lo stesso un computer.



Figura 10 - Parole & Morfologia

Per quanto riguarda le parti del discorso possiamo analizzare questa semplice frase utilizzando la tabella sottostante:



Figura 11 - Lista delle parti del discorso

Avremo quindi:



Figura 12 - Parti del discorso

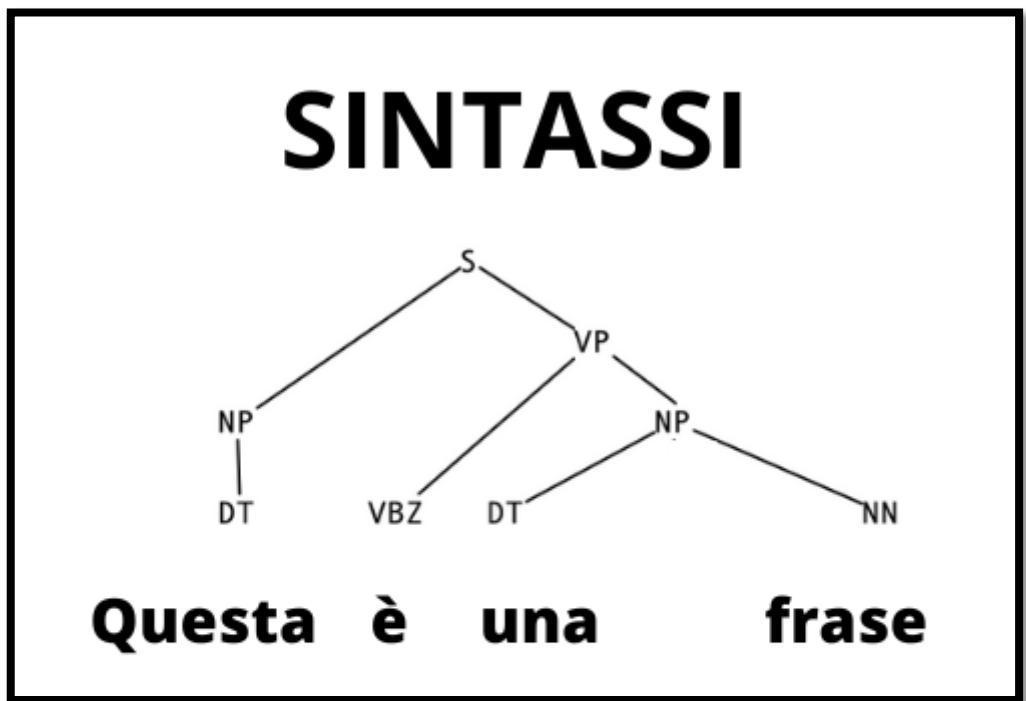


Figura 13 - Sintassi

Tra le tante difficoltà dell'elaborazione del linguaggio naturale abbiamo l'ambiguità della lingua, la vastità di linguaggi e alfabeti differenti, gli slang, gli hashtag, le emoticon, abbreviazioni, mancanza di espressività, dipendenza dal contesto...

Ecco perché il NLP si distingue da tutti gli altri studi appartenenti al Machine Learning e Intelligenza Artificiale.

Ecco un esempio di un mix di fattori che rendono il NLP una scienza complessa e interessante [7]:



Figura 14 - Emoticon, slang, abbreviazioni

Un algoritmo NLP per elaborare il linguaggio naturale comprende alcune tecniche particolari che fondono l'informatica, la grammatica e la logica: ***tokenization, stemming, lowering & ignore punctuation, lemmatization*** [19].

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 più 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 $X \leq N$, 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:

- a. I computer sono incredibilmente veloci, accurati e stupidi
- b. Gli uomini sono incredibilmente lenti, inaccurati e intelligenti.
- c. L’insieme dei due costituisce una forza incalcolabile.

Tokenization per parole:

- i. I
- ii. Computer
- iii. Sono
- iv. Incredibilmente
- v. Veloce
- vi. Accurati
- vii. E
- viii. 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 più token e significati differenti, come la parola “عقد”:



Figura 15 - 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 “*università*” 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.

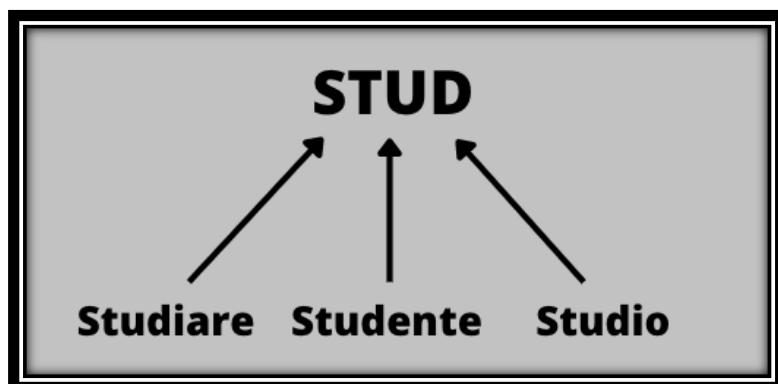


Figura 16 - Stemming

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.

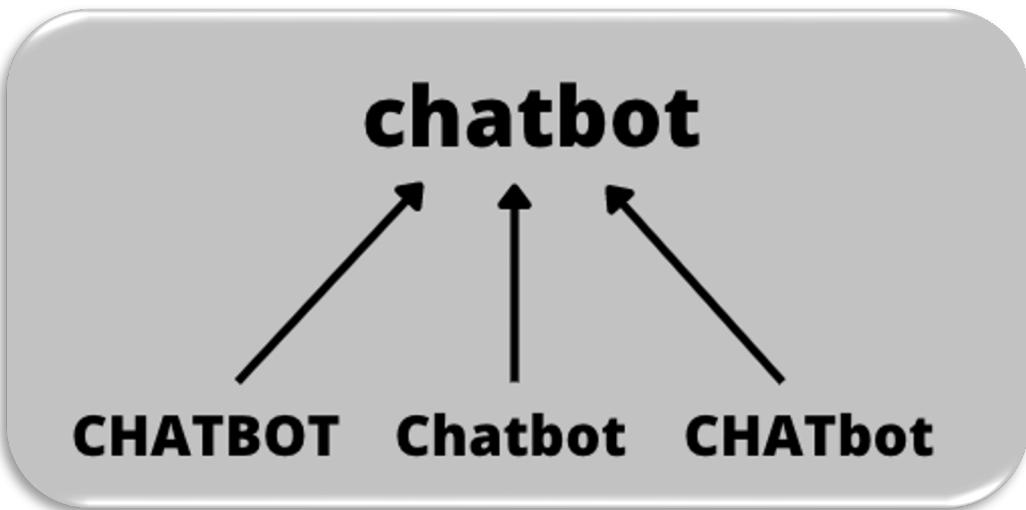


Figura 17 - Lowering

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 18 - 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 intuitibile usare la lemmatization potrebbe essere più corretto, ovviamente resta una scelta dello sviluppatore quale tecnica utilizzare.

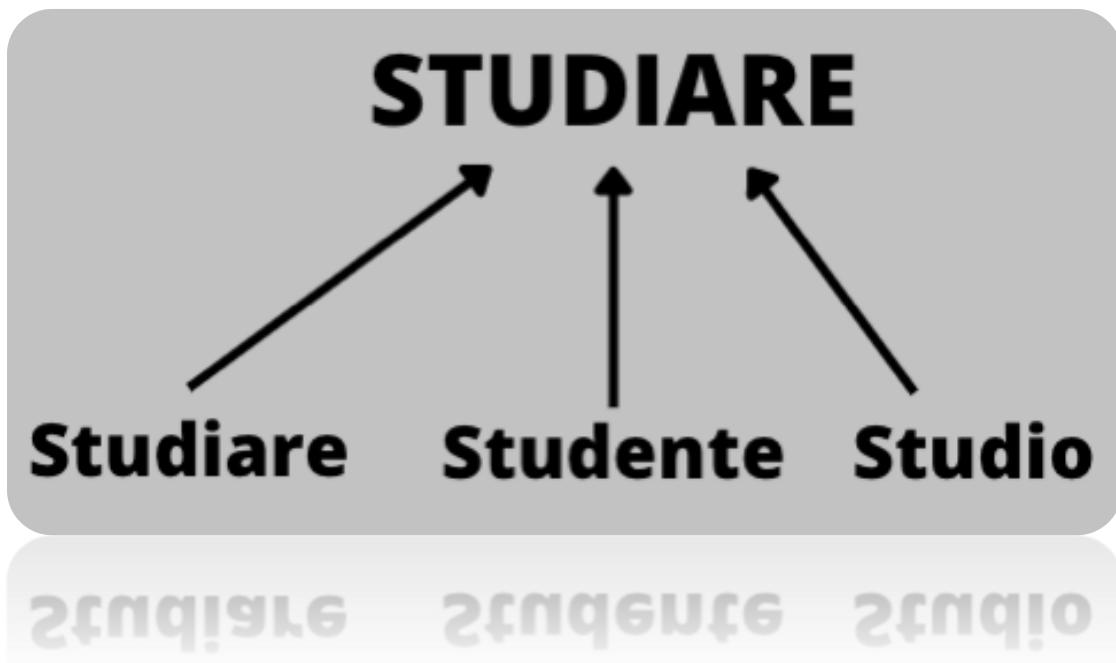


Figura 19 – Lemmatization

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 estratte da esperienze personali, siti dell'università, moduli online e altro e sono accuratamente e completamente affidabili.

Possiamo descrivere il sistema OrientaBot con un semplice *use case diagram*:

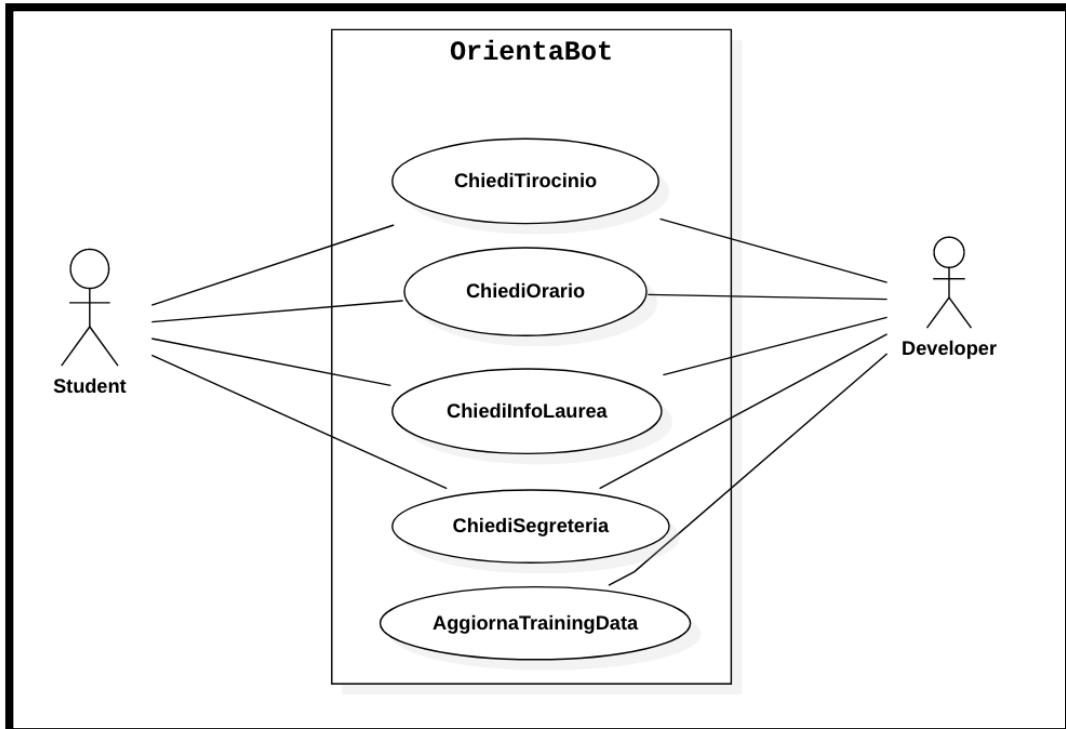


Figura 20 - Use Case Diagram OrientaBot

Ovviamente solo lo sviluppatore può aggiornare al training data completo e aggiornarlo quando lo ritiene opportuno.

Di seguito riportiamo due esempi di tabelle di casi d'uso, una per l'attore *studente* e l'altra per l'attore *developer*, per esplicitare meglio come funzionano alcune operazioni del Sistema.

Nome Scenario	<u>ChiediTirocinio</u>
Attori Partecipanti	Studente
Condizioni di ingresso	Lo studente ha accesso a OrientaBot
Flusso Eventi:	<ol style="list-style-type: none"> 1. Lo Studente chiede a OrientaBot informazioni sul tirocinio. 2. OrientaBot elabora la frase dello Studente e risponde al quesito. 3. Lo Studente, ottenuta la risposta, decide se continuare l'interazione o meno.
Condizioni di uscita	Lo Studente esce dal sistema.
Requisiti speciali	-

Tabella 1 - Caso d'uso ChiediTirocinio

Nome Scenario	<u>AggiornaTrainingData</u>
Attori Partecipanti	Sviluppatore
Condizioni di ingresso	Lo sviluppatore ha accesso a OrientaBot
Flusso Eventi:	<ol style="list-style-type: none"> 1. Lo sviluppatore, dopo aver raccolto nuove informazioni dalle sue fonti (siti web, segreteria, studenti...), aggiorna OrientaBot ampliando il Training Data. 2. OrientaBot elabora il nuovo Training Data e completa l'addestramento. 3. Lo sviluppatore, completata l'operazione, testa il sistema per verificarne il corretto addestramento.
Condizioni di uscita	Lo sviluppatore esce dal sistema.
Requisiti speciali	<i>Solo lo sviluppatore può accedere a questa funzionalità.</i>

Tabella 2 - Caso d'uso AggiornaTrainingData

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 modalità di viaggio.

- Mario, **studente fuoricorso**, vuole cambiare percorso di studi ma non riesce a prenotare un appuntamento con la segreteria.
Mario accede ad OrientaBot e riceve tutte le informazioni, la modulistica e le procedure da seguire per completare l’operazione richiesta.

- 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.

Questi appena elencati sono solo una serie di possibili scenari reali che potrebbero idealizzarsi, non si escludono situazioni limite che potrebbero portare il sistema sotto stress causando un malfunzionamento.

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.

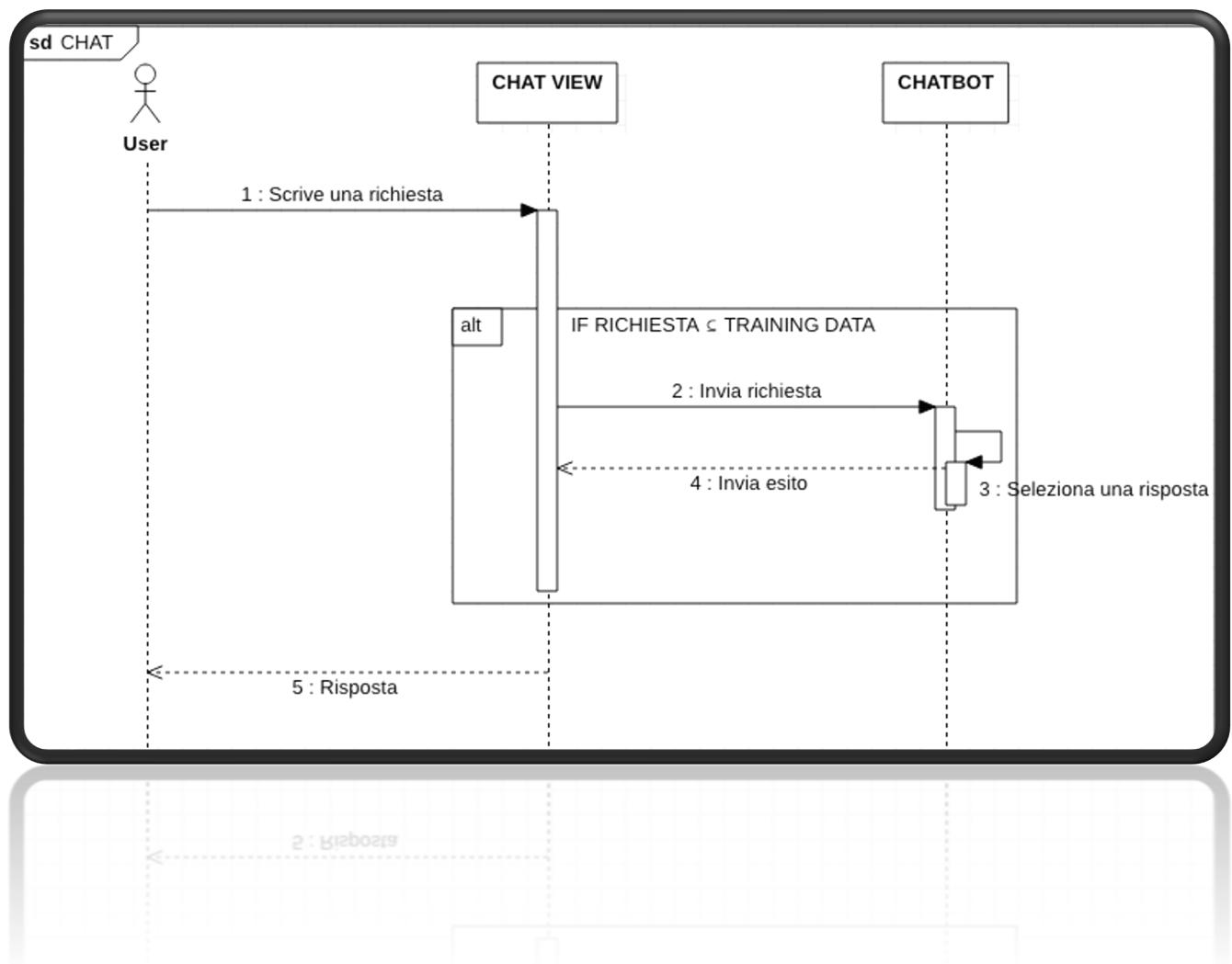


Figura 21 – Sequence Diagram Chatbot

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 [20].

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. [8]

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 [9].

Feb 2023	Feb 2022	Change	Programming Language	Ratings	Change
1	1		 Python	15.49%	+0.16%
2	2		 C	15.39%	+1.31%
3	4		 C++	13.94%	+5.93%
4	3		 Java	13.21%	+1.07%
5	5		 C#	6.38%	+1.01%
6	6		 Visual Basic	4.14%	-1.09%
7	7		 JavaScript	2.52%	+0.70%
8	10		 SQL	2.12%	+0.58%
9	9		 Assembly language	1.38%	-0.21%
10	8		 PHP	1.29%	-0.49%

Figura 22 - 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:

- a. **Natural Language ToolKit (NLTK)**
- b. **NumPy**
- c. **PyTorch**
- d. **Tkinter**

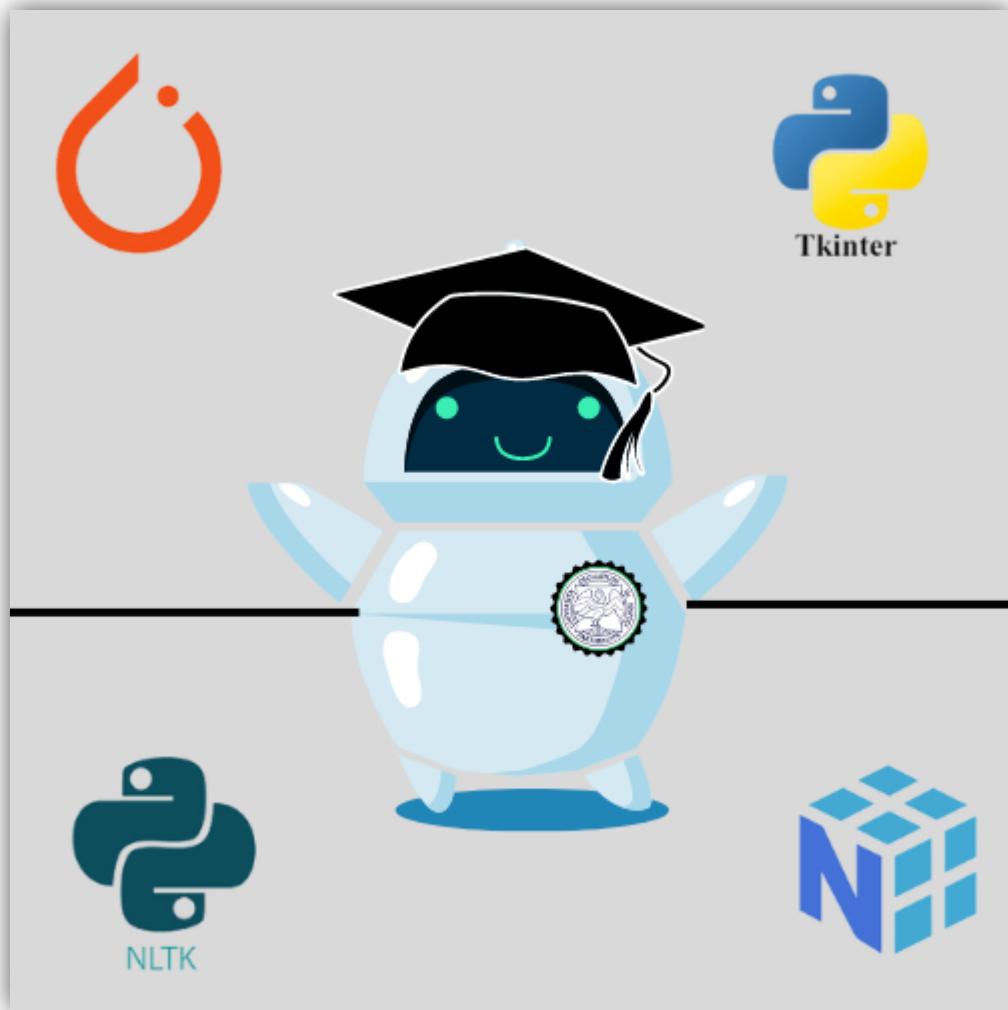


Figura 23 - OrientaBot Package

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* [10].

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 puntuale 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 [11].

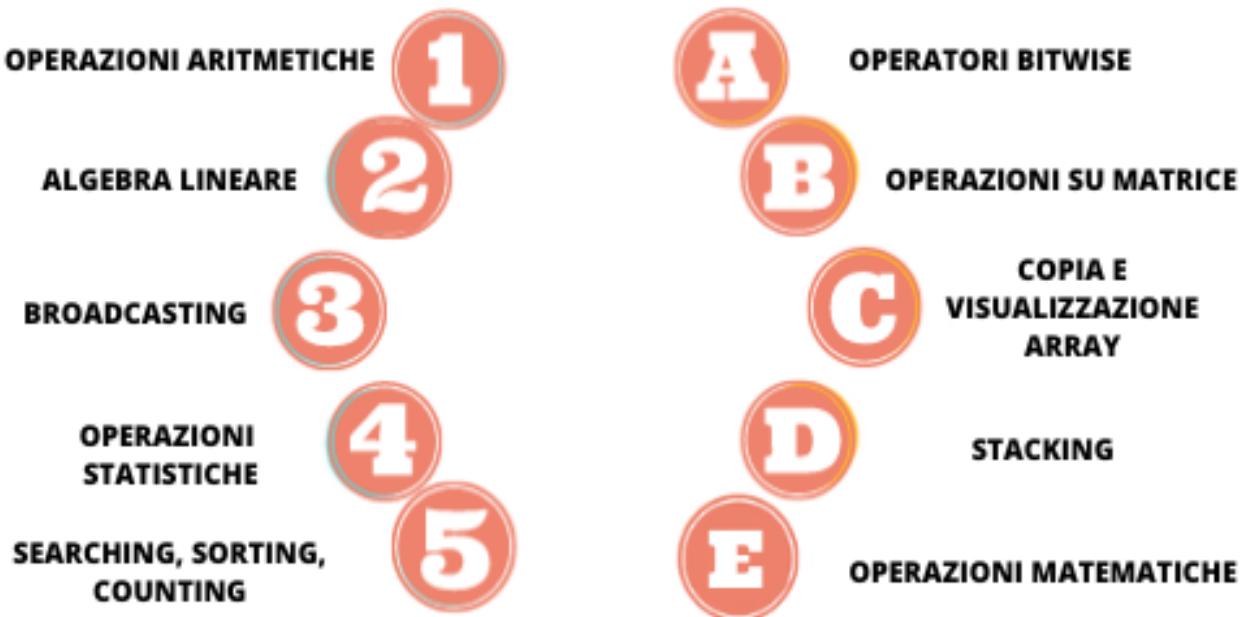


Figura 24 - Funzioni di NumPy

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 [12].

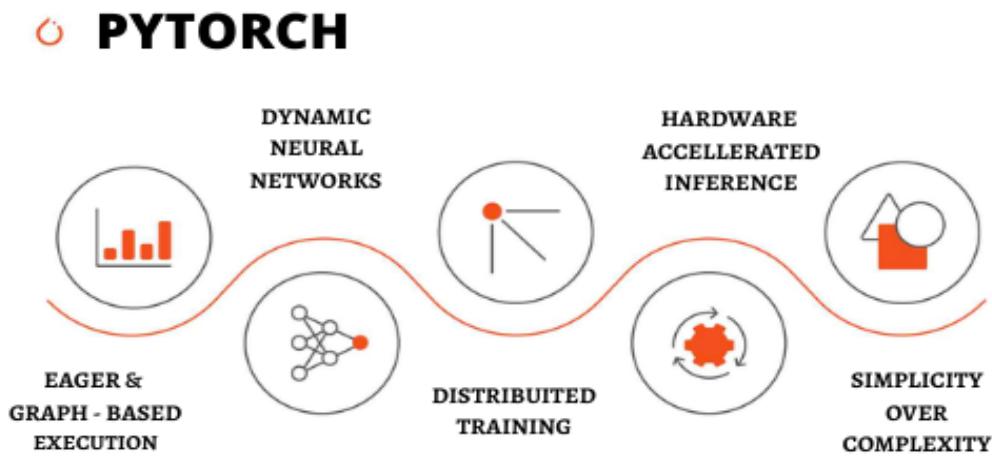


Figura 25 – PyTorch

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, menù 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 [13].

```
MacBook-Pro-di-Dominick:~ dominickferraro$ pip install tk
Collecting tk
  Downloading tk-0.1.0-py3-none-any.whl (3.9 kB)
Installing collected packages: tk
Successfully installed tk-0.1.0
```

Figura 26 - Tkinter Install

CAPITOLO 4

SVILUPPO DELL'APPLICAZIONE

OrienteBot 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 OrienteBot 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 chatbot 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:

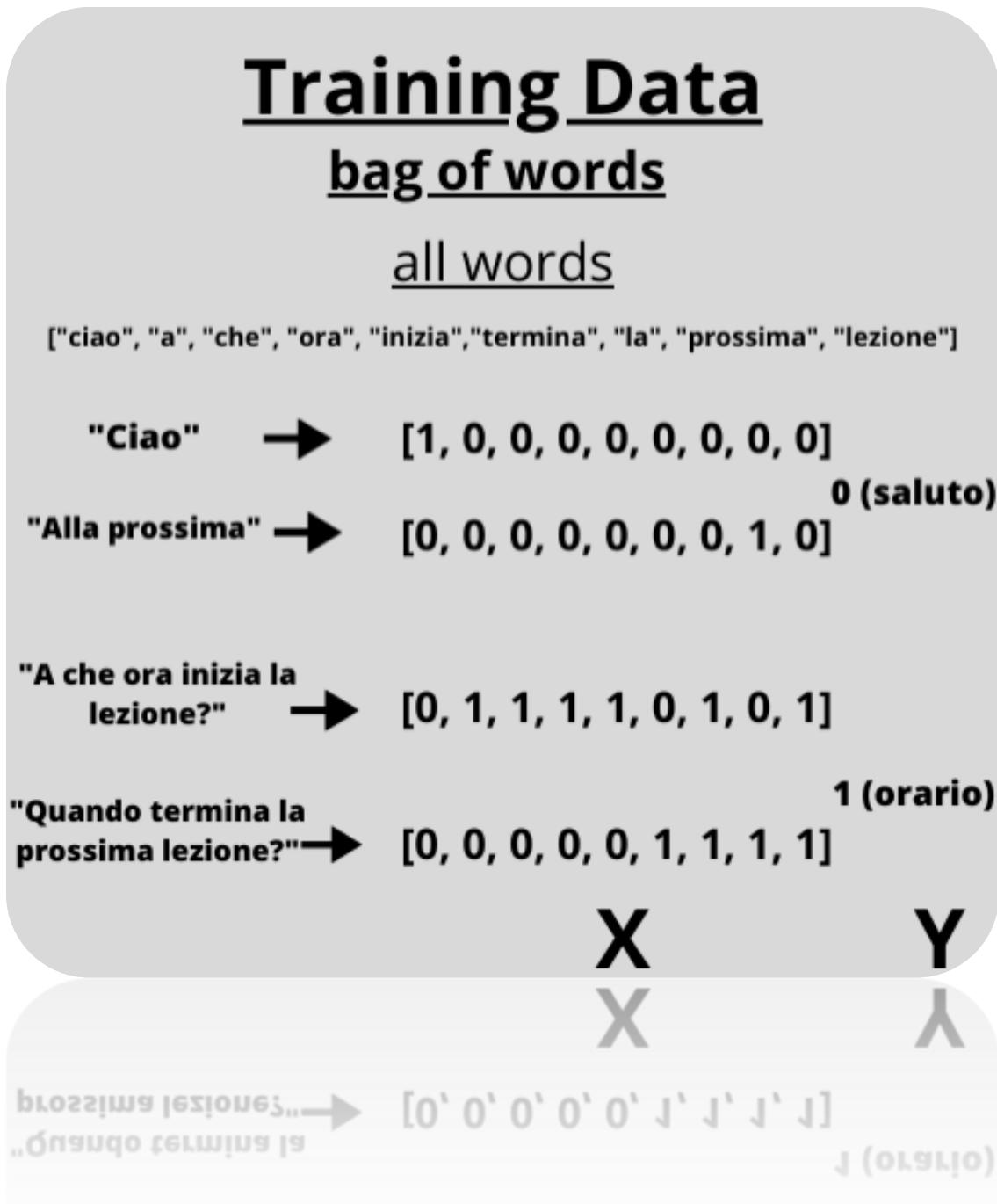


Figura 27 - Creazione della bag of words

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:

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



Figura 28 - 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_train* mettiamo le bag of words e in *Y_train* 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 *nlk 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/dom000D/Bot> [21].

4.2 – SVILUPPO DI UNA RETE NEURALE

L'idea di rete neurale prende ispirazione dal cervello, ma cosa sono i neuroni e come sono collegati tra loro?

Il **neurone** è un qualcosa che contiene un numero, più precisamente un numero compreso tra 0 e 1.

Ad esempio, se ho un'immagine 28×28 avremo 784 neuroni, dove ognuno di esso rappresenta la scala di grigio di ogni pixel corrispondente con valori che vanno da 0 (pixel neri) e 1 (pixel bianchi):

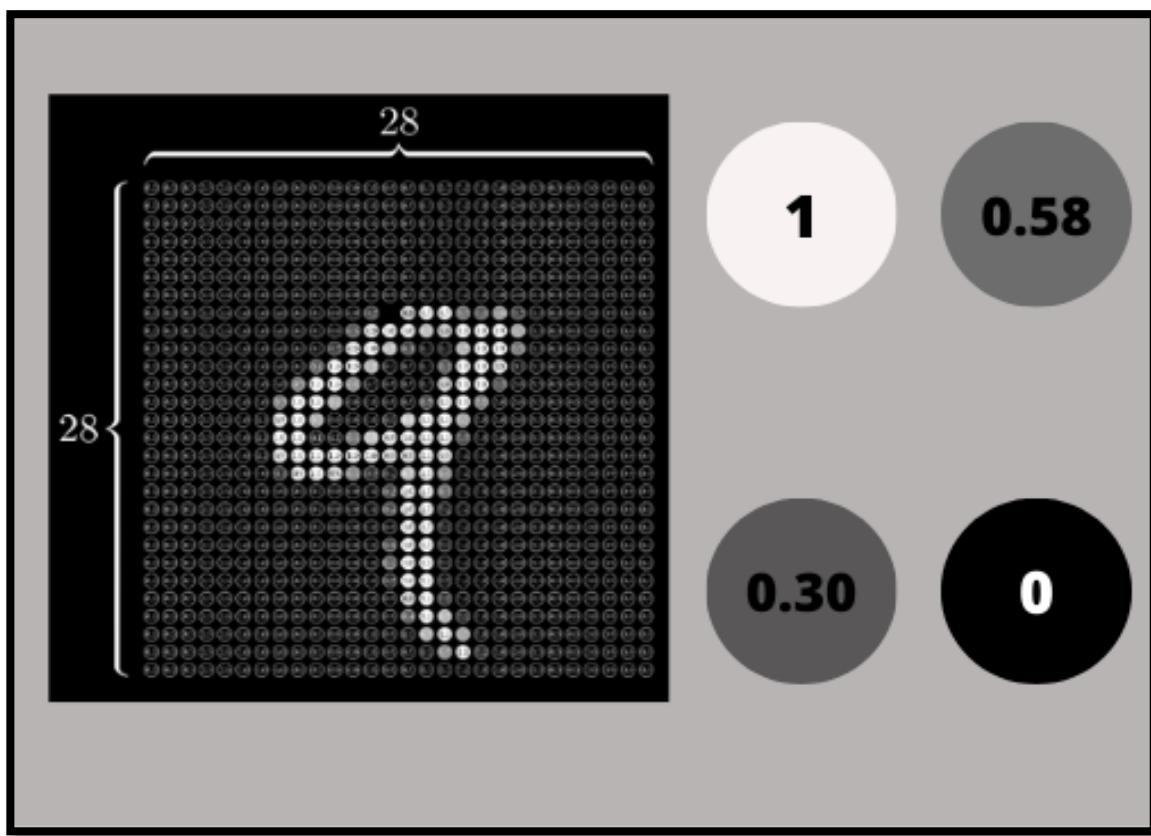


Figura 29 - Scala di Grigio

Una rete neurale è composta da molti neuroni, divisi in strati. Ogni neurone è collegato a tutti i neuroni dello strato successivo tramite una **connessione pesata**. Una connessione non è altro che un valore numerico, il peso, che viene moltiplicato per il valore del neurone collegato.

Ciascun neurone somma i valori pesati di tutti i neuroni ad esse collegato e aggiunge un valore di **bias**, un valore costante che va sommato al calcolo che coinvolge il peso menzionato.

Anche il bias è un fattore che viene modificato durante l'addestramento.

Una rete neurale è in grado di generalizzare e modellare un problema nel mondo reale (che non è altro che una funzione matematica) proprio grazie al costante aggiustamento di pesi e bias, che modulano l'output e l'input di ogni singolo neurone finché non la rete non approccia ad una soluzione accettabile.

L'output di un neurone è espresso dalla formula:

$$\text{output} = \text{input} \times \text{weights} + \text{bias}$$

L'aggiustamento di pesi e bias viene fatto nelle **hidden layers** (strati nascosti), che sono gli strati presenti tra lo strato di input e quello di output.

Sono detti "nascosti" proprio perché non vediamo il comportamento di aggiustamento di pesi e bias.

Ecco perché le reti neurali sono delle black box.

La caratteristica che rende complesse le reti neurali è proprio la enorme mole di calcoli che avviene a livello sia di rete che di singolo neurone. Insieme ai pesi e bias ci sono le funzioni di attivazione che aggiungono una ulteriore complessità matematica ma influenzano enormemente la performance di una rete neurale.

Pesi e bias possono essere interpretati come un sistema di manopole che possiamo ruotare per ottimizzare il nostro modello - come quando cerchiamo di spostare le manopole del gas per avere una mole di fuoco gradita.

La differenza sostanziale è che in una rete neurale, abbiamo centinaia se non migliaia di manopole da girare per raggiungere il risultato finale.

Poiché pesi e bias sono dei parametri della rete, questi saranno oggetto del cambiamento generato dalla rotazione della manopola immaginaria.

Visto che i pesi sono moltiplicati all'input, questi influenzano la magnitudine dell'input.

Il bias, invece, poiché è sommato all'espressione $inputs \times weights$, sposterà la funzione nel piano dimensionale.

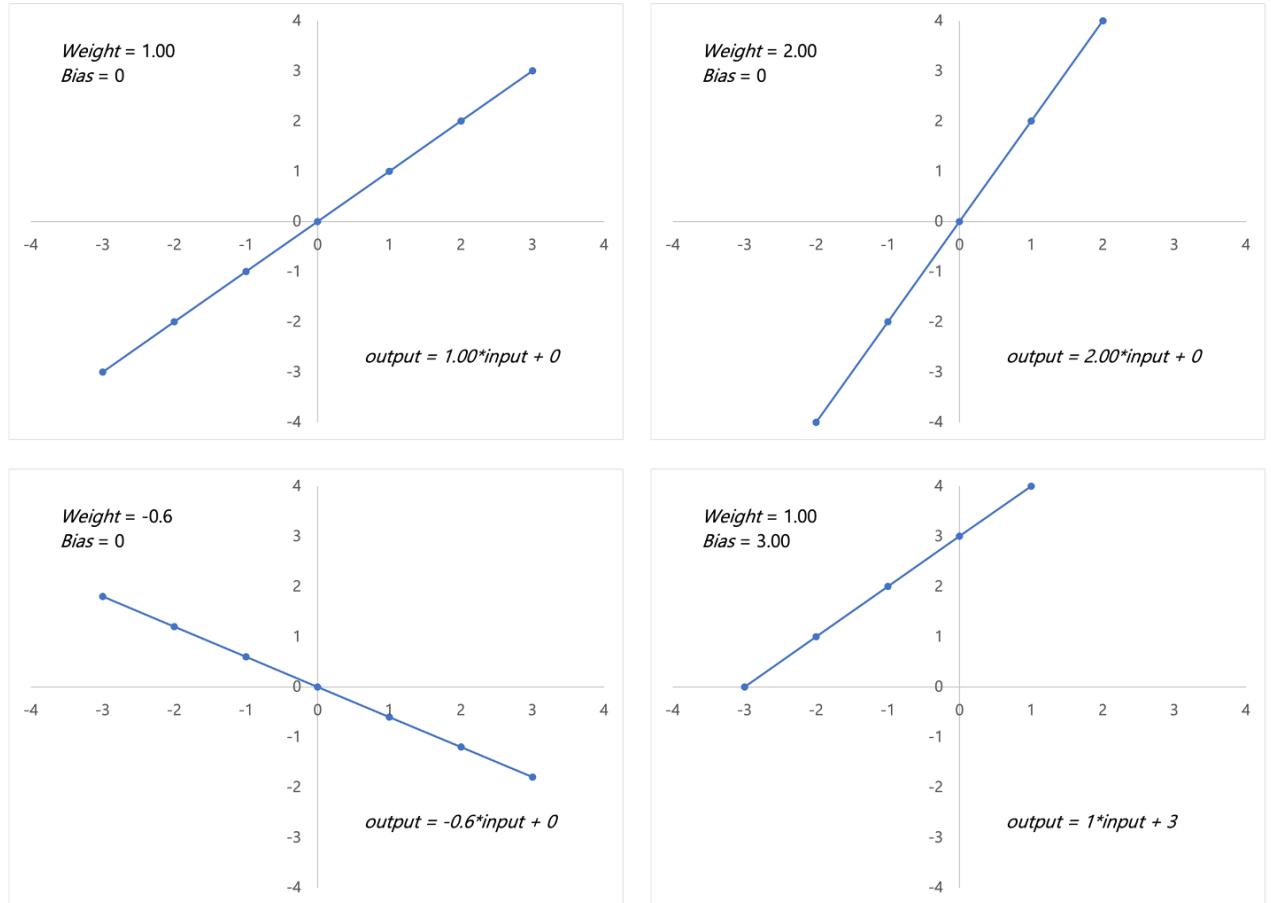


Figura 30 - Pesi e bias cambiano output di un neurone

Com'è possibile notare, pesi e bias impattano il comportamento di ogni neurone artificiale, ma lo fanno in maniera rispettivamente diversa. I pesi sono solitamente inizializzati randomicamente mentre il bias a 0.

Il comportamento di un neurone è anche influenzato dalla sua **funzione di attivazione** che, parallela al potenziale d'azione per un neurone naturale, definisce le condizioni di attivazione e relativi valori dell'output finale.

In gergo informatico/matematico, chiamiamo questa funzione una **step function** (funzione gradino).

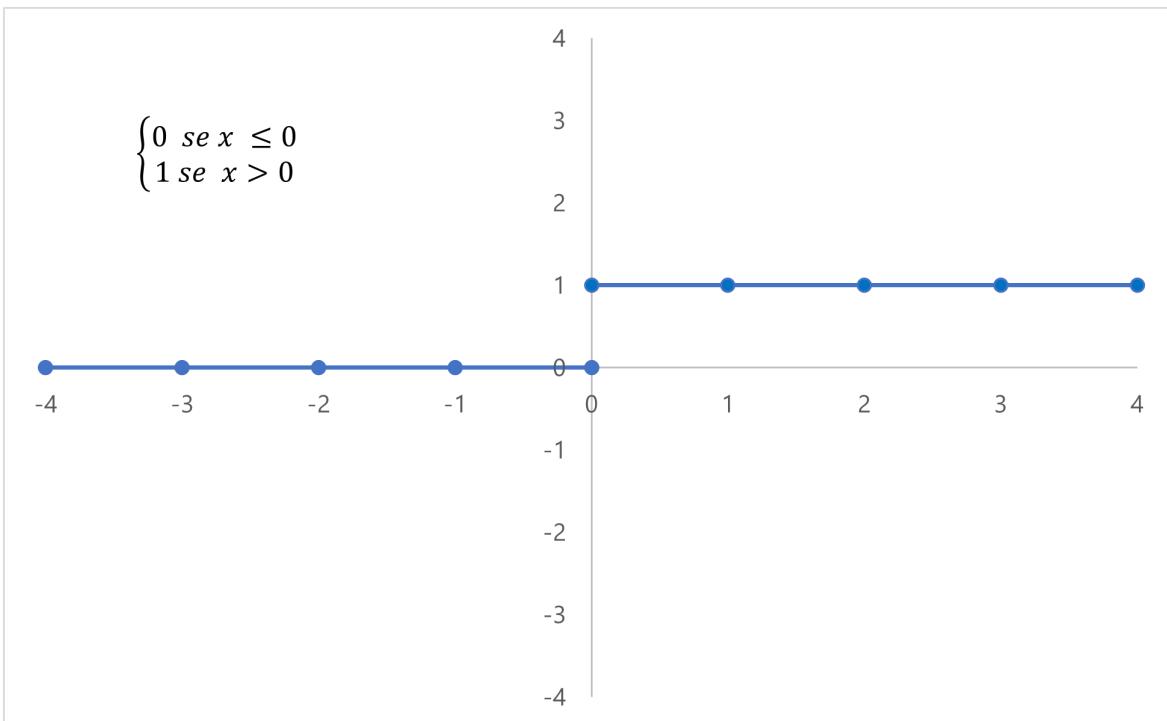


Figura 31 - Step function

Seguendo la logica

$$1 \ x > 0; 0 \ x \leq 0$$

la funzione gradino permette al neurone di restituire 1 se l'input è maggiore di 0 oppure 0 se l'input è minore o uguale a 0.

Questo comportamento simula il comportamento di un neurone naturale e segue la formula

$$\text{output} = \text{sum}(\text{inputs} \times \text{weights}) + \text{bias}$$

La step function è però molto semplice, e nel settore si tende ad usare delle funzioni di attivazione più complesse, come l'unità lineare rettificata (ReLU) e SoftMax.

La funzione **SoftMax** è utilizzata nello strato finale della classificazione multi classe.

Essa prende un vettore di K numero reale e lo normalizza in una distribuzione di probabilità costituita da probabilità K corrispondenti agli esponenziali del numero di input.

Prima di applicare questa funzione, alcuni componenti del vettore potrebbero essere negativi o maggiori di uno ma dopo aver applicato la SoftMax ogni componente sarà compresa tra 0 e 1 e si sommerà a 1.

I pro di questa funzione è che può essere usata per la classificazione multi classe e usata nel livello di output delle reti neurali, i contro è che la SoftMax è computazionalmente molto costosa in quanto dobbiamo calcolare molti termini di esponente.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

Figura 32 – SoftMax

Per quanto riguarda la **ReLU**, unità lineare rettificata, risulta essere molto più semplice da calcolare, veloce ed efficiente.

Questa potrebbe causare alcuni problemi poiché non è centrata sullo zero e potrebbe uccidere alcuni neuroni definitivamente perché dà sempre 0 per valori negativi.

La ReLu è data dalla formula:

$$f(x) = x^+ = \max(0, x)$$

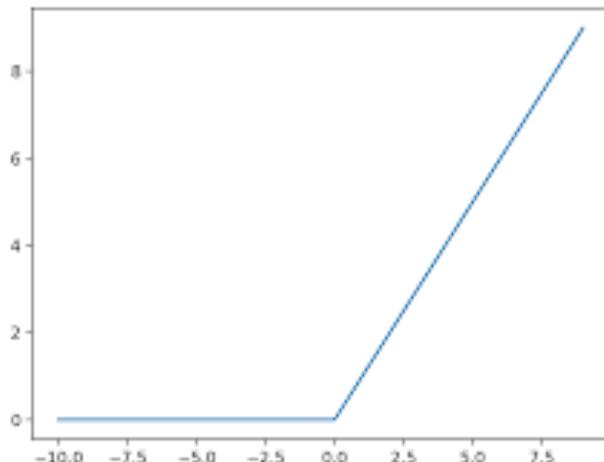


Figura 33 – ReLu

In conclusione per gli strati nascosti si utilizza una funzione di attivazione ReLu con le sue varianti mentre per lo strato finale si utilizza una SoftMax [14].

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*.

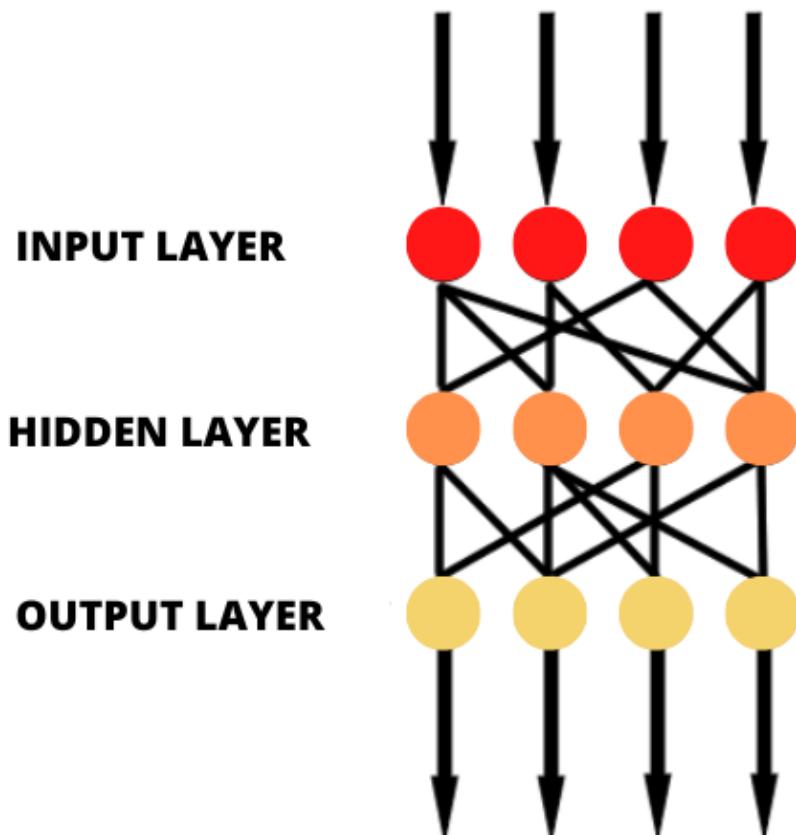


Figura 34 - FeedForward neural net con 1 hidden layer

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 errori 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**.

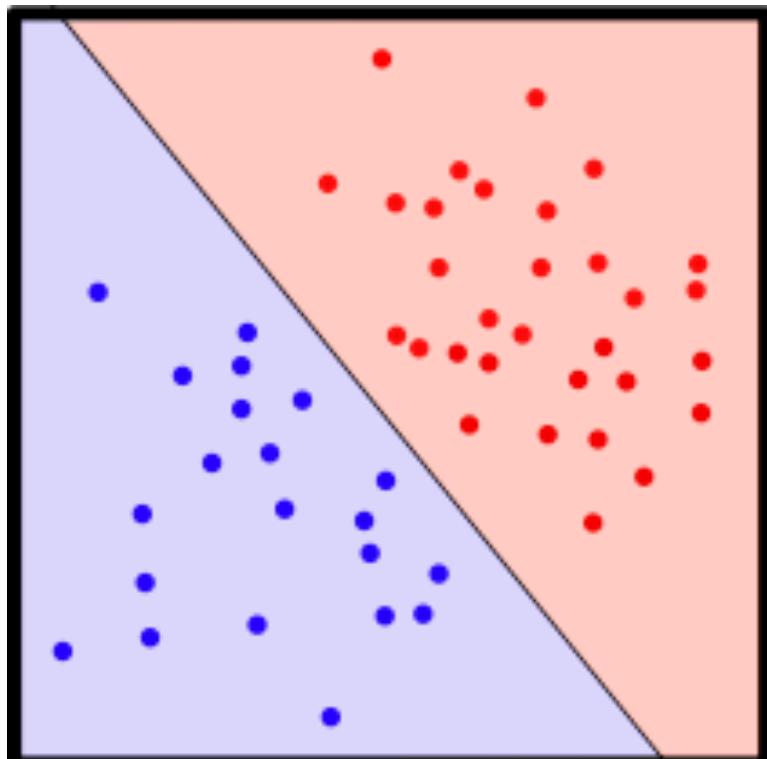


Figura 35 - Separabilità Lineare

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.

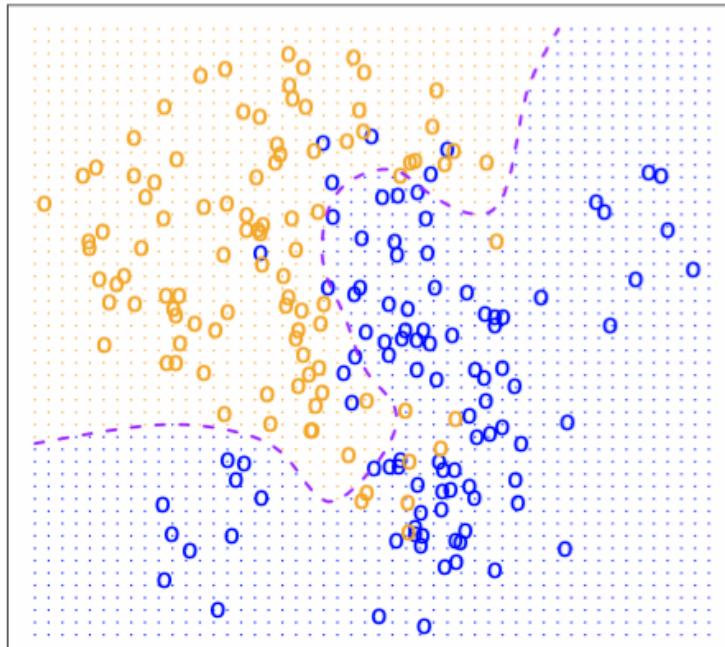


Figura 36 - Problema non separabile linearmente

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 3×3 .

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 [22].

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 percetroni"*.

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:

$$f(x) = \frac{1}{1 + e^{-x}}$$

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 percettore 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 x_1, x_2, \dots, x_m e invia un valore y 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.

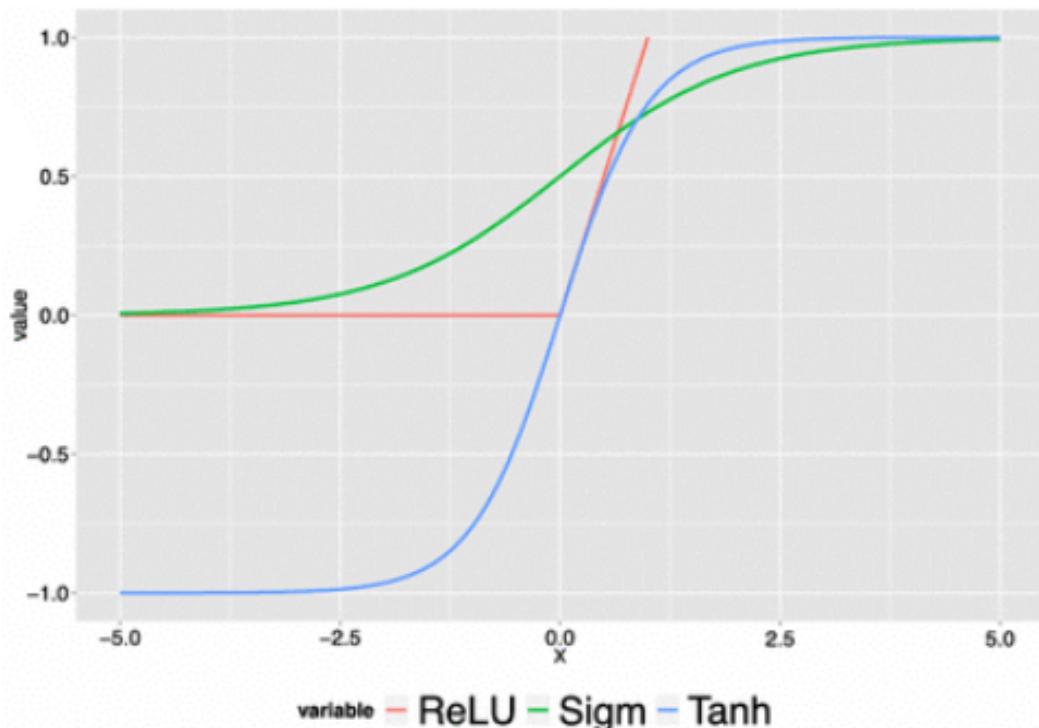


Figura 37 - Funzione di attivazione dei neuroni

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

OrienteBot 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 [23].

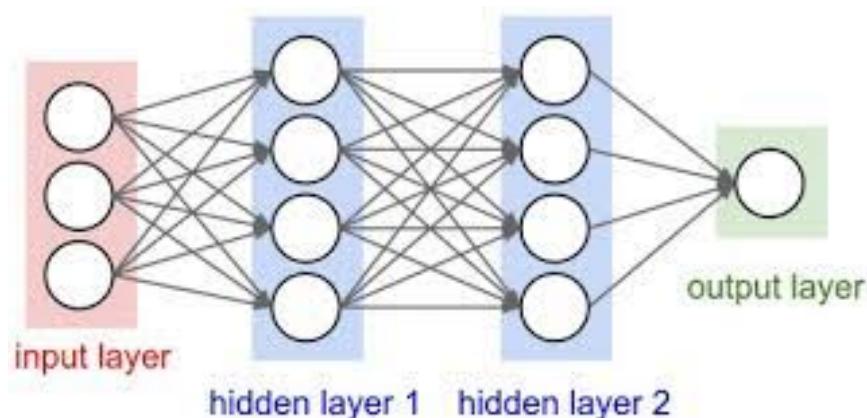


Figura 38 - FNN 2 hidden layers

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 39 - 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'interesse come si presenta OrientaBot:

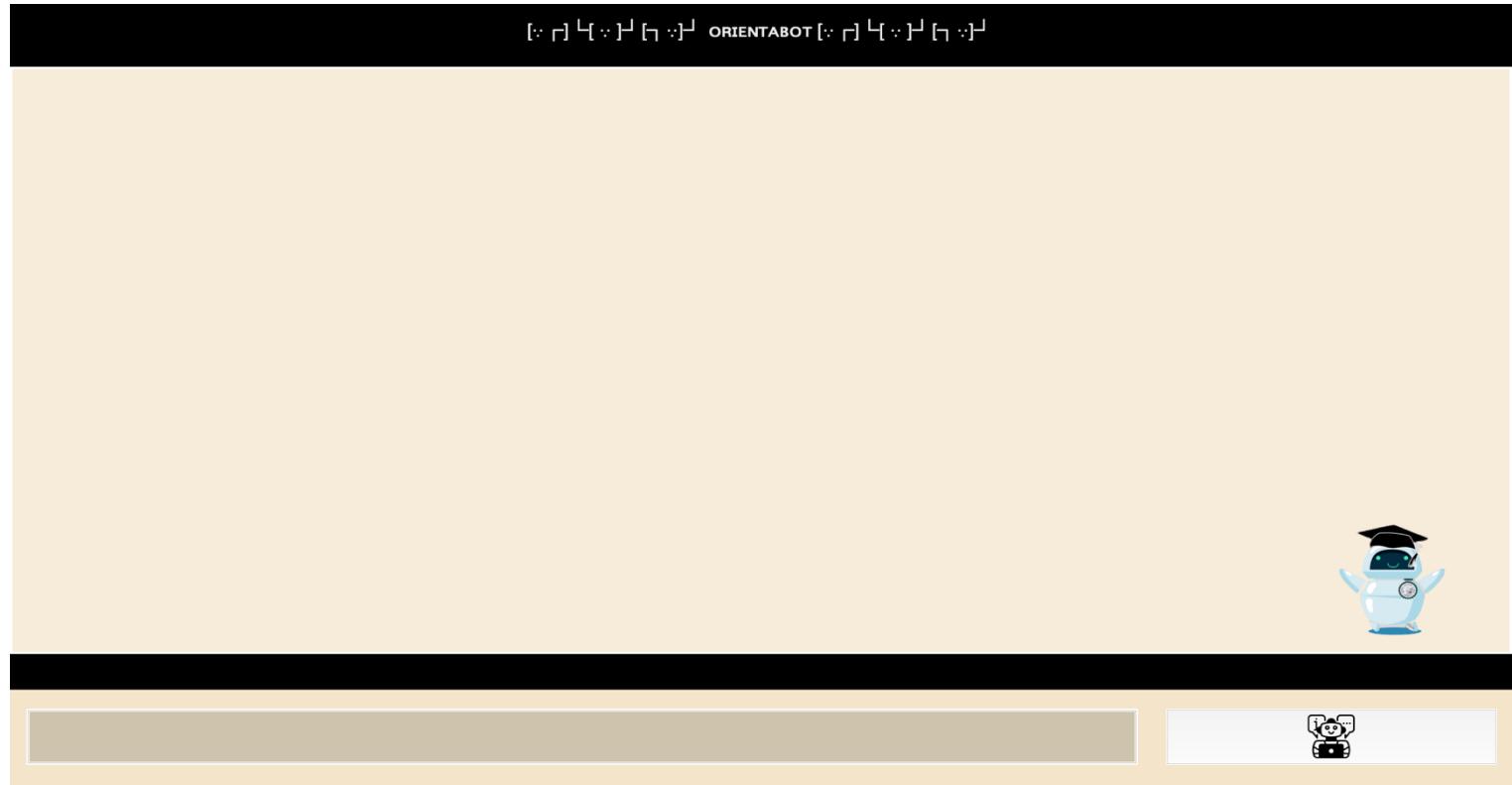


Figura 40 - OrientaBot GUI

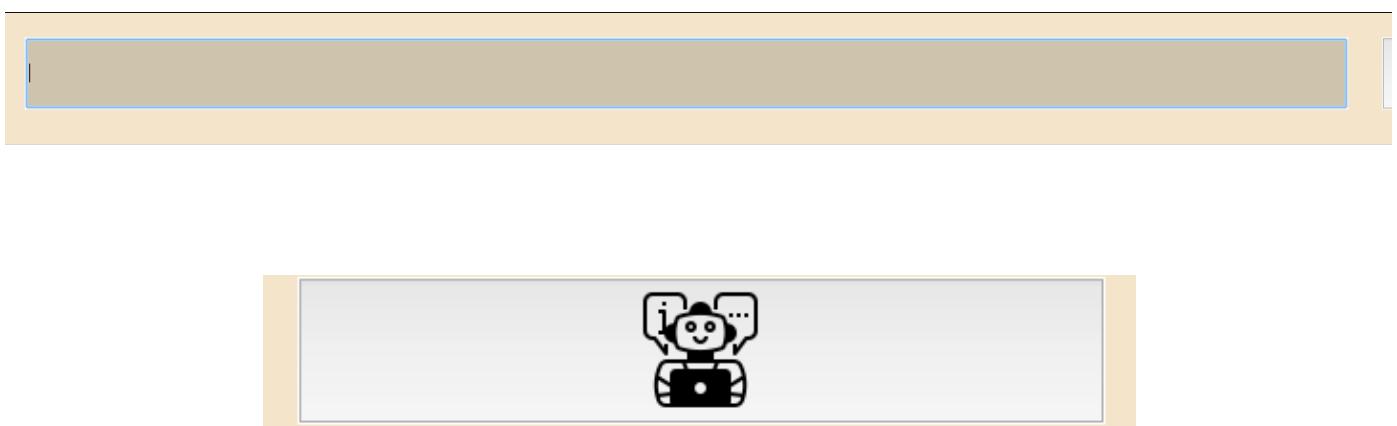


Figura 41 - Focus sulla barra di testo e pulsante di invio

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.

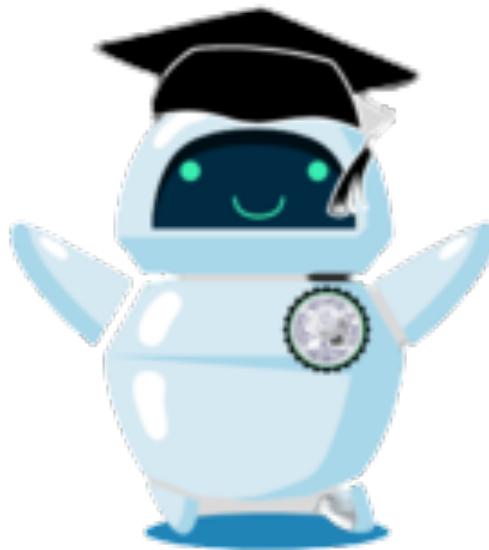


Figura 42 - Avatar OrientaBot

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 principi 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
- Dark mode

L'interfaccia minimal e familiare di OrientaBot possiede un'***affordance*** molto forte.

Più alta è l'affordance, più sarà automatico ed intuitivo l'utilizzo di un dispositivo o di uno strumento.

Ad esempio, l'aspetto di una maniglia dovrebbe far intuire al meglio e automaticamente come la porta vada aperta: se tirata, spinta, o fatta scorrere (una porta che si apre automaticamente al passaggio ha una scarsa affordance, poiché è molto poco intuitivo il suo funzionamento).

Tra gli oggetti con un'ottima affordance vi sono, ad esempio, la forchetta o il cucchiaio, strumenti che nel corso dei millenni sono stati affinati dall'uomo fino alla forma odierna, estremamente intuitiva e di semplicissimo utilizzo.

A primo impatto, infatti, qualsiasi utente riesce a capire come utilizzare il chatbot, non c'è bisogno di nessuna guida o manuale, nessun tutorial o supporto all'utente, tutto si racchiude in una chat tra l'user e il chatbot.

Questo aspetto non è da sottovalutare: una delle cause primarie di “abbandono del software” da parte di un utente è proprio la difficoltà di capire come utilizzarlo, ciò implica avere un'affordance molto scarsa.

Accade ciò perché siamo sempre più portati ad utilizzare applicazioni ben strutturate, anche per fare il task più semplice e quindi OrientaBot è stato ben studiato anche sotto questo aspetto.

In particolare OrientaBot ha una ***pattern affordance***: un'affordance stabilita dalle convenzioni; infatti, abbiamo la casella di testo in basso a sinistra dove l'utente può interagire e a destra il pulsante per inviare.

Non è da tralasciare è il tasto di invio: si è preferito usare un'icona piuttosto che il classico pulsante con scritta per rendere l'interfaccia più moderna, ciò non toglie che il nostro tasto possiede un'***affordance metaforica***: anche se il bottone non scritte, è facilmente intuibile il suo funzionamento [15].



Figura 43 - Affordance

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 principi 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:

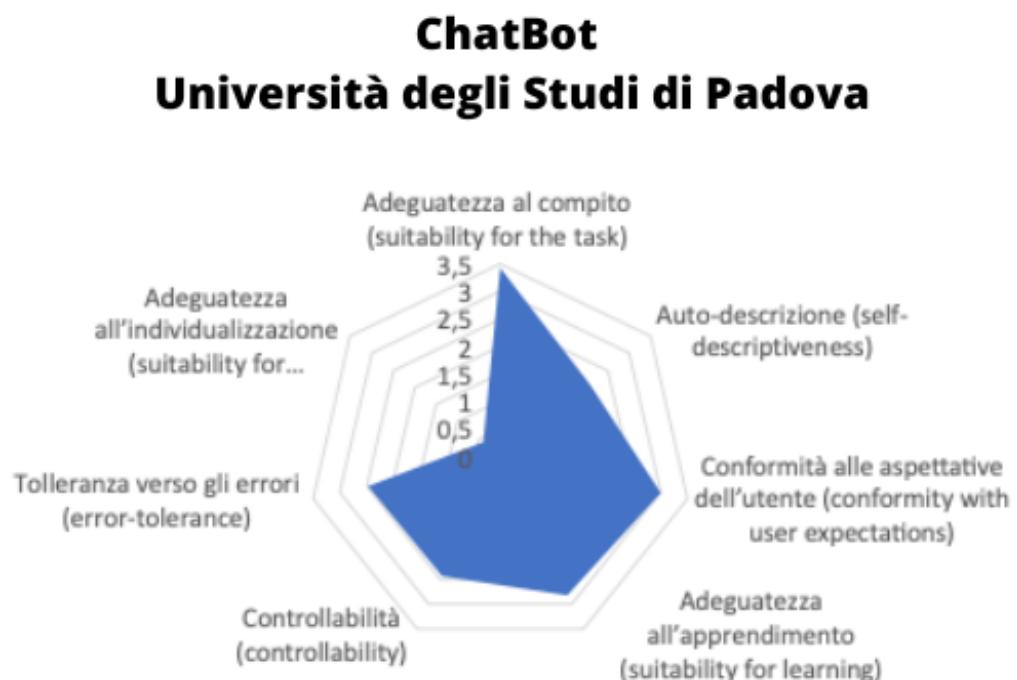


Figura 44 - Valutazione Chatbot Università di Padova

ORIENTABOT

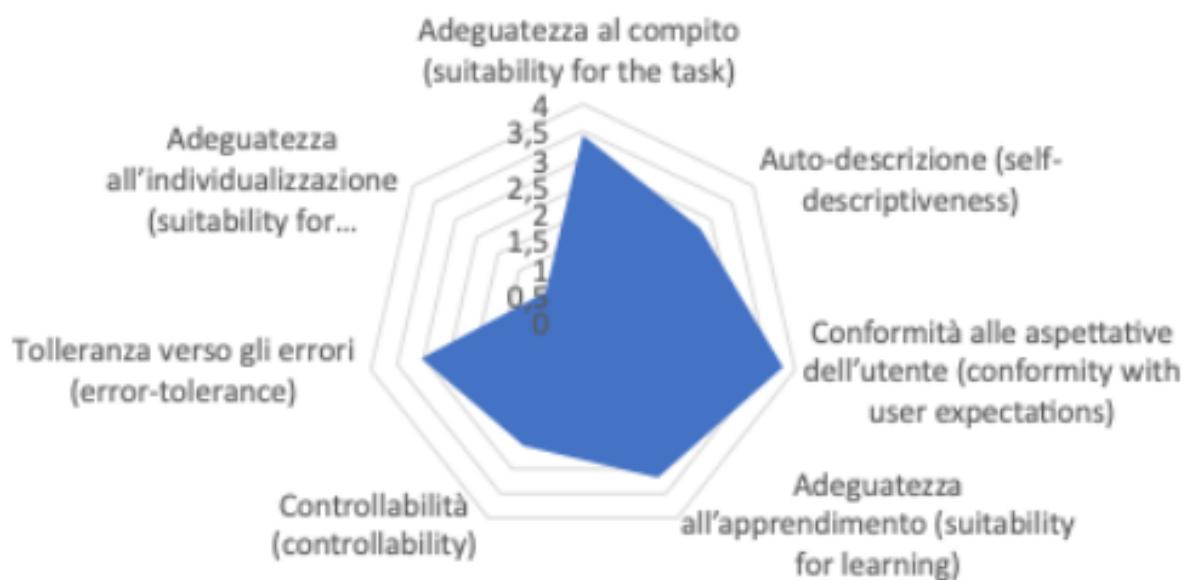


Figura 45 - Valutazione OrientaBot

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 più possibile OrientaBot, in modo tale da diventare un servizio effettivo e personale dell'Università degli Studi di Napoli Parthenope.

5.3 – MANUALE UTENTE PER L’ESECUZIONE

Dopo aver scaricato il codice sorgente di OrientaBot non resta che eseguire il programma per poterlo testare.

Per prima cosa dobbiamo eseguire il programma *train.py*: questo farà partire il processo di addestramento del nostro chatbot.

Ovviamente più il nostro training data è complesso e grande, più tempo verrà impiegato per l’addestramento, sempre restando nell’ordine dei secondi o al più di pochi minuti.

Da linea di comando, dopo esserci posizionati nel path corretto, basterà digitare:

```
MacBook-Pro-di-Dominick:Bot dominickferraro$ python3 train.py
```

Figura 46 - Esecuzione di train.py

Durante l’addestramento verranno mostrate a schermo informazioni interessanti, come l’elenco e il numero di tag del training data, l’elenco e il numero di parole univoche ricavate dallo *stemming* e il numero delle *epoch* e delle *loss*.

```
41 tags: ['saluti', 'arrivederci', 'feel', 'curiosit', 'nome', 'sediazione', 'servizi', 'uso', 'permesso', 'lavoro', 'cambioPercorso', 'spomatica', 'zonaStudio', 'manifestoInformatica', 'tolc', 'portineria', 256 unique stemmed words: ['', 'a', 'acced', 'ad', 'adisurc', 'aggior, 'arriv', 'arrived', 'atene', 'aul', 'aule', 'aver', 'azi', 'bar', 'b'cellul', 'centr', 'che', 'chi', 'chiam', 'chiar', 'chid', 'chied', ']
```

Figura 47 - Estratto dei Tag - Patterns

I valori di *loss* indicano la differenza rispetto agli stati target desiderati, noteremo che all'aumentare delle epoch il numero di *loss* tende a diminuire, questo è lo scenario che ci auspichiamo ogni volta che addestriamo un modello.

```
Epoch [100/1000], Loss: 0.0221
Epoch [200/1000], Loss: 0.0031
Epoch [300/1000], Loss: 0.0002
Epoch [400/1000], Loss: 0.0000
Epoch [500/1000], Loss: 0.0000
Epoch [600/1000], Loss: 0.0000
Epoch [700/1000], Loss: 0.0000
Epoch [800/1000], Loss: 0.0000
Epoch [900/1000], Loss: 0.0000
Epoch [1000/1000], Loss: 0.0000
final loss: 0.0000
```

Figura 48 - Rapporto Epoch – Loss

Fatto ciò, non ci resta che eseguire il programma *app.py* per lanciare l'applicazione e la sua interfaccia grafica:

```
MacBook-Pro-di-Dominick:Bot dominickferraro$ python3 app.py
```

Figura 49 - Esecuzione di OrientaBot

5.4 – 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.

Tu: Ciao	OrientaBot: Ciao!
Tu: A che ora iniziano le lezioni?	OrientaBot: Puoi consultare il sito della scuola, troverai gli orari di tutti i corsi di studi.
Tu: Qual è il sito della scuola?	OrientaBot: Il sito dell'Università è https://www.uniparthenope.it . Qui puoi trovare tutto ciò che cerchi. Visita anche https://orienta.uniparthenope.it per info su Placement, lavoro e tante opportunità.
Tu: Grazie mille	OrientaBot: Alla prossima

Figura 50 - 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:

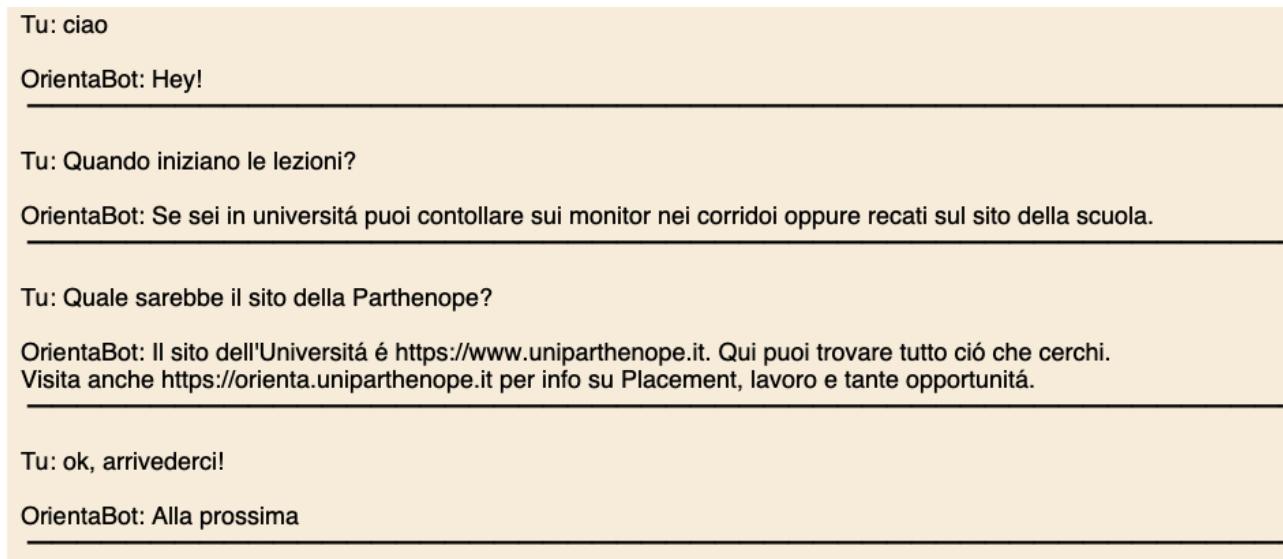


Figura 51 - Interazione con OrientaBot n.2

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

Da sottolineare 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 più 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.

Vediamo altri esempi di conversazione:

Tu: Come mi collego a internet?
OrientaBot: L'università Parthenope offre un ottimo servizio Internet libero per tutti gli studenti, basterà collegarsi alla rete EDUROAM e il gioco è fatto.

Tu: Come si chiama il rettore dell'Ateneo?
OrientaBot: Il nome del Magnifico Rettore è il Prof. Antonio Garofalo.

Tu: Come ti chiami?
OrientaBot: Puoi chiamarmi OrientaBot!

Figura 52 - 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.

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.

Tu: dove posso studiare
OrientaBot: Puoi studiare in sede tutti i giorni feriali, ci sono molti spazi dedicati allo studio su ogni piano dell'edificio oppure puoi recarti in biblioteca.

Tu: dove posso studiare?
OrientaBot: Puoi studiare in biblioteca, in aula studio o in uno dei banchi che si trovano sui piani dell'edificio!

Figura 53 - 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:

Tu: a chw ora inizadno le lezioni

OrientaBot: Puoi consultare il sito della scuola, troverai gli orari di tutti i corsi di studi.

Figura 54 - 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:

Tu: quali sonno i social dela Parthenope?

OrientaBot: Puoi seguire la Parthenope su tutti i social: Facebook, Instagram, Twitter e Youtube. Resta sempre aggiornato con noi!

Figura 55 - Interazione con OrientaBot n.6

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

Tu: wuaali sono i social dela parthenope?

OrientaBot: Puoi seguire la Parthenope su tutti i social: Facebook, Instagram, Twitter e Youtube. Resta sempre aggiornato con noi!

Figura 56 - 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.

Tu: **a che ora inizialalezione**

OrientaBot: Scusami, non credo di aver capito, prova a essere più preciso...

Figura 57 - Interazione con OrientaBot n.8

Tu: **questa frase non è nel training data**

OrientaBot: Scusami, non credo di aver capito, prova a essere più preciso...

Figura 58 - 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:

Tu: **a che ora è 123321 abc**

OrientaBot: Puoi andare in segreteria tutti i giorni dalle 09.00 alle 11.00

Figura 59 - 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.

All'avvio, OrientaBot presenta uno *splash screen* dove è possibile visualizzare il logo dell'applicazione.

Secondo *Kissmetrics*, il 40% delle persone abbandonano un sito web o un'applicazione dopo aver atteso 3 secondi, lo splash screen dà la possibilità all'applicazione di caricare tutti i dati e intrattiene l'utente [16].

Inoltre l'esperienza dell'utente inizia nel momento in cui il nostro agente conversazionale viene aperto per essere utilizzato.

Si è scelto di utilizzare uno splash screen per dare un tono all'intera esperienza progettata per gli studenti.

Gli splash screen progettati bene riconoscono la presenza dell'utente, facendolo sentire apprezzato e curato.

Inoltre è possibile che il sistema hardware su cui OrientaBot verrà eseguito potrebbe avere ritardi a causa di politiche di scheduling adottate o sovraccarico di CPU, applicando uno splash screen riusciremo a colmare secondi di attesa snervanti.

Per avere uno splash screen ottimo si deve progettare la schermata iniziale nel formato immagine corretto, è stata utilizzata una variante del logo di OrientaBot in PNG che rende la schermata più pulita e accattivante.

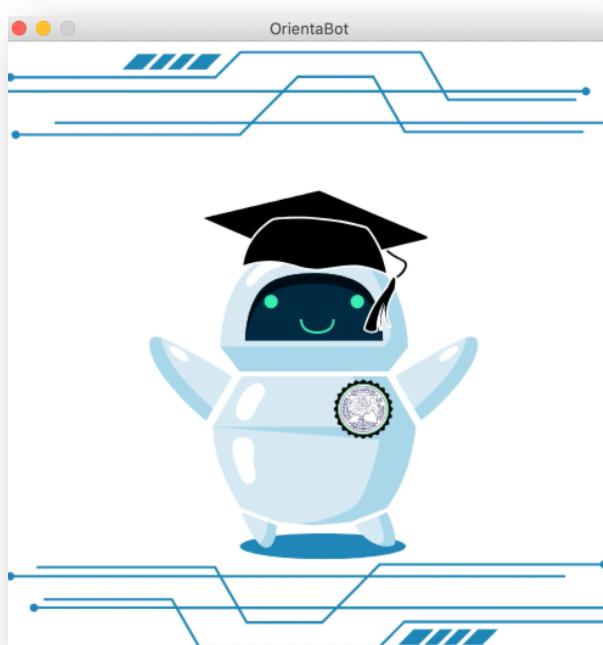


Figura 60 - OrientaBot splash screen

CAPITOLO 6

CONCLUSIONI E SVILUPPI FUTURI

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 61 - 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:

- a. 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.
- b. Il convertitore da analogico a digitale riceve i suoni da un file audio, misura le onde con grande precisione e le filtra per distinguere i relativi suoni.
- c. 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.

- d. I fonemi vengono quindi fatti passare attraverso una rete tramite un modello matematico che li confronta con proposizioni, parole e frasi conosciute.
- e. 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 62 - 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.



Figura 63 - OrientaBot UNI

Oltre a fornire un supporto informativo e di orientamento per gli studenti universitari dell'Università Parthenope, OrientaBot in futuro potrebbe essere utilizzato per raccogliere dati, ovviamente rispettando tutte le policy di privacy e le leggi, in modo da poter essere d'aiuto ai funzionari dell'Università. L'idea è raccogliere il parere degli studenti su alcuni temi, problematiche, iniziative, in modo da creare un insieme di dati analizzabile e processabile da umani e macchine.

Ad esempio se la domanda “*come effettuare l'immatricolazione*” oppure “*come prenotare una seduta di laurea*” risulta avere un numero elevato di occorrenze, vuol dire che probabilmente le procedure non sono abbastanza chiare e quindi occorre prendere provvedimenti.

Sicuramente ciò renderebbe OrientaBot meno efficiente ma il punto è proprio questo: l'obiettivo non è competere con i servizi universitari ma essere un'appendice e un supporto per tutti ciò che necessitano di informazioni o semplicemente vogliono avere conferma di ciò che già conoscono.

Questo servizio che OrientaBot potrebbe implementare potrebbe essere utile per migliorare alcune carenze del mondo universitario, dando voce a tutti gli studenti e aiutando in modo attivo gli addetti ai lavori.

Ciò renderebbe il chatbot una vera piattaforma universitaria, dove tutti possono accedere sia per avere un supporto alle informazioni, sia per esprimere i propri disagi che proporre nuove iniziative.

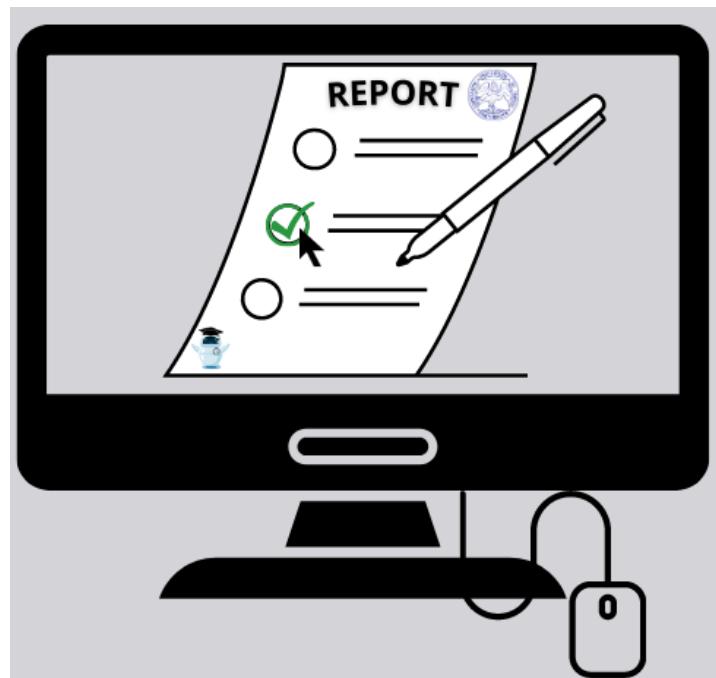


Figura 64 - Report OrientaBot

Questo elaborato si è concentrato su un argomento davvero interessante, di cui sentiamo parlare ogni giorno e che in futuro sarà una costante nella nostra vita.

L'idea di sviluppare un chatbot per l'orientamento degli studenti universitari nasce dalle mille difficoltà che ha un universitario nella ricerca di informazioni: molte volte ci sentiamo spaesati, con informazioni contrastanti e difficili da certificare.

La ricerca e lo studio dell'elaborazione del linguaggio naturale mi ha fornito conoscenze importanti, che variano dalla linguistica al machine learning, creando una coesione tra il mio percorso liceale e quello universitario.

La fase di testing è stata fondamentale per OrientaBot, ho visto nell'user un riscontro positivo, le informazioni fornite sono coerenti ed esaustive, centrando in pieno l'obiettivo prefissato dall'inizio.

È stato importante creare un solido training data, addestrando al meglio il chatbot per evitare di fornire informazioni errate o sconnesse.

I risultati sono stati ottimi, meglio di quelli pronosticati.

Spero che in futuro questo lavoro venga utilizzato nella mia Università, aiutando tutti gli studenti, i docenti e gli addetti ai lavori.



Figura 65 - OrientaBot X Università Parthenope

SITOGRAFIA

- [1] www.garanteedellaprivacy.it
- [2] [https://it.wikipedia.org/wiki/ELIZA_\(chatterbot\)](https://it.wikipedia.org/wiki/ELIZA_(chatterbot))
- [3] <https://www.wired.it/internet/web/2016/05/30/chatbot-umani>
- [4] www.marketingtechnology.it/chatbot-per-aziende
- [5] www.tech4future.info
- [6] www.poste.it/chatta-con-noi.html
- [7] https://elearning.uniparthenope.it/content/2/1_Introduction.pdf
- [8] www.python.org
- [9] <https://www.tiobe.com/tiobe-index>
- [10] www.nltk.org
- [11] www.numpy.org
- [12] www.pytorch.org
- [13] www.tkinter.org
- [14] www.netai.it
- [15] www.uxblog.it
- [16] www.kissmetrics.io

BIBLIOGRAFIA

- [17] Ewan Klein e Steven Bird - Natural Language Processing with Python – 2009
- [18] A. Menzli - Tokenization in NLP: Types, Challenges, Examples, Tools - 2023
- [19] N.L. - Natural Language Processing di base- nasoluca.it – 2020
- [20] J. PREECE, Y. ROGERS, H. SHARP, “INTERACTION DESIGN”, WILEY
- [21] G.K, Contextual Chatbots with TensorFlow, Chatbot Magazine, 2017
- [22] Gonzalez, Woods, Elaborazione delle Immagini Digitali, Terza Edizione, Pearson
- [23] A. Koutsoukas, J. Monaghan, Xiaoli Li, Jun Huan – Deep learning: Investigating deep neural networks hyper-parameters, 2017

RINGRAZIAMENTI

Alla fine del mio percorso, non posso far altro che ringraziare chi mi è stato accanto durante questo viaggio, ricco di esperienze, ostacoli, gioie e paure.

Un ringraziamento va al mio **Relatore**, il Prof. Antonino Staiano, grazie alla sua pazienza e dedizione, per i suoi consigli e per ciò che mi ha trasmesso durante gli anni di lezione e durante la stesura dell'elaborato. Auguro a tutti di avere nel proprio percorso universitario docenti di questo calibro.

Grazie ai miei **genitori**, Luigi e Caterina, senza di voi non sarei qui.

Mi avete sempre sostenuto, incoraggiato, anche nei momenti più difficili, senza far pesare mai nulla e permettendomi di studiare nelle condizioni migliori possibili, senza preoccuparmi di nulla.

Grazie per avermi capito tutte le volte in cui ero nervoso, quando non volevo parlare oppure quando vi cacciavo dalla mia stanza perché ero indaffarato.

Grazie per essere sempre disponibili e comprensivi.

Non ho mai avuto modo di ringraziarvi, forse per il mio carattere, ma queste parole di oggi valgono quotidianamente.

Grazie per ogni sforzo che fate per me ogni giorno, oggi è una vittoria anche per voi.

Grazie ai miei **fratelli**, Ivan e Carmine e grazie ad Elvira, nonostante la nostra differenza di età siete stati importanti per il mio percorso universitario.

Grazie Carmine per avermi trasmesso questa passione, anche indirettamente.

Auguro a Ivan un percorso universitario sereno, non arrendersi mai.

Grazie ai miei **nonni**, Chiara e Carmine, purtroppo il tempo è stato duro con noi e non possiamo abbracciarci e festeggiare, ma so che da qualche parte siete fieri di me.

Grazie ai miei **nonni**, Fausta e Domenico, per esservi interessati sempre di come andasse il mio percorso universitario, per le mille domande e curiosità. Vedere la soddisfazione sulle vostre facce è stato un motivo in più per andare avanti e dare il massimo.

Grazie alla mia **famiglia**, ai miei zii, cugini e parenti che in un modo o nell'altro avete sempre fatto il tifo per me, questo giorno di festa è anche per voi.

Durante questi anni, sono successe davvero tante cose, ho condiviso esperienze con persone speciali, che mi hanno dato tanto.

Grazie a **IPT**, Alfredo, Denny e Max, insieme abbiamo affrontato tante difficoltà, prendendoci molte soddisfazioni e passato nottate e giornate intere a lavorare insieme come una vera squadra.

Grazie a **Ilaria e Gianfranco**, oltre alla nostra amicizia, abbiamo condiviso soprattutto nell'ultimo anno gioie e dolori, studiato ore e ore, giorno e notte, senza mai stancarci, creando un gruppo divertente, attivo e produttivo.

Grazie a **Francesco, Giuseppe, Sara, Chiara e Gianluca**, con alcuni di voi ho condiviso bellissime esperienze all'interno dell'università, giornate intere a parlare di qualsiasi cosa, con altri invece ho passato bellissimi momenti di spensieratezza.

Da tempo ormai siete i miei amici, persone a cui tengo molto e so che è lo stesso per voi.

A tutti voi grazie per ogni momento passato insieme, vi auguro di realizzare i vostri sogni.

Concludendo, voglio ringraziare una delle persone più importanti per me, senza la quale non potrei essere qui.

Grazie **Margherita**, ci siamo conosciuti ormai quasi 10 anni fa, durante una lezione di matematica, la materia in cui non eravamo mai così preparati, che tanto ci faceva dannare ma anche divertire, da quel momento abbiamo condiviso insieme infinite esperienze, gioie, dolori, amicizie, viaggi e animali.

Grazie per ogni singolo momento insieme, da amici prima e da partner dopo, grazie per avermi supportato tutte le volte che non avevo più voglia di studiare, le tue parole erano magiche.

Grazie per ogni singolo giorno in cui hai ascoltato i miei problemi, le mie lamentele riguardo qualche esame, grazie per ogni singolo giorno in cui hai fatto il tifo per me.

Grazie per i tuoi riti scaramantici, per il bracciale, per tutte le volte che mi hai accompagnato a sostenere un esame e per tutte le interrogazioni fatte la sera prima di un orale, nonostante fossi stanca o impegnata con il tuo percorso universitario.

Le parole non sono abbastanza, se oggi sono qui è anche grazie a te.
Ti auguro di realizzare tutti i tuoi sogni, di essere sempre felice e di superare gli ostacoli che ogni giorno dovrai affrontare.

Grazie a **me stesso**, se posso, per non aver mai mollato, per essermi circondato di persone splendide e genuine.

Grazie a me stesso che dopo il Liceo ha ricominciato da zero, seguendo la propria passione, senza aver paura di cose nuove.

Sono stati anni difficili, soprattutto quelli in lockdown che mi hanno messo a dura prova.

Grazie all'**Università Parthenope** per avermi formato, fatto conoscere docenti professionali e disponibili e colleghi e amici ottimi.

Grazie a **chiunque** si senta appartenere al mio mondo, a chi ci è sempre stato e a chi ci sarà, sono orgoglioso di festeggiare questo traguardo con tutti voi.