UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

CAREFREE: Chatbot per l'Identificazione di Privacy nell'Ingegneria dei Requisiti

RELATORE

Prof. Fabio Palomba

TUTOR

Dott. Francesco Casillo

Università degli Studi di Salerno

CANDIDATO

Piero Agosto

Matricola: 0512106550

Anno Accademico 2021-2022

Questa tesi è stata realizzata nel

Siate affamati, siate folli

-Steve Jobs

Sommario

In un mondo sempre più digitale, la privacy delle informazioni personali sta diventando sempre più una questione di grande importanza: c'è una maggiore esposizione dei dati personali e aziendali a rischio di violazione della privacy ed è per questo, fondamentale garantire che tali informazioni vengano protette e trattate in modo sicuro.

Il chatbot sviluppato in questa tesi potrebbe rappresentare un buon contributo alla raccolta e gestione dei requisiti di privacy nello sviluppo software. Esso consente di identificare i contenuti di privacy presenti in User Story, garantendo che vengano adeguatamente considerati durante lo sviluppo, rendendo più efficace la comunicazione tra sviluppatori e consumatori, grazie all'utilizzo di un canale apposito in Slack. In questo modo, si può contribuire a prevenire violazioni della privacy e garantire che i dati personali vengano trattati in modo sicuro.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto Applicativo

L'Ingegneria dei Requisiti è una delle attività più complesse nell'ingegneria del software: incomprensioni e disattenzioni possono portare ad avere un sistema scarno. Oltretutto, la raccolta dei requisiti è una fase che richiede molta attenzione. Essi si dividono in requisiti funzionali e requisiti non funzionali: i requisiti funzionali rappresentano un insieme di funzionalità o servizi che il sistema da sviluppare, deve offrire. Essi descrivono anche il comportamento del sistema a fronte di particolari input e come esso dovrebbe reagire in determinate situazioni. I requisiti non funzionali, invece, rappresentano i vincoli e le proprietà/caratteristiche relative al sistema, come vincoli di natura temporale, vincoli sul processo di sviluppo e sugli standard da adottare. Tra i requisiti non funzionali, quelli di privacy sono molto importanti perché potrebbero riguardare dati personali, i quali devono essere protetti perché rappresentano informazioni riservate e sensibili che possono essere utilizzate per scopi dannosi. Inoltre, le aziende hanno la responsabilità di proteggere i dati dei propri clienti e dipendenti, in conformità con le leggi sulla privacy come il Regolamento Generale sulla Protezione dei Dati (GDPR) dell'Unione Europea. Entrata in vigore il 25 maggio 2018, stabilisce standard elevati per la protezione dei dati personali e

impone sanzioni severe per le violazioni di privacy.
A questo scopo, un chatbot che identifichi la privacy all'interno di User Stories è molto importante perché può essere utilizzato nell'Ingegneria dei Requisiti per supportare i team di sviluppo, progettazione e test, aiutando ad identificare e risolvere eventuali problemi di privacy prima che il software venga rilasciato. In questo modo, il Chatbot può contribuire a garantire che il sistema abbia le caratteristiche necessarie per soddisfare le normative sulla privacy e la protezione dei dati personali degli utenti.

1.2 Motivazioni e Obiettivi

L'Ingegneria dei Requisiti si occupa di raccogliere, analizzare e gestire le esigenze degli utenti per garantire che il software sviluppato soddisfi le loro necessità. Tuttavia, una delle sfide più grandi che si presentano in questa fase è garantire che i dati privati vengano gestiti in modo sicuro e protetto.
L'obiettivo principale di questa tesi sarà quello di affrontare questa sfida sviluppando CAREFREE (ChAtbot pRivacy idEntifier For Requirements EnginEering), un chatbot utilizzabile nell'Ingegneria dei Requisiti per l'identificazione della privacy, permettendo così di prevenire la violazione dei dati personali degli utenti. Inoltre, si cercherà di valutare la sua efficacia attraverso una serie di test e valutazioni e di integrarlo nell'ambiente di lavoro, per consentirne l'utilizzo in un contesto reale.

1.3 Approccio

Lo sviluppo di CAREFREE è stato possibile soprattutto grazie all'utilizzo di due tecniche come il Transfer Learning e l'NLP.
Il **Transfer Learning** è una tecnica di apprendimento automatico che permette di sfruttare conoscenze acquisite da un modello addestrato in un compito simile per essere utilizzate in un altro compito. Questo significa che, anziché addestrare un modello da zero per ogni nuovo compito, è possibile partire da un modello pre-addestrato e personalizzarlo per adattarlo al nuovo compito, avendo così un risparmio di tempo ed una maggiore precisione. In questo caso, è stata utilizzata una rete neurale convo-

luzionale pre-addestrata che ha permesso al bot di identificare i contenuti di privacy con un'alta precisione.
L'**NLP** (Natural Language Processing), invece, è una branca dell'intelligenza artificiale che si concentra sulla comprensione e l'analisi del linguaggio naturale. Le tecniche NLP utilizzate sono:

- **Tokenizzazione:** divisione del testo in token, ossia singole parole o frasi significative.
- **Part of Speech (POS) Tagging:** identificazione e etichettatura delle parti del discorso.
- **Named Entity Recognition (NER):** identificazione e classificazione delle entità, come persone, organizzazioni, date e luoghi, all'interno di un testo.
- **Dependency Parsing:** analisi della struttura grammaticale di una frase e identificazione dei rapporti tra le parole all'interno del testo.
- **Keyword Searching:** permette di individuare la presenza di specifiche parole all'interno del testo, in questo caso le parole ricercate sono quelle relative alla privacy.

Queste tecniche sono state utilizzate per analizzare i contenuti delle User Stories e determinare se esse contenevano parole di privacy o meno. Ulteriori dettagli sono presenti nel Capitolo 3.

1.4 Struttura della Tesi

La tesi è strutturata in 6 capitoli.
Il primo capitolo introduce il contesto nel quale il chatbot potrebbe essere applicato e le motivazioni che hanno portato ad avere l'idea di sviluppare CAREFREE.
Il secondo capitolo introduce i bot/chatbot, spiegando la loro architettura; inoltre definisce la User Story ed analizza i lavori presenti in letteratura: sia quelli nei quali vengono utilizzate le User Stories e sia quelli che riguardano l'individuazione di informazioni sensibili. L'ultima parte del secondo capitolo analizza ciò che è stato

fatto con i bot nell'Ingegneria dei Requisiti, spiegando inoltre quali sono i vantaggi di utilizzare CAREFREE e in cosa si differenzia dagli altri lavori effettuati.
Il terzo capitolo spiega nei dettagli il modo con cui il bot elabora le User Stories, gli strumenti che utilizza e come questo metodo è stato validato.
Il quarto capitolo descrive lo sviluppo di CAREFREE, partendo dalle tecnologie utilizzate, la sua architettura e l'implementazione.
Nel quinto capitolo vengono mostrati vari casi d'uso del bot.
Il sesto ed ultimo capitolo è quello delle conclusioni e degli sviluppi futuri che potrebbero essere portati sul bot.

CAPITOLO 2

Background e stato dell'arte

Nel presente capitolo, si introduce, innanzitutto, cosa sono i bot, come sono "nati" e la loro architettura. In seguito si parlerà delle User Stories, cosa sono, come vengono utilizzate e verranno esaminati i vari lavori presenti in letteratura per quanto riguarda l'utilizzo dei bot nell'Ingegneria dei Requisiti e come essi possono essere integrati con le User Stories per rendere più efficace ed efficiente il processo di sviluppo software e garantire la protezione dei dati degli utenti.

2.1 Bot e Chatbot

Nel 1950, Alan Turing propose un criterio, chiamato oggi *Test di Turing*, che è in grado di determinare se una macchina è capace di pensare o meno. Per soddisfare questo criterio un software deve fingere di essere umano in una conversazione in tempo reale in modo che l'interlocutore non sia in grado di distinguere, basandosi solo sul contenuto della conversazione, se stia conversando con un programma o con un essere umano.
Nel 1966, venne sviluppato un programma chiamato *Eliza* con il quale si dimostrò che era possibile far interagire un essere umano con un computer utilizzando il linguaggio naturale. Questo era stato creato da Joseph Weizenbaum, un Professore

dell'MIT (Massachusetts Institute of Technology) negli Stati Uniti ed è a tutti gli effetti il primo bot/chatbot. Esso simulava un psicoterapeuta Rogersiano, rispondendo al paziente con domande ottenute dalla riformulazione delle affermazioni del paziente stesso.
Da questo si può capire cos'è un chatbot: esso è un software con il quale si può avere una conversazione, utilizzando un linguaggio naturale, attraverso un'interfaccia conversazionale come ad esempio Telegram, un'app di messaggistica, che mette a disposizione dei propri utenti la possibilità di ricevere messaggi in automatico (seguendo i propri interessi) grazie alla creazione di bot specializzati.

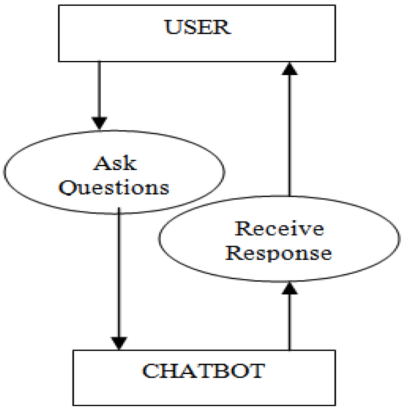

Figura 2.1: Use Case del ChatBot

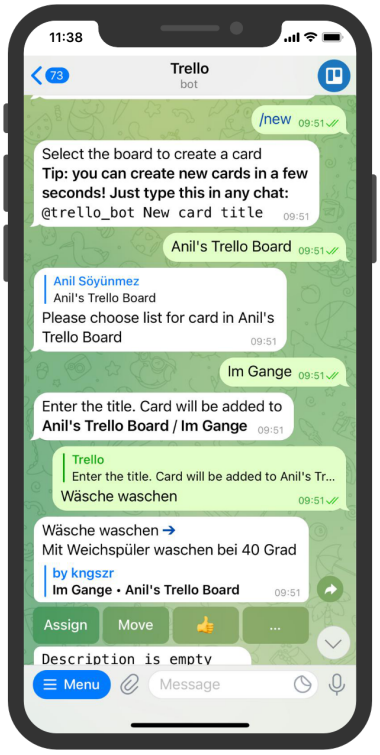

Figura 2.2: Trello Bot in Telegram

2.1.1 Architettura dei bot

Per capire meglio come funziona un bot è doveroso introdurre alcuni concetti sulla loro architettura interna.
Essi sono divisi in 3 sottosistemi:

1. **Application Interface**: si occupa di offrire al consumatore un'interfaccia conversazionale col bot. Esistono diverse tipologie di integrazioni:

- **Custom Integrations**: integrazioni in nuove applicazioni, esse possono variare tra portali web ed app Android o iOS.
- **Messaging Platform Integrations**: integrazioni con piattaforme di messaggistica popolari che permettono di integrare il bot realizzato. (Figura 2.2)

2. **Conversational Subsystem**: di questo sottosistema fanno parte tutte le componenti che hanno l'incarico di processare il linguaggio naturale. Ne fanno parte:

   - **Voice Utils**: componente essenziale per le applicazioni che hanno un interfaccia vocale, in quanto i bot cercano di lavorare sempre con input testuali anziché con input vocali. Questa componente permette quindi di convertire il messaggio vocale in messaggio testuale. Permette inoltre di trasformare il messaggio testuale in vocale ad esempio per fornire una risposta.
   - **Intent and Parameter**: gli intent sono le diverse classi di input che l'agente conversazionale può prevedere, mentre i parametri sono gli attributi della richiesta fatta al bot ed entrambi sono necessari per dare una risposta.
   - **Response Generator**: è la componente che permette di generare una risposta da dare all'utente, in modo da fornirgli un feedback al precedente input. Le risposte generate possono essere statiche oppure dinamiche ed in questo caso comprendono i paramentri che l'utente ha fornito in input.
   - **Flow Manager**: questa componente si occupa del flusso della conversazione, in particolare tiene traccia dello stato attuale della conversazione, e in base agli input forniti riesce a capire qual è il passo successivo.

3. **Other Components**: per altre componenti si intende quell'insieme di elementi che devono fornire le funzionalità dell'applicazione.

Fondamentalmente, un chatbot funziona basandosi su una *Knowledge Base* che è una raccolta di domande e le loro risposte vengono attivate sulla base di alcune parole

chiave individuate nella conversazione. Essa è il cervello dietro il chatbot: se ben progettata consente ad esso di comprendere le richieste degli utenti, migliorare la sua capacità di rispondere direttamente agli input degli utenti, e interagire efficacemente dando le risposte appropriate. I progressi nell'intelligenza artificiale, e più specificamente nel Machine Learning, hanno permesso di creare agenti conversazionali molto più avanzati. Questa evoluzione è dovuta all'adozione degli attuali sistemi di analisi del linguaggio naturale (Natural Language Processing) che sono potenti e capaci di auto-migliorarsi. I chatbot più complessi, grazie alla tecnica del Machine Learning, riescono ad interpretare i dati che vengono raccolti durante le conversazioni per imparare e diventare più efficaci. Maggiori sono i dati a disposizione, maggiore sarà l'efficienza che svilupperà il bot.
Questo strumento, nelle mani di un'azienda, diventa fondamentale quando, ad esempio, vuole capire quali sono le domande che gli utenti fanno più spesso o per migliorare i servizi offerti ai propri clienti. In particolare, per esempio l'assistente di ricerca di Google si occupa proprio di migliorare l'esperienza dell'utente, aiutando a effettuare delle ricerche, oppure pianificare eventi e promemoria ai quali poi si legano altri servizi.

2.2 Che cos'è una User Story

Una User Story rappresenta una pratica utilizzata per "catturare" le esigenze degli utenti esprimendo in maniera informale e non dettagliata, caratteristiche, funzioni e requisiti per il sistema da realizzare.
Le user stories hanno le loro origini nell'extreme programming, una metodologia di sviluppo del software che enfatizza la scrittura di codice di qualità e la rapidità di risposta ai cambiamenti di requisiti. Kent Beck, il fondatore dell'extreme programming, ha dichiarato che le user stories sono state create per soddisfare le esigenze specifiche dello sviluppo software, condotto da piccoli team che si trovavano a confrontarsi con requisiti mutevoli e vaghi [1]. Secondo un sondaggio del 2014, le user stories sono diventati i requisiti più utilizzati in un ambiente agile [2].Infatti, esse sono integrate in molte tecniche agili, come la pianificazione del rilascio e dell'iterazione e il monitoraggio dell'avanzamento di un progetto.

Pertanto, le user stories sono sempre più un mezzo di comunicazione con gli utenti finali e i clienti ed inoltre costituiscono le basi per sviluppare le relative funzionalità ed integrarle nel sistema [3]. Esse vengono utilizzate soprattutto in Scrum, un framework agile per la gestione del ciclo di sviluppo software, creato per gestire progetti e prodotti software.
Sono un elemento all'apparenza semplice, ma molto efficace in quanto consentono di focalizzarsi su necessità e bisogni dell'utente.
Seguono tipicamente una struttura formata dal *who* (chi), *what* (cosa) e *why* (perché) di un requisito.
Il modello proposto da Cohn[4] è il seguente:

> As a <type of user/role>, I want to <feature> so that <some reason>.

Ad esempio: *As a user, I want to click on the address, so that it takes me to a new tab with Google Maps.*

2.2.1 Approcci effettuati con User Stories

Le User Stories sono state utilizzate per diversi framework e metodologie, come quelli proposti per analizzare la qualità delle stesse attraverso la loro analisi sintattica, con l'obiettivo di renderle più accurate e chiare [5, 6, 7]. Altri lavori sono stati effettuati con lo scopo di generare automaticamente dei diagrammi da User Stories per fornire una rappresentazione visuale di esse o per evidenziare potenziali problemi nella loro definizione [8, 9, 10].
Ad esempio Gilson et al. [11] hanno sviluppato un sistema in grado di generare degli Use Case. Sempre gli stessi hanno mostrato che le User Stories potrebbero avere un grande impatto sulle decisioni iniziali nello sviluppo software, perché potrebbero fare riferimento ad attributi che definiscono la qualità del software, fornendo così agli architetti del software la possibilità di farsi un'idea delle conseguenze delle possibili decisioni di design. [12]. In particolare utilizzano le tecniche del machine learning per classificare se le User Stories si riferiscono ad attributi di qualità.

Nonostante questi lavori, è difficile trovare un tool che identifichi contenuti di privacy in User Stories. Il nostro obiettivo è proprio quello di permettere questo, utilizzando un bot che notifichi gli sviluppatori quando una nuova US viene caricata dall'utente.

2.3 Individuare informazioni sensibili

Con la rapida crescita dei servizi internet e degli smart devices, gli utenti continuano a condividere grandi copie di dati giornalmente. Mentre questo può essere utile in alcuni campi, come ad esempio la ricerca comportamentale o medica, produce anche delle violazioni di informazioni sensibili degli utenti.
Negli ultimi anni, molti sforzi sono stati dedicati alla divulgazione di privacy, sia per facilitare il lavoro degli analisti e degli sviluppatori [13, 14] e sia per definire una tassonomia linguistica della privacy per l'analisi dei contenuti [15, 16]. Molti degli approcci per l'individuazione della privacy si concentrano sul riconoscimento automatico di informazioni sensibili in testo non strutturato [17, 18, 19].
Tra questi è importante il lavoro fatto da Tesfay et al.[20], con il quale, utilizzando un approccio basato sul machine learning, è possibile individuare e classificare informazioni di privacy a partire da un testo generato dagli utenti. L'obiettivo principale di questo lavoro è di avvisare gli utenti al momento della divulgazione, attirando la loro attenzione sul fatto che le informazioni sensibili vengono condivise e consentendo loro di prendere una decisione in maniera consapevole.
Un sistema simile è stato sviluppato da Neerbeky et al.[21]: la differenza principale è che il sistema precedente utilizza come dataset, un insieme di tweet, mentre questo utilizza un insieme di documenti contenenti informazioni sensibili.
Miyazaki et al. [22] hanno sviluppato un tool per migliorare la raccolta di requisiti di privacy nelle fasi iniziali dello sviluppo software. Questo viene fatto utilizzando un database generale di requisiti di privacy, derivati da leggi sulla privacy e requisiti di privacy empirici.
Benbenisty et al. [23] hanno sviluppato un tool per prevenire falle di privacy, individuando requisiti di privacy appropriati durante il System Design.

Altri approcci sono stati realizzati per migliorare le definizioni dei requisiti dell'utente e facilitare le decisioni prese durante il processo di sviluppo software. Ad esempio Riaz et al. hanno proposto un tool basato sul machine learning che prende in input un insieme di testi scritti in linguaggio naturale e identifica automaticamente frasi rilevanti dal punto di vista della sicurezza in base a degli obiettivi di sicurezza predefiniti [24].

2.4 Uso di bot nell'Ingegneria dei Requisiti

Per quanto riguarda lo sviluppo software, l'utilizzo degli agenti conversazionali è aumentato di molto, in quanto, vanno ad alleggerire molte attività ripetitive. A livello di team, i bot hanno migliorato molto la comunicazione e l'efficienza in ogni stato del ciclo di vita dello sviluppo software. Questo è stato possibile grazie a strumenti come Slack che permettono l'integrazione dei loro servizi con i bot. Con esso e altri strumenti è stato possibile integrare gli agenti conversazionali nelle chat di comunicazione. Questo ha permesso di migliorare l'interazione tra sviluppatori e bot. In particolare essi possono essere impiegati nell'Ingegneria dei Requisiti e soprattutto per la raccolta di questi ultimi. Proprio la raccolta dei requisiti è probabilmente la fase più importante dell'Ingegneria dei Requisiti, se non di tutto il processo di sviluppo software. A questo proposito è bene menzionare il lavoro fatto da Rajender et al.[25], i quali hanno proposto un approccio automatizzato per raccogliere requisiti di sistema utilizzando un chatbot. I requisiti raccolti sono classificati in requisiti funzionali e non funzionali attraverso un algoritmo. Il chatbot utilizza diversi algoritmi di machine learning per capire la risposta dell'utente, intraprendere ulteriori azioni e rispondere di conseguenza.
La Figura 2.3 mostra un esempio di interazione tra l'utente e il chatbot: l'utente descrive le funzioni che il sistema dovrebbe avere e il chatbot estrae i requisiti attraverso l'interazione. Il chatbot è in grado di identificare chiaramente l'utente cosa vuole, sia essa una semplice conversazione o delle affermazioni importanti in cui viene descritto un requisito di sistema. Esso può catturare requisiti descritti in una singola frase o attraverso diverse domande, in base alla completezza data dalle risposte dell'utente.

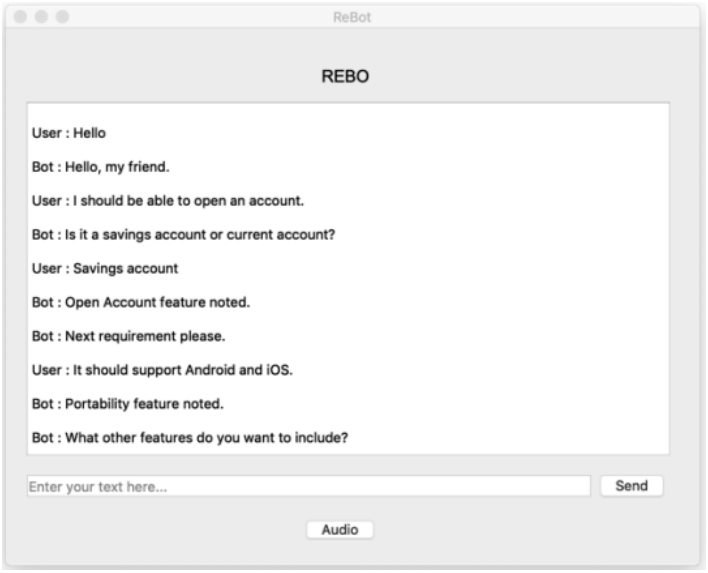

Figura 2.3: Interazione tra l'utente e il chatbot

Un lavoro molto più simile al nostro è quello compiuto da Dwitama e Rusli [26] i quali hanno sviluppato un chatbot Android che comunica con due tipi di utenti: l'utente stakeholder e lo sviluppatore. L'utente stakeholder comunica con il chatbot inviandogli i requisiti sotto forma di User Story mentre lo sviluppatore può visualizzare tutte le user stories sottomesse dall'utente, automatizzando così la fase di raccolta dei requisiti. Il bot utilizza l'Artificial Intelligence Markup Language per le sue conoscenze, grazie al quale riesce a rispondere all'utente. Il nostro lavoro invece separa gli ambienti nei quali l'utente e lo sviluppatore comunicano. Infatti, essendo utilizzate due piattaforme molto impiegate dai team di sviluppo software, come Trello e Slack, l'utente non avrà bisogno di scaricare un'app e avere un particolare sistema operativo per poter comunicare i suoi requisiti attraverso user stories. Inoltre la differenza più marcata è il fatto che il chatbot, nel nostro caso, è stato sviluppato per identificare contenuti di privacy all'interno delle US caricate dall'utente e quindi non viene utilizzato direttamente per la raccolta dei requisiti.

CAPITOLO 3

Approccio per l'individuazione di contenuti di privacy

In questo capitolo si parlerà del metodo utilizzato per identificare divulgazioni di privacy in User Stories.
Essa sfrutta una rete neurale convoluzionale per identificare contenuti di privacy all'interno di User Stories, sfruttando prima di tutto un approccio basato sul lessico che è utile per identificare le parole di privacy, confrontando ogni parola all'interno della User Story con un dizionario di privacy. Poiché questo approccio non tiene conto del contesto nel quale le parole sono usate, non tutte quelle di privacy potrebbero essere individuate. Per migliorare questo aspetto, viene utilizzato il Natural Language Processing, che comprende un insieme di algoritmi di intelligenza artificiale, in grado di analizzare, rappresentare e quindi comprendere il linguaggio naturale e di conseguenza il contesto nel quale le parole vengono utilizzate.
Il modello di rete neurale convoluzionale proposto è formato da più canali per eseguire efficientemente l'attività di classificazione in privacy o non privacy. Ogni canale fa riferimento a una diversa rappresentazione della stessa User Story. Viene inoltre fatto uso del transfer learning con il quale le conoscenze acquisite per la risoluzione di una task vengono riutilizzate per risolverne un'altra diversa ma correlata.

3.1 Natural Language Processing

L'insieme di tutte le tecniche e algoritmi usati per elaborare il linguaggio naturale viene anche chiamato Natural Language Processing (NLP).
La maggior parte delle persone ha probabilmente interagito con l'NLP senza rendersene conto. Ad esempio, l'NLP è la tecnologia di base dietro gli assistenti virtuali, come Siri o Alexa. Quando poniamo domande a questi assistenti virtuali, l'NLP è ciò che consente loro non solo di comprendere la richiesta dell'utente, ma anche di rispondere in linguaggio naturale. L'NLP si applica sia al testo scritto che al discorso e può essere applicata a tutte le lingue dell'uomo.
Essa comprende diverse task che, semplificate, possono essere:

- **Analisi lessicale:** scomposizione di un'espressione linguistica in token (in questo caso le parole).
- **Analisi grammaticale:** associazione delle parti del discorso a ciascuna parola nel testo.
- **Analisi sintattica:** arrangiamento dei token in una struttura sintattica (ad albero: parse tree).
- **Analisi semantica:** assegnazione di un significato (semantica) alla struttura sintattica e, di conseguenza, all'espressione linguistica.

Nel modello proposto la prima fase è quella di estrarre dalla User Story le parole di privacy utilizzando semplicemente un'analisi lessicale che fa uso di dizionari di privacy, attraverso il quale si possono contare quante volte viene utilizzata una determinata parola nel testo e classificare le parole in base alla sua categoria di appartenenza all'interno del dizionario. Di conseguenza ogni volta che viene trovata una parola e viene assegnata ad una determinata categoria, vengono anche incrementate le categorie rilevanti alle quali le parole appartengono. Ciò che viene fuori da questa fase sono dei valori per ogni categoria di privacy, rappresentati come percentuale delle parole totali nel testo. A seguire, un esempio di utilizzo del dizionario di privacy:

As a project manager , I want to access to data about my colleagues progress, so I can better report our success and failures.

Category name (# of words)	Description	Dictionary words found
OpenVisible(2)	open and public access to people	access, report
PrivateSecret(1)	the 'content' of privacy, i.e., what is considered private	data

Tabella 3.1: Categorie di privacy individuate.

La fase successiva, nella quale si è utilizzato l'NLP spaCy toolkit, è composta innanzitutto dalla *tokenizzazione*, che è un processo che suddivide le parole del testo in sequenze di token. In particolare viene rimossa la punteggiatura ed eliminate le parole insignificanti, lasciando solo i termini lessicali.
Di seguito un esempio di come questa tecnica viene effettuata:

Input: As a project manager , I want to access to data about my colleagues progress , so I can better report our success and failures.

Output: ["As", "a" , "project" , "manager" " , "I" , "want" , "to" , "access" , "to" , "data" , "about" , "my" , "colleagues" , "progress" , "so" , "I" , "can" , "better" , "report" , "our" , "success" , "and" , "failures"]

Il Dependency Parser Toolkit di spaCy è stato utilizzato per analizzare la struttura grammaticale del testo e identificare le dipendenze tra le parole all'interno della US. Questo ha permesso di capire la funzione sintattica di ogni parola all'interno di una frase e a identificare il soggetto, l'oggetto e gli altri elementi del testo.
Per identificare ed etichettare le parti del discorso (POS - part of speech) si è usufruito del POS Tagger di spaCy. Le parti del discorso non sono altro che categorie grammaticali come pronomi, nomi, verbi che descrivono il ruolo che una parola svolge in una frase.
Il Named Entity Recognition (NER) di spaCy è stato impiegato per riconoscere e classificare, con categorie predefinite, le entità presenti nel testo come persone, organizzazioni, luoghi, date e altro.

Text	Part of Speech	Dependency
As	ADP	prep
a	DET	det
project	NOUN	compound
manager	NOUN	pobj
I	PRON	nsubj
want	VERB	ROOT
to	PART	aux
access	VERB	xcomp
to	ADP	prep
data	NOUN	pobj
about	ADP	prep
my	PRON	poss
colleagues	NOUN	pobj
progress	VERB	dobj
so	SCONJ	mark
I	PRON	nsubj
can	AUX	aux
better	ADV	advmod
report	VERB	advcl
our	PRON	poss
success	NOUN	dobj
and	CCONJ	cc
failures	NOUN	conj

Tabella 3.2: Parti del discorso e dipendenze estratte dalla User Story.

3.2 Modello di Rete Neurale e Transfer Learning

Dopo queste fasi di pre-elaborazione, per far sì che il modello impari a conoscere le caratteristiche e gli schemi nascosti, i dati ottenuti vengono inseriti in una rete

neurale convoluzionale multi-input per identificare testi inerenti a concetti di privacy. In particolare, sono state utilizzate due reti neurali convoluzionali, una che si basa sulle caratteristiche NLP e un'altra su quelle basate sul lessico. La rete neurale che si basa sull'NLP ha come input le caratteristiche sintattiche (i.e. dependecies, POS e Entities) mentre l'altra rete neurale ha come input le caratteristiche lessicali (parole di privacy e categorie). L'output di queste reti sono due vettori che vengono uniti tra loro formando così i dati presenti nel layer Flatten. In seguito i due layer vengono uniti e dati in input al layer contenente i perceptron completamente collegati tra loro. Nella parte finale della rete neurale un singolo neurone viene utilizzato per distinguere tra rilevazione di privacy e non privacy.
Come visibile dalla Figura 3.1, per questo progetto di tesi è stato sfruttato un modello di rete neurale che utilizza il *Transfer Learning* per la classificazione in divulgazione o non divulgazione di privacy. Questo perché le reti neurali descritte richiedono un insieme di dati specifico per addestrare il modello a capire se si tratta di privacy o no, ma poiché i dataset di User Stories disponibili sono formati da poche centinaia di esempi, è possibile che i modelli non riescano ad addestrarsi in maniera adeguata per classificare una US.
Il Transfer Learning è un approccio nel quale le conoscenze apprese da un dataset per risolvere una task particolare, vengono riutilizzate per risolvere una task differente ma correlata. Grazie ad esso, specificatamente, è possibile utilizzare dei modelli di apprendimento pre-addestrati sfruttando un set di dati relativamente piccolo.
In questo modello, le caratteristiche basate sull'NLP descritte nella sezione precedente, sono elaborate da una rete neurale convoluzionale pre-addestrata, il cui scopo è di identificare se un testo non strutturato contiene privacy, analizzando la struttura semantica e sintattica del testo, ottenendo le caratteristiche risultanti dell'NLP come entità, dipendenze e POS.
Il modello da noi utilizzato è stato addestrato per il rilevamento di informazioni di privacy utilizzando diecimila post e commenti degli utenti di Reddit. [27]

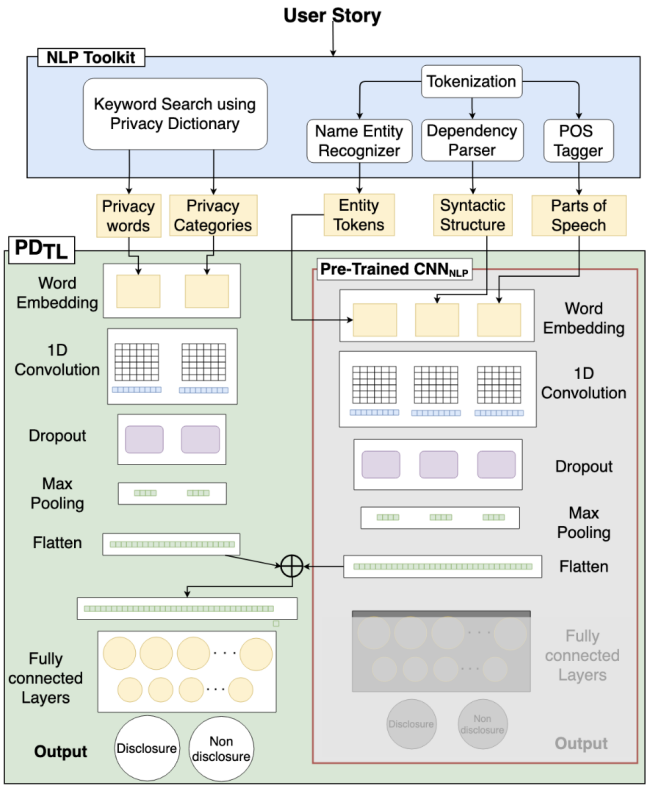

Figura 3.1: L'elaborazione della US in dettaglio

3.3 Validazione della metodologia

Il modello utilizzato per l'individuazione di privacy nelle User Stories ha bisogno di un insieme di USs per apprendere ed effettuare predizioni. Ogni User Story dovrebbe essere "etichettata" in modo tale da effettuare una classificazione che indichi se essa contenga informazioni di privacy. In letteratura però, non è presente un dataset di questo tipo ed è per questo che sono stati identificati 22 dataset, ognuno contenente più di 50 User Stories.[28] [29]. Questi dataset sono stati analizzati per verificare se fossero abbastanza eterogenei, ad esempio per vedere se includessero diversi tipi di User Stories. In particolare, i tipi identificati sono: USs che contengono parole di privacy ed effettuano divulgazione, USs che contengono solo parole di privacy, USs che contengono solo divulgazioni e USs semplici, cioè che non contengono né parole di privacy né effettuano divulgazione. La Figura 3.2 mostra le percentuali trovate di ogni tipologia diversa di User Story.

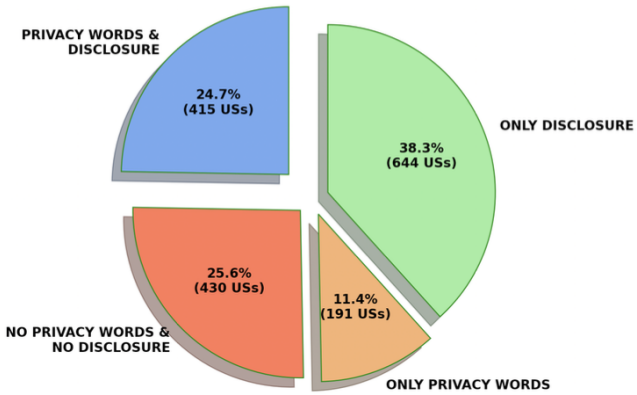

Figura 3.2: Tipologie di User Stories nel dataset

3.3.1 Criteri di valutazione

Per valutare l'accuratezza delle predizioni, abbiamo utilizzato 4 metriche di valutazione [30] - Accuratezza, Precisione, Recupero e F1-*score*. L'accuratezza è la parte della previsione che il nostro modello ha fatto in modo corretto. Formalmente, l'accuratezza ha la seguente definizione:

$$Accuratezza = \frac{\text{Predizioni corrette}}{\text{Numero totale di predizioni}}$$

Può essere calcolata anche in termini di positivi e negativi:

$$Accuratezza = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

Dove TP=*True Positive*, TN=*True Negative*, FP=*False Positive*, *FN=False Negative*.
La precisione è calcolata come:

$$Precisione = \frac{\text{TP}}{\text{TP+FP}}$$

ed indica la correttezza della previsione, in particolare cerca di capire quali delle previsioni positive sono effettivamente corrette.
Il recupero misura la completezza della risposta ed è calcolato come:

$$Recupero = \frac{\text{TP}}{\text{TP+FN}}$$

Dalla Figura 3.3 si può notare come la precisione indichi quante delle previsioni effettuate sono corrette, mentre il recupero indica quanti *true positive* sono stati effettivamente trovati.

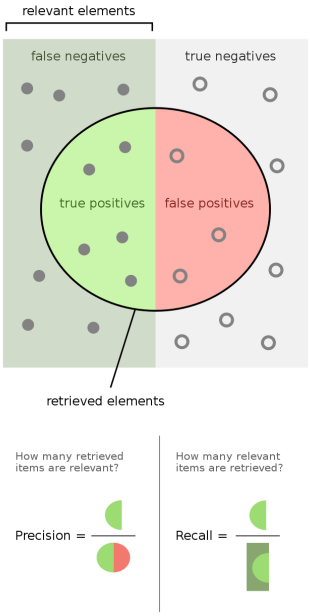

Figura 3.3: Differenza tra precisione e recupero

F1-score è, invece, definito come la media armonica tra la precisione e il recupero. Questi criteri sono stati scelti per provare a capire sia quanto i modelli fossero precisi nell'identificazione della privacy e sia quali fossero i loro limiti nella previsione.

3.3.2 Metodo di Validazione

Per determinare il grado di accuratezza del modello utilizzato abbiamo applicato una cross-validation o convalida incrociata. Essa viene detta k-fold e consiste nella suddivisione dell'insieme di dati totale in k parti di uguale numerosità e, a ogni passo, la k^a parte dell'insieme di dati viene a essere quella di convalida, mentre la restante parte costituisce sempre l'insieme di addestramento. In altre parole, si suddivide il campione osservato in gruppi di uguale numerosità, si esclude iterativamente un gruppo alla volta e si cerca di predirlo coi gruppi non esclusi, al fine di verificare la bontà del modello di predizione utilizzato.
Nel nostro caso, il dataset originale è stato diviso per k=5 ed è stato eseguita la convalida con questo k per 40 volte. Gli insiemi definiti per l'addestramento consistevano di 664 istanze, dove il 50% sono User Stories contenenti sia parole di privacy che divulgazioni, mentre il restante è stato diviso negli altri tre tipi. L'insieme di test, invece, consisteva di 166 istanze, dove 83 erano User Stories

contenenti sia divulgazioni che parole di privacy.

Model	Accuracy	F1-Score
CNN_{NLP}	0.720	0.713
CNN_{PW}	0.805	0.823
PD_{TL}	0.937	0.937

Tabella 3.3: Risultati ottenuti con ogni modello in termini di accuratezza e F1-Score.

Come si può notare dalla tabella, l'utilizzo del modello ottenuto dall'applicazione del Transfer Learning (PD_{TL}) ha fornito valori di F1-Score e di accuratezza migliori di quelli ottenuti con i modelli basati sul deep learning analizzati in precedenza, dove CNN_{NLP} è il modello che utilizza le tecniche basate sull'NLP mentre CNN_{PW} utilizza quelle basate sul lessico. In particolare, PD_{TL} migliora i risultati di CNN_{NLP} e CNN_{PW} di più del 10%.
Quindi, in base a questi risultati si può tranquillamente dire che utilizzare il Transfer Learning è meglio dell'utilizzare i modelli di deep learning per l'analisi dei contenuti di privacy.

CAPITOLO 4

Sviluppo di CAREFREE

Questo capitolo parlerà dello sviluppo del progetto CAREFREE, il quale consiste nello sviluppo di un bot tramite l'API di Slack. Esso è pensato per aiutare lo sviluppo software: l'utente stakeholder tramite un Task Management Software può caricare un requisito sotto forma di User Story ed attraverso l'integrazione del TMS con Slack, il bot viene notificato della creazione di una User Story. In maniera del tutto automatizzata effettua un controllo di privacy sul suo contenuto ed invia, tramite un messaggio in Slack i risultati allo sviluppatore. Il messaggio del bot avrà sia la User Story caricata e sia un'indicazione sul contenuto di privacy della US. Se sono stati individuati problemi di privacy, tramite un pulsante lo sviluppatore potrà mostrare o nascondere le parole di privacy individuate, con la relativa categoria e descrizione della stessa.

4.1 Tecnologie utilizzate

Lo sviluppo di CAREFREE ha implicato l'utilizzo di varie tecnologie.

- **Linguaggio di programmazione:** come linguaggio di programmazione è stato scelto *Python*, poiché grazie alle sue numerose librerie e integrazioni con i framework, risulta essere un linguaggio semplice, potente e che permette di ottimizzare lo sviluppo.
- **Framework:** per lo sviluppo del bot è stato utilizzato un framework che permettesse di facilitare lo sviluppo stesso.
  - **Flask:** è un popolare framework Python per lo sviluppo di applicazioni web. Viene fornito con funzionalità e requisiti minimi incorporati, rendendo semplice l'avvio e flessibile l'utilizzo.
  - **Slack API:** oltre a Flask, che ci è servito per far sì che la nostra applicazione, nella quale è stato sviluppato il bot, potesse essere una web app, abbiamo utilizzato Slack API proprio poiché il nostro bot deve comunicare in Slack. Questa API ci ha permesso di sviluppare il bot direttamente in Slack, attraverso le librerie slack e slackeventsapi.
- **ngrok:** è un software che serve a testare facilmente il bot in locale. In particolare, permette di creare un indirizzo web che punta al localhost. Questo significa che, esso permette di creare una connessione sicura con il nostro computer, cosi da poter eseguire dei test sui siti Web e sulle Web application che operano localmente, in modo tale da evitare di dover caricare il proprio codice su dei servizi cloud.

4.2 Architettura

In questa sezione si parlerà di come funziona effettivamente il bot e della sua architettura nei dettagli. Come si può notare dalla Figura 4.1 il processo parte dallo stakeholder, il quale utilizzando un qualsiasi TMS, e quindi attraverso un qualsiasi device, carica la User Story sulla piattaforma. Il TMS notifica Slack. A sua volta

quest'ultimo instaura una connessione HTTPS con il server, all'interno del quale viene eseguito il codice del bot. Il bot, effettua un'analisi del contenuto e restituisce a Slack, tramite il server, i risultati ottenuti. Questi ultimi possono essere visti all'interno di un canale apposito sia dallo sviluppatore che dall'utente stakeholder, o comunque da chiunque sia interessato alla User Story definita sul TMS.

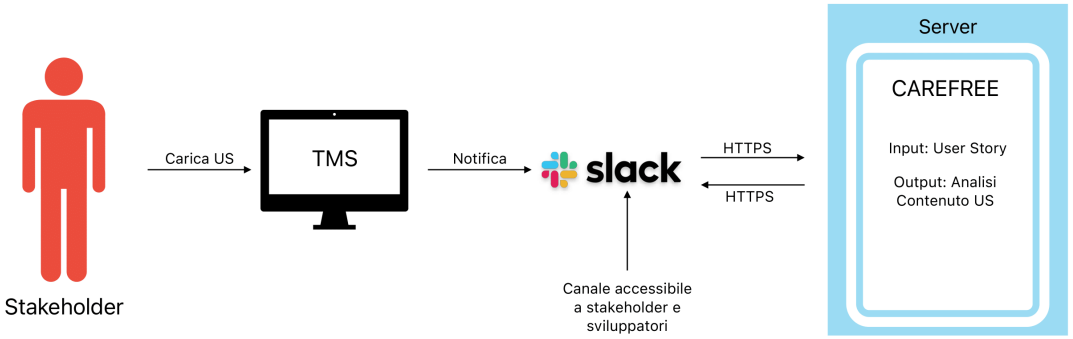

Figura 4.1: Architettura del bot CAREFREE

4.3 Implementazione

In questa sezione viene descritta l'implementazione vera e propria del progetto. Viene descritto principalmente cos'è Slack e gli strumenti che mette a disposizione per sviluppare un bot.

4.3.1 Slack

Slack è un software che rientra nella categoria degli strumenti di collaborazione aziendale utilizzato per inviare messaggi in modo istantaneo ai membri del team. Una delle funzioni di Slack è la possibilità di organizzare la comunicazione del team attraverso canali specifici che possono essere accessibili a tutto il team o solo ad alcuni membri.
Grazie all'integrazione con diverse applicazioni, tra cui alcuni TMS come Trello, è possibile aumentare le prestazioni del software e la produttività del team. Inoltre mette a disposizione una propria API per gli sviluppatori che può essere utilizzata per creare la propria app in Slack e sviluppare un bot.

Quindi per poter creare un bot in Slack si deve innanzitutto creare la propria app utilizzando la Slack API, dopodiché bisogna aggiungere le finalità del bot come ad esempio "chat:write" che permette al bot di inviare un messaggio in un canale. Infine viene generato un token che dovrà essere inserito nel codice per permettere il collegamento tra codice e chatbot.

4.3.2 Gestire gli eventi e le interazioni

```
@slack_event_adapter.on('message')
def message(payload):
    print(payload)
    event = payload.get('event', {})
    channel_id = event.get('channel')

    if event.get('bot_id') == os.environ['TRELLO_ID']:
        global user_story
        user_story = event.get('attachments')[0]['title']
... ... ...
```

Per permettere al bot di gestire gli eventi in Slack abbiamo utilizzato la libreria *slackeventsapi*.
Utilizzando *@slack_event_adapter.on('message')* è possibile gestire un messaggio in entrata: viene definita in questo modo una funzione listener che viene attivata non appena viene ricevuto l'evento *message* dall'adapter.
Nel nostro caso, poiché il messaggio che a noi interessa è quello proveniente da Trello che ci avvisa della creazione di una User Story, nel codice si va a controllare prima se si tratta proprio di Trello, se è cosi, si ottiene dal payload la User Story e viene richiamata la funzione ***prediction()***, passando come parametro proprio la US.

Questa funzione permette di effettuare l'analisi sulla US ed è quindi il punto cardine di CAREFREE.
Per inviare il messaggio con tutte le informazioni relative ai risultati ottenuti dall'analisi viene utilizzato il metodo chat_postMessage().
Di seguito viene mostrata una parte della funzione con cui vengono gestite le interazioni ed il modo in cui vengono mostrati i dettagli dell'elaborazione della US.

```
@app.route('/slack/actions', methods=['POST'])
def handle_action():
    data = json.loads(request.form["payload"])
    action = data.get("actions")[0].get("action_id")
    if action == "action-show":
        blocks = [{"type": "section",
                   "text": {"type": "plain_text",
... ... ...
```

Per costruire un messaggio interattivo e personalizzato si è utilizzato Block Kit Builder che è un framework di Slack per l'interfaccia utente, grazie al quale si è potuto inserire all'interno del messaggio un pulsante per mostrare o nascondere i dettagli dell'analisi. Quando si clicca sul pulsante viene mandata una richiesta HTTPS con metodo POST al path /slack/actions, di conseguenza l'interazione può essere gestita, grazie a Flask, tramite *@app.route('/slack/actions', methods=['POST'])*.
Inoltre, per mostrare i dettagli si è deciso di non inviare un nuovo messaggio, ma semplicemente viene aggiornato il precedente tramite il metodo chat_update().
Viene poi inviata la risposta alla richiesta HTTPS tramite il metodo Response().

4.3.3 Funzionamento

Fin'ora abbiamo parlato dell'architettura del bot e di come esso gestisce gli eventi e le interazioni, ma non abbiamo parlato passo per passo del modo in cui si raccolgono le informazioni sulla User Story. Innanzitutto lo stakeholder deve creare un account su un Task Management Software, cioè un software che permette di gestire il ciclo di vita di una task. Per l'esempio utilizziamo Trello; dopo aver creato un account, tramite la sua integrazione con Slack, deve collegarlo al canale nel quale verranno visualizzati i risultati. Una volta aver integrato Trello con Slack, lo stakeholder carica la User Story sotto forma di task, questo permette al bot di "attivarsi" poiché arriverà un messaggio in Slack che informa i membri dell'aggiunta di una US su Trello.

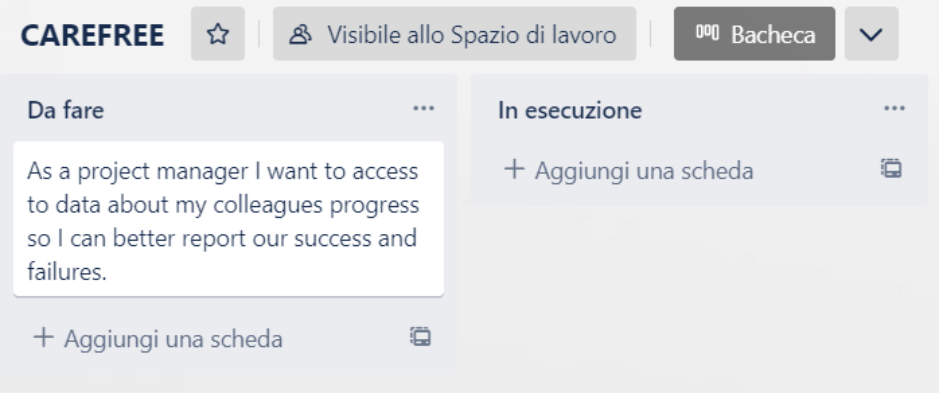

Figura 4.2: User Story in Trello

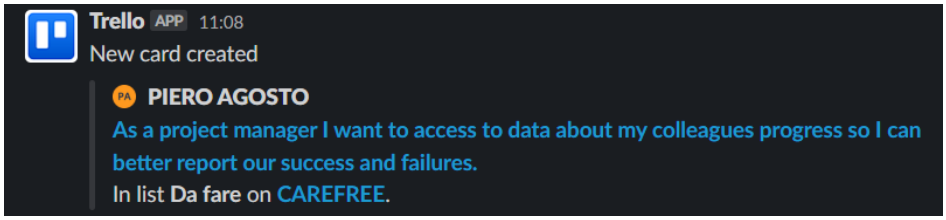

Figura 4.3: Messaggio da Trello in Slack

A questo punto il bot effettua l'analisi di cui si è parlato nelle sezioni precedenti e invia un messaggio in Slack contenente: la User Story aggiunta su Trello e se essa contiene privacy oppure no.
Nel caso in cui la User Story contenga privacy è presente un pulsante "Show Privacy Words" che permette di mostrare ai membri del canale le parole di privacy trovate, la categoria di privacy alla quale appartengono e una descrizione della categoria.

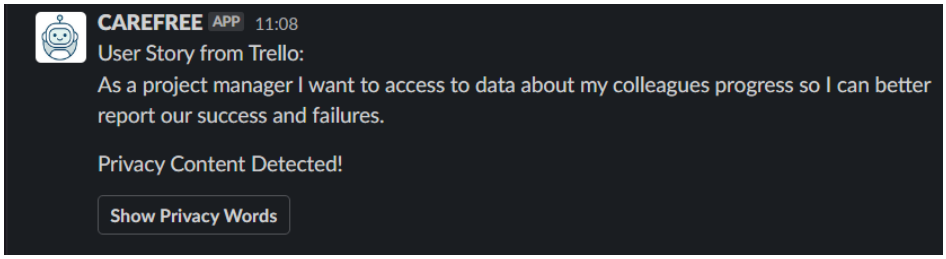

Figura 4.4: Messaggio del bot post elaborazione US

Cliccando sul pulsante:

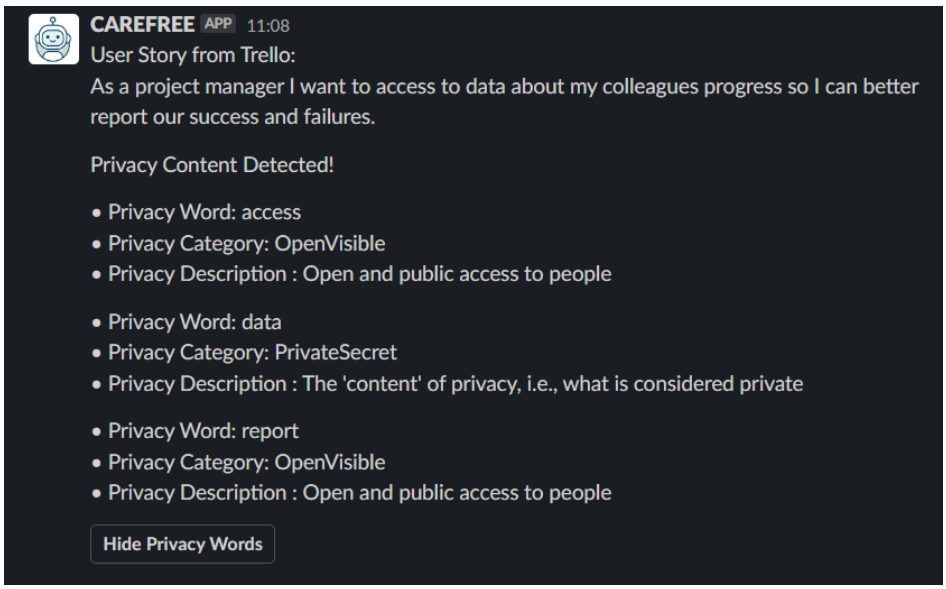

Figura 4.5: I risultati nel dettaglio

Per nascondere i dettagli è possibile utilizzare il pulsante "Hide Privacy Words".

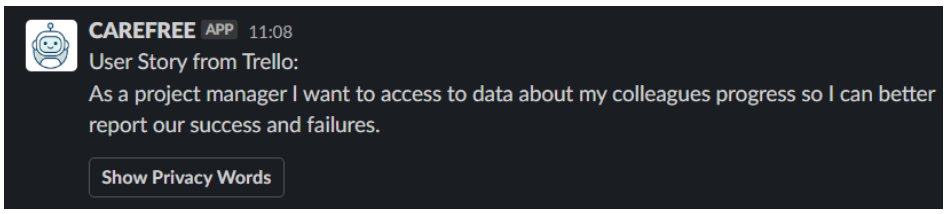

Figura 4.6: Dettagli Nascosti

CAPITOLO 5

Casi d'uso

In questo capitolo verranno esaminati vari casi d'uso di CAREFREE, mettendo in evidenza l'effettiva usabilità del chatbot. In particolare vengono analizzati 3 casi d'uso:

- User Story che **contiene privacy**;
- User Story che **non contiene privacy**;
- Diverse User Stories, **alcune contenenti privacy e altre non**, esaminate dal bot sequenzialmente;

5.1 User Story contenente privacy

La User Story che andiamo ad analizzare contiene privacy, quindi il bot dovrà essere in grado di identificarla.

> **User Story:** As a site admin, I want to delete any course or event, so that I can remove things that will no longer occur.

Come detto nel capitolo precedente, una volta caricata la User Story su un TMS, viene notificato Slack e CAREFREE avvia l'analisi, il cui risultato in questo caso è:

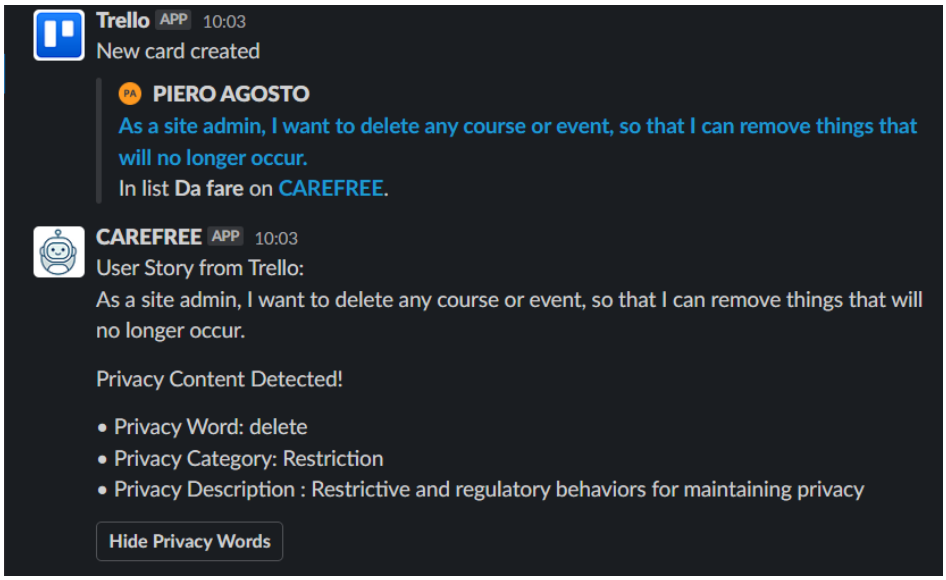

Figura 5.1: User Story con Privacy

Come si può notare dalla Figura 5.1 CAREFREE è stato effettivamente in grado di identificare contenuti di privacy ed in particolare ha identificato:

Privacy Word	Privacy Category	Privacy Description
delete	Restriction	Restrictive and regulatory behaviors for maintaining privacy

Tabella 5.1: Risultati ottenuti dal primo caso d'uso.

L'invio della notifica per ogni User Story può essere utile per evitare che lo sviluppatore effettui manualmente sia la raccolta dei requisiti e sia l'identificazione dei contenuti di privacy. Inoltre la visualizzazione della categoria di privacy e della relativa descrizione possono essere utili per diversi motivi:

- **Comprensione:** possono aiutare lo sviluppatore o lo stakeholder a comprendere meglio i concetti di privacy contenuti nella User Story, evitando confusione o interpretazioni errate;
- **Comunicazione:** possono rendere più semplice la comunicazione tra i vari stakeholder, come sviluppatori e ad esempio un legale aziendale, responsabile

della privacy dei clienti e dei dipendenti dell'azienda. Di conseguenza queste informazioni aiutano a garantire il rispetto delle norme e delle leggi relative alla privacy;

- **Implementazione:** possono permettere allo sviluppatore di prendere una decisione relativa all'implementazione, riguardante la privacy, in maniera più semplice;

5.2 User Story non contenente privacy

Vediamo ora il caso nel quale la User Story non contiene privacy.

> **User Story:** As a user, I want to be able to view a map display of the special waste drop off sites around my area.

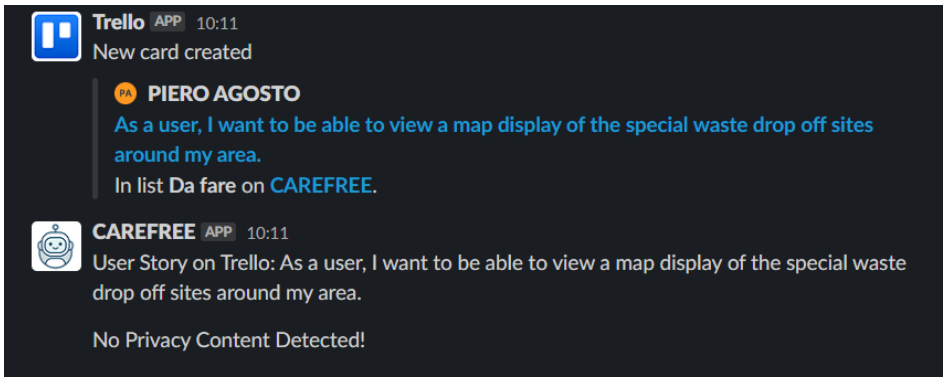

Figura 5.2: User Story senza Privacy

La Figura 5.2 mostra che CAREFREE ha elaborato la User Story ed ha concluso correttamente, che non contiene privacy, notificando i membri del canale su Slack.

5.3 User Stories analizzate sequenzialmente

In questa sezione, esaminiamo il caso in cui si richiede al bot di analizzare in maniera sequenziale più User Stories. In particolare, in questo caso CAREFREE analizza 2 User Story contenenti privacy e altre 2 senza contenuti di privacy.

US with privacy	US without privacy
As a Plan Review Staff member, I want to Assign Plans for Review, so that I can ensure plans have been assigned to the appropriate Plan Reviewer.	As an agency user, I want to know when the submission periods start and end, so that I know when the submission starts and ends.
As a Staff member, I want to Develop a Staff Report, so that I can publish and distribute the report according to business rules	As a camp administrator, I want to be able to set a reminders related to specific tasks that has to be done.

Tabella 5.2: User Stories analizzate consecutivamente.

Ecco come vengono visualizzate le User Stories definite in Trello:

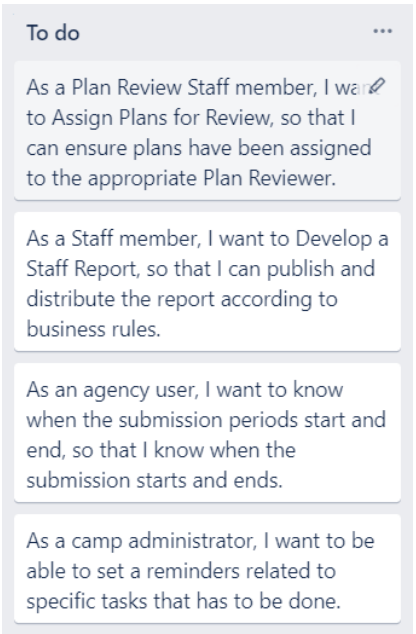

Figura 5.3: User Stories in Trello

In Figura 5.4 vengono mostrate le predizioni visualizzate in Slack.

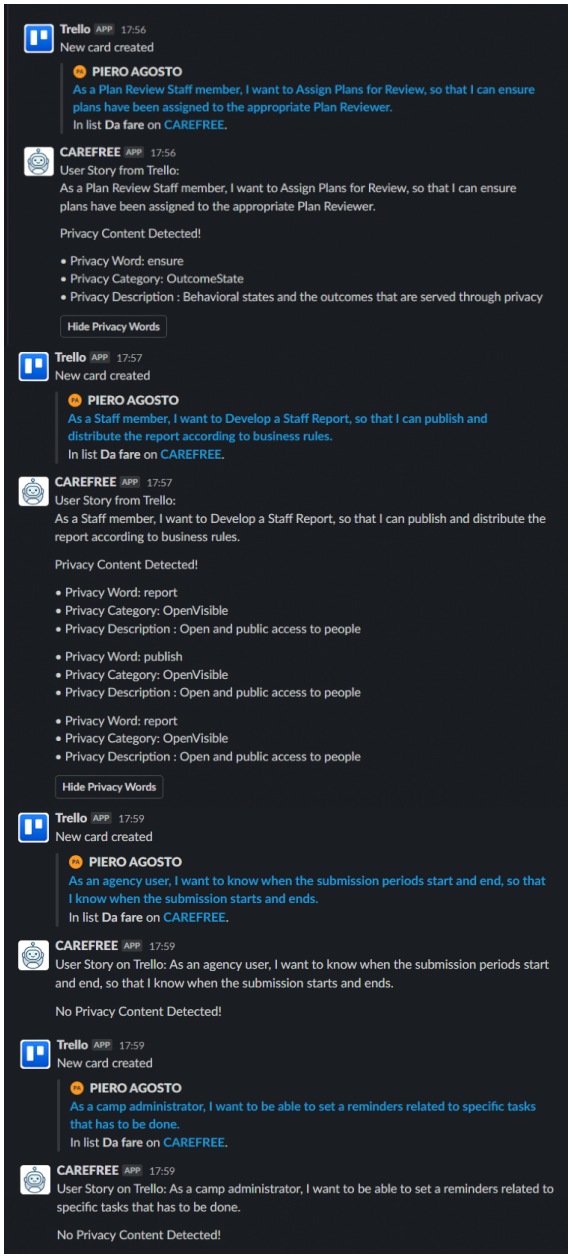

Figura 5.4: Risultati dell'analisi delle USs

Concludendo, si può dire che CAREFREE riesce effettivamente a elaborare più User Stories consecutivamente, ma la cosa da sottolineare è la possibilità di ricevere le predizioni in Slack, il che rende la comunicazione tra stakeholders semplice e rapida, migliorando il processo di sviluppo. Inoltre, la correlazione di ogni parola di privacy con la categoria di privacy e la relativa descrizione è molto utile, poiché aiuta a comprendere meglio il significato e l'utilizzo di ogni categoria di privacy, aiutando a rispettare le leggi sulla privacy. Questo rende il processo di sviluppo trasparente e garantisce che i dati personali vengano gestiti in modo responsabile e sicuro.

CAPITOLO 6

Conclusioni

Come detto nell'introduzione, la privacy sta diventando un argomento sempre più importante e come tale essa deve essere affrontata quanto prima nello sviluppo di un software. Per questo aspetto l'utilizzo di CAREFREE potrebbe rilevarsi importante. Inoltre, durante il suo sviluppo ho compreso quanto sia fondamentale potersi avvalere di uno strumento del genere nell'Ingegneria dei Requisiti. Esso infatti permette facilmente di individuare divulgazioni di privacy, permettendo così di evitare che informazioni così importanti possano essere rese pubbliche ed inoltre permette di raccogliere indirettamente anche i requisiti di privacy e non, riuscendo, in pratica, a non tralasciare nessun dettaglio.
Il punto cardine di questa tesi è stato sicuramente l'utilizzo delle tecniche del machine learning basate sull'NLP e del Transfer Learning: quest'ultimo particolarmente è stato molto utile ai fini dell'identificazione della privacy, poiché il suo utilizzo permette, innanzitutto di ridurre i costi computazionali, in quanto parte del modello è già stato addestrato su un compito simile e quindi richiede meno dati per l'addestramento; può migliorare, inoltre, sia il riconoscimento del linguaggio naturale e sia la precisione della classificazione dei contenuti in privacy o non privacy.
Utilizzando queste tecniche di machine learning, ho appreso quanto sia interessante

e importante questo argomento che con ogni probabilità sarà sempre più presente nel nostro futuro. Il suo utilizzo infatti permette diversi vantaggi, tra cui:

- *molti compiti ripetitivi come quelli che si affrontano nell'Ingegneria dei Requisiti, possono essere automatizzati;*
- *i modelli di machine learning possono migliorarsi continuamente, con il risultato che le prestazioni fornite saranno sempre migliori col tempo.*

6.1 Sviluppi futuri

Per quanto riguarda gli sviluppi futuri ci sarebbero diverse cose che si potrebbero apportare a CAREFREE. Prima fra tutte, l'aggiunta dei Diagrammi di Flusso dei Dati (DFD), che permettono di rappresentare visivamente come i dati fluiscono attraverso un processo o un sistema; in particolare essi mappano la sequenza di informazioni, attori e fasi di un processo. Quindi i DFD, servono nel caso specifico, per rappresentare la logica del flusso dei dati all'interno del processo, permettendo alle persone non tecniche coinvolte in un progetto di capire più facilmente come i dati in ingresso, diventano i dati in uscita.
Poi si potrebbero migliorare le tecniche di machine learning utilizzate ed integrare il bot con altri strumenti di upload della User Story, come Jira e Asana.
Inoltre, si potrebbe rendere più efficiente il bot quando si trova ad operare con più USs consecutivamente ed in particolare migliorare la visualizzazione delle predizioni quando ci si trova in questa situazione. Ulteriori sviluppi potrebbero riguardare l'aggiunta di altri requisiti non funzionali, oltre alla privacy e l'utilizzo di altre tecniche di NLP per rendere più precise le predizioni.

Ringraziamenti

A conclusione di questo lavoro di tesi, desidero menzionare tutte le persone che mi hanno aiutato prima e durante tutto il percorso universitario.

Un ringraziamento particolare va al Prof. Fabio Palomba, che mi ha seguito durante tutto il lavoro di tesi, dimostrandosi competente e disponibile. Grazie ai suoi consigli ho potuto migliorare la tesi in diversi aspetti e accrescere le mie conoscenze.

Ringrazio il Dott. Francesco Casillo, persona gentilissima, disponibile e appassionata, sempre pronto ad aiutarmi e motivarmi.

Ringrazio infinitamente i miei genitori, senza di loro non potrei essere chi sono e dove sono oggi. Siete la mia forza, Vi Amo.

Ringrazio i miei nonni, grazie per tutti i sacrifici e i consigli che mi avete dato. Saranno sempre con me.

Ringrazio la mia fidanzata, Benedetta, che mi ha trasmesso ogni giorno la sua forza e il suo coraggio. Grazie perché ci sei sempre stata, nei momenti più belli e soprattutto in quelli più bui. Mi hai motivato ad andare avanti e a te devo tanto, Ti Amo.

Infine, ringrazio tutti i miei amici per i momenti di divertimento e spensieratezza assieme.

Bibliografia

[1] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2004. (Citato a pagina 8)

[2] X. Wang, L. Zhao, Y. Wang, and J. Sun, "The role of requirements engineering practices in agile development: An empirical study," in *Requirements Engineering*, 2014, pp. 195–209. (Citato a pagina 8)

[3] S. Dimitrijević, J. Jovanović, and V. Devedzic, "A comparative study of software tools for user story management," *Inf. Softw. Technol.*, vol. 57, pp. 352–368, 2015. (Citato a pagina 9)

[4] M. Cohn, *User Stories Applied: For Agile Software Development*. USA: Addison Wesley Longman Publishing Co., Inc., 2004. (Citato a pagina 9)

[5] S. Jiménez and R. Juàrez-Ramirez, "A quality framework for evaluating grammatical structure of user stories to improve external quality," in *Proceedings of 7th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 2019, pp. 147–153. (Citato a pagina 9)

[6] G. Lucassen, F. Dalpiaz, M. van der Werf, and S. Brinkkemper, "Forging high-quality user stories: Towards a discipline for agile requirements," in *Proceedings of IEEE 23rd International Requirements Engineering Conference (RE)*, 2015, pp. 126–135. (Citato a pagina 9)

[7] P. Heck and A. Zaidman, "A quality framework for agile requirements: A practitioner's perspective," *ArXiv*, vol. abs/1406.4692, 2014. (Citato a pagina 9)

[8] W. B. A. Karaa, Z. B. Azzouz, A. Singh, N. Dey, A. S. Ashour, and H. H. B. Ghézala, "Automatic builder of class diagram (abcd): an application of uml generation from functional requirements," *Software: Practice and Experience*, vol. 46, pp. 1443 – 1458, 2016. (Citato a pagina 9)

[9] M. Elallaoui, K. Nafil, and R. Touahni, "Automatic transformation of user stories into uml use case diagrams using nlp techniques," in *The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018)*, 2018, pp. 42–49. (Citato a pagina 9)

[10] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Extracting conceptual models from user stories with visual narrator," *Requirements Engineering*, vol. 22, pp. 339–358, 2017. (Citato a pagina 9)

[11] F. Gilson and C. Irwin, "From user stories to use case scenarios towards a generative approach," in *Proceedings of 25th Australasian Software Engineering Conference (ASWEC)*, 2018, pp. 61–65. (Citato a pagina 9)

[12] F. Gilson, M. Galster, and G. F., "Extracting quality attributes from user stories for early architecture decision making," in *Proceedings of IEEE International Conference on Software Architecture Companion(ICSA-C)*, 2019, pp. 129–136. (Citato a pagina 9)

[13] K. Barker, M. Askari, M. Banerjee, K. Ghazinour, B. Mackas, M. Majedi, S. Pun, and A. Williams, "A data privacy taxonomy," in *Proceedings of the 26th British National Conference on Databases: Dataspace: The Final Frontier*, 2009, pp. 42–54. (Citato a pagina 10)

[14] S. De Capitani Di Vimercati, S. Foresti, G. Livraga, and P. Samarati, "Data privacy: Definitions and techniques," *International Journal of Uncertainty*, pp. 793–817, 2012. (Citato a pagina 10)

[15] A. Gill, A. Vasalou, C. Papoutsi, and A. Joinson, "Privacy dictionary: A linguistic taxonomy of privacy for content analysis," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 3227–3236. (Citato a pagina 10)

[16] A. Vasalou, A. Gill, F. Mazanderani, C. Papoutsi, and A. Joinson, "Privacy dictionary: A new resource for the automated content analysis of privacy," *Journal of the American Society for Information Science and Technology (JASIST)*, pp. 2095–2105, 2011. (Citato a pagina 10)

[17] N. Mehdy, C. Kennington, and H. Mehrpouyan, "Privacy disclosures detection in natural-language text through linguistically-motivated artificial neural networks," in *Security and Privacy in New Computing Environments*, 2019, pp. 152–177. (Citato a pagina 10)

[18] G. Xu, C. Qi, H. Yu, S. Xu, C. Zhao, and J. Yuan, "Detecting sensitive information of unstructured text using convolutional neural network," in *Proceedings of International Conference on Cyber-Enabled Dis- tributed Computing and Knowledge Discovery (CyberC)*, 2019, pp. 474–479. (Citato a pagina 10)

[19] P. Silva, C. Goncalves, C. Godinho, N. Antunes, and M. Curado, "Using nlp and machine learning to detect data privacy violations," in *Proceedings of IEEE Conference on Computer Communications Workshops*, 2020, pp. 972–977. (Citato a pagina 10)

[20] W. Tesfay, J. Serna, and K. Rannenberg, "Privacybot: Detecting privacy sensitive information in unstructured texts," in *Proceedings of Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 2019, pp. 53–60. (Citato a pagina 10)

[21] J. Neerbeky, I. Assentz, and P. Dolog, "Taboo: Detecting unstructured sensitive information using recursive neural networks," in *Proceedings of IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017, pp. 1399–1400. (Citato a pagina 10)

[22] S. Miyazaki, N. Mead, and J. Zhan, "Computer-aided privacy requirements elicitation techniques," in *2008 IEEE Asia-Pacific Services Computing Conference*, 2008, pp. 367–372. (Citato a pagina 10)

[23] Y. Benbenisty, I. Hadar, G. Luria, and P. Spoletini, "Privacy as first-class requirements in software development: A socio-technical approach," in *36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 1363–1367. (Citato a pagina 10)

[24] M. Riaz, J. King, J. Slankas, and L. Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," in *Proceedings of IEEE 22nd International Requirements Engineering Conference (RE)*, 2014, pp. 183–192. (Citato a pagina 11)

[25] C. S. Rajender Kumar Surana, Shriya, D. B. Gupta, and S. P. Shankar, "Intelligent chatbot for requirements elicitation and classification," in *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, 2019, pp. 866–870. (Citato a pagina 11)

[26] F. Dwitama and A. Rusli, "User stories collection via interactive chatbot to support requirements gathering," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, pp. 890–898, 2020. (Citato a pagina 12)

[27] F. Casillo, V. Deufemia, and C. Gravino, "Detecting privacy requirements from user stories with nlp transfer learning models," *Information and Software Technology*, vol. 146, p. 106853, 2022. (Citato a pagina 17)

[28] F. Dalpiaz. (2018) Requirements data sets (user stories). [Online]. Available: https://data.mendeley.com/datasets/7zbk8zsd8y/ (Citato a pagina 18)

[29] F. Dalpiaz, I. Schalk, S. Brinkkemper, F. Aydemir, and G. Lucassen, "Detecting terminological ambiguity in user stories: Tool and experimentation," *Information and Software Technology*, vol. 110, 12 2018. (Citato a pagina 18)

[30] D. Powers, "Evaluation: From precision, recall and fmeasure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, pp. 37–63, 01 2007. (Citato a pagina 19)