**UNIVERSITA’ DEGLI STUDI DI NAPOLI**

**“PARTHENOPE”**



**SCUOLA INTERDIPARTIMENTALE**

**DELLE SCIENZE, DELL’INGEGNERIA**

**E DELLA SALUTE INFORMATICA**

**Dipartimento di Studi**

**Scienze e Tecnologie**

**Corso di Laurea in**

**Informatica**

*Tesi di Laurea in*

*Natural Language Processing e agenti conversazionali*

**Titolo tesi**

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

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

ANNO ACCADEMICO 2021/2022

**INDICE**

***ABSTRACT***

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

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

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

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

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

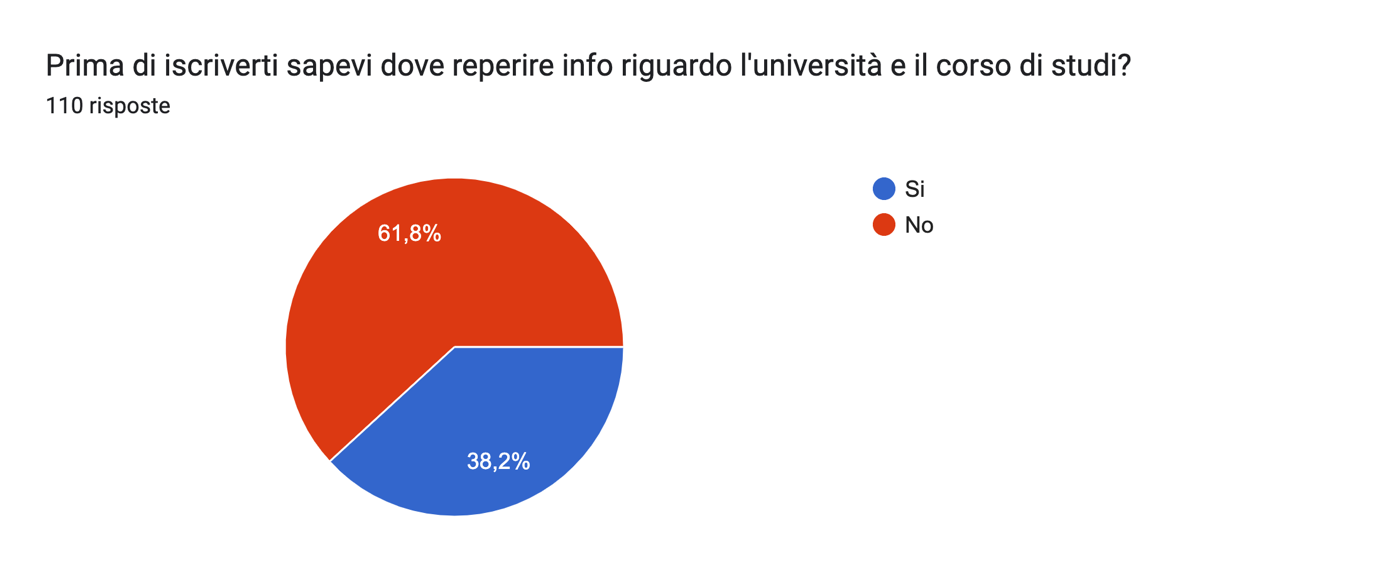
Secondo un sondaggio da me effettuato, su 110 studenti universitari:

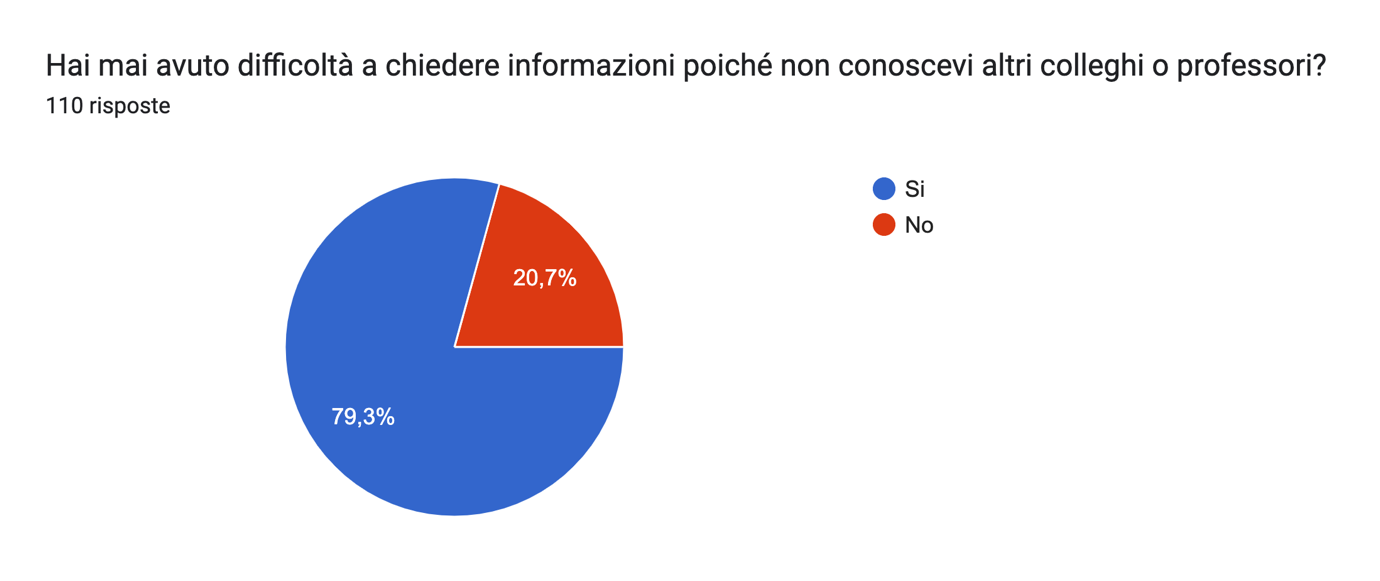
* **68** studenti non sapevano dove reperire informazioni prima di iscriversi all'universitá.
* **88** studenti hanno avuto difficoltà a chiedere informazioni poiché non conoscevano professori o altri colleghi.
* **104** studenti userebbero un Chatbot che risponde alle loro richieste riguardanti l'ambiente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

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

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

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





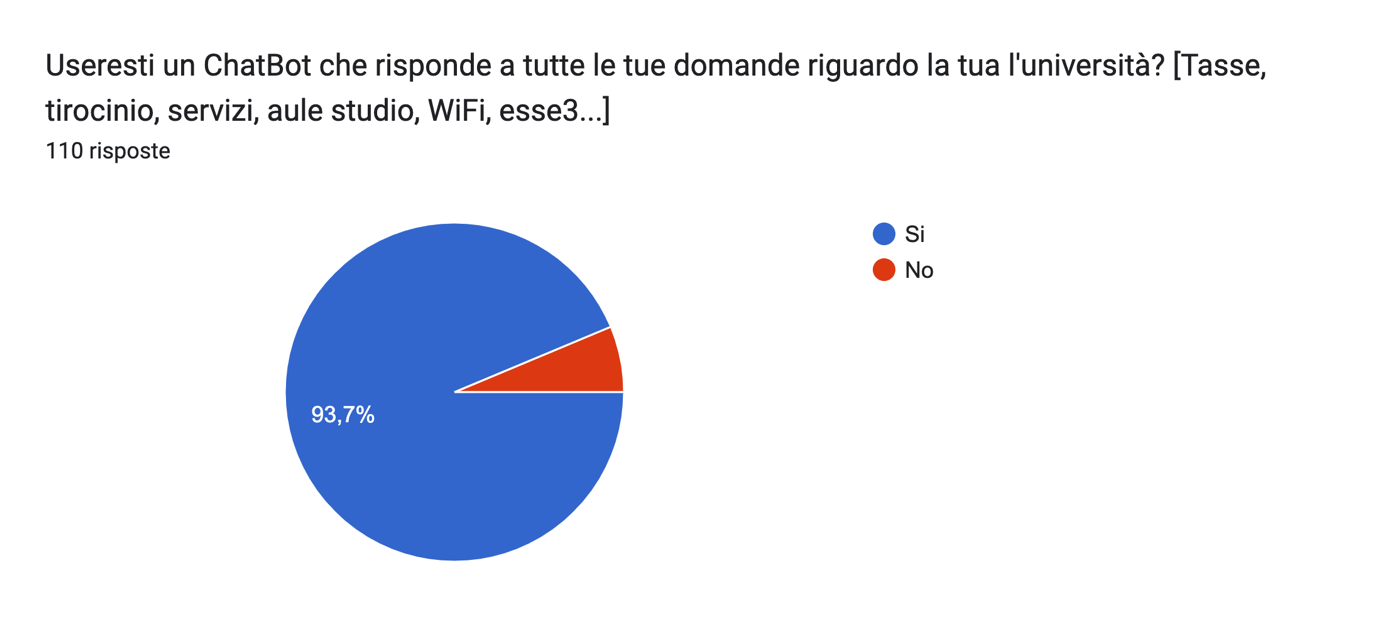
****

Figura 1 - Questionario agli studenti

**CAPITOLO 1 - INTRODUZIONE**

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

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

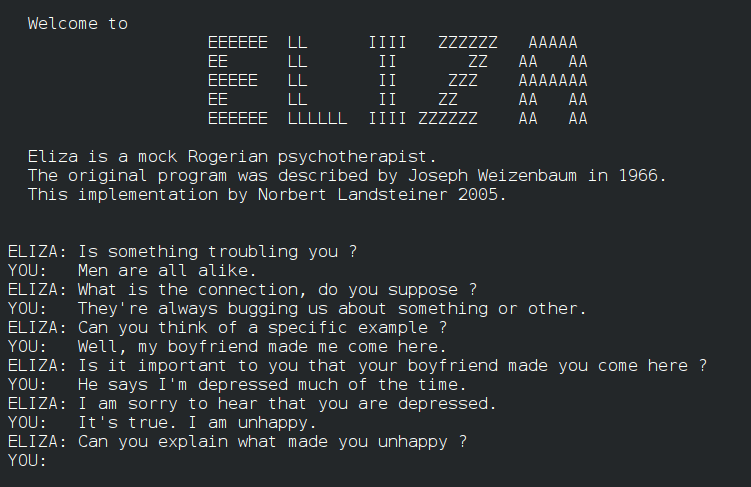


Figura 2 - Interfaccia di ELIZA

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

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

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

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

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

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

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

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

**1.1 – DIFFUSIONE DEI CHATBOT**

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

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

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

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

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

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

**1.2 – CLASSIFICAZIONE DI UN CHATBOT**

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

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

Immagine che contiene testo

Descrizione generata automaticamente

Figura 3 - Esempio Chatbot Poste Italiane

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

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

Abbiamo due tipi di chatbot:

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

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

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

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

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

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

**1.3 – FUNZIONAMENTO DI UN CHATBOT**

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

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

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

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

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

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

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

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

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

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

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

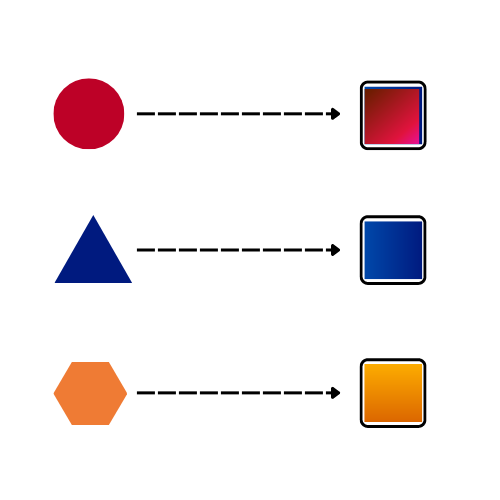


Figura 4 - Associazione Pattern → Response

Ecco un esempio: acquisto di uno smartphone.

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

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

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

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

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

Di seguito sono riportati i campi principali della NLP:

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

Immagine che contiene testo, elettronico

Descrizione generata automaticamente

Figura 5 - Campi NLP

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

**1.3.1.1 – TOKENIZATION**

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

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

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

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

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

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

**ESEMPIO:**

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

Tokenization per frasi:

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

Tokenization per parole:

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

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

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



Figura 6 - I sei significati di عقد

**1.3.1.2 – STEMMING**

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

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

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

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

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

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

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

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

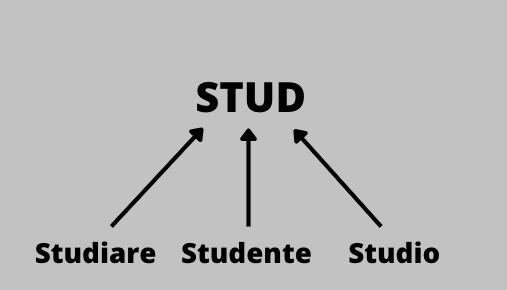


Figura 7 - Stemming

**1.3.1.3 – LOWERING AND IGNORE PUNCTUATION**

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

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

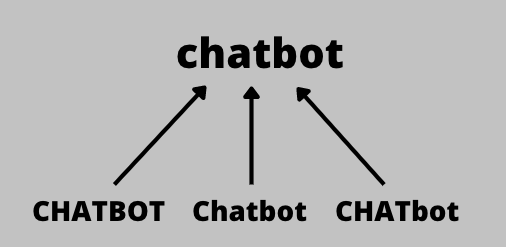


Figura 8 - Lowering

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



Figura 9 - Lowering + Ignore Punctuation

**1.3.1.4 – LEMMATIZATION**

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

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

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

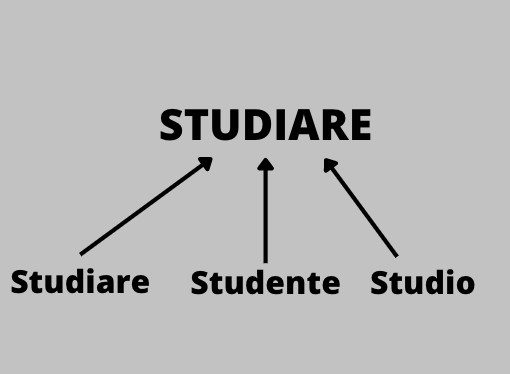


Figura 10 – Lemmatization

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

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

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

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

**2.1 – SCENARI**

Di seguito si elencano alcuni scenari del software sviluppato:

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

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

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

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

**2.2 – REQUISITI FUNZIONALI E NON FUNZIONALI**

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

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

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

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

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

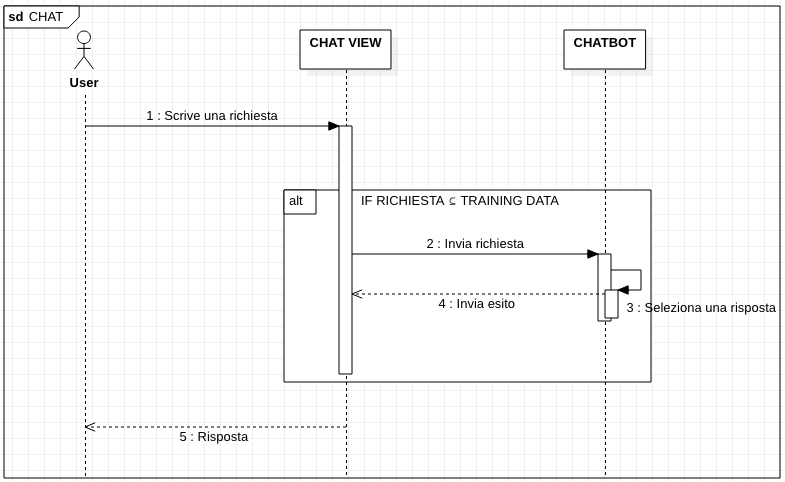
**NFR1:** Usabilità: il sistema dovrebbe essere intuitivo da usare e l’interfaccia utente dovrebbe essere semplice da capire.

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

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

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

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

****Figura 111 – Sequence Diagram Chatbot

Il seguente Sequence Diagram esplicita il funzionamento del Chatbot.

L’utente scrive una richiesta attraverso una chat, la richiesta viene inviata al chatbot che la analizza, verifica se coincide con qualche *tag* del training data, se così seleziona una risposta dal 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.

**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.

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

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

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

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

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

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

**CAPITOLO 3 – STRUMENTI PER LO SVILUPPO**

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

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

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

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

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

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

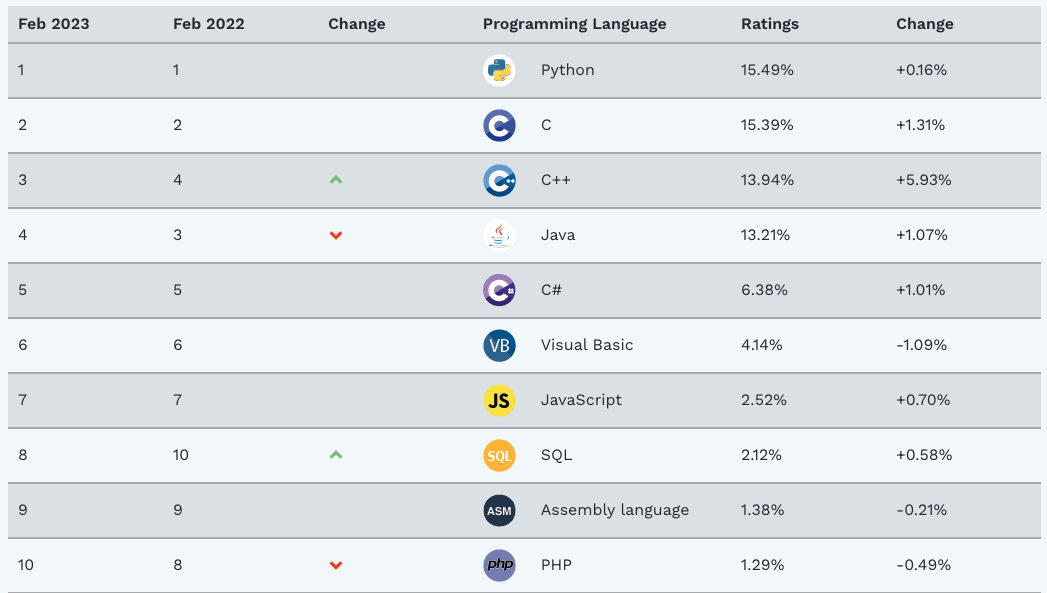


Figura 12 - TIOBE Top 10 Programming Language

**3.1 – LIBRERIE E FRAMEWORK**

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

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

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

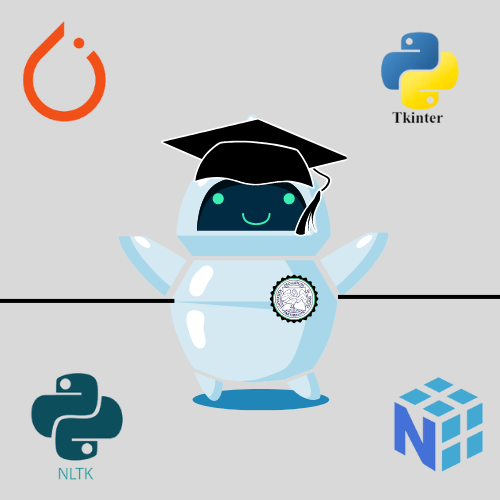


Figura 13 - OrientaBot Package

**3.1.1 – NATURAL LANGUAGE TOOLKIT**

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

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

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

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

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

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

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

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

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

Per installare NLTK su un Sistema Operativo Windows:

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

**3.1.2 – NUMPY**

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

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

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

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

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

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



Figura 14 - Funzioni di NumPy

**3.1.2 – PYTORCH**