UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea Triennale in Informatica

Generazione di User Story da Requisiti in Linguaggio Naturale: un caso di studio

Relatore:
Ch.mo Prof.
Carmine GRAVINO

Candidato:
Alfonso ANZELMO
Mat. 0512113154

Correlatore:
Ch.mo Prof.
Francesco CASILLO

ANNO ACCADEMICO 2024/2025

Questa tesi è stata sviluppata nel

A mio padre,

per il sostegno instancabile, la pazienza infinita e il coraggio che mi hai trasmesso.

Lista delle figure

Lista delle tabelle

Abstract

Le metodologie agili si sono affermate come essenziali per organizzazioni che cercano di adattarsi rapidamente alle mutevoli esigenze dei clienti e le richieste del mercato.
Nello sviluppo agile, un fondamentale artefatto sono le *user stories*, ovvero un formato conciso per esprimere i requisiti software dal punto di vista dell'utente. L'estrazione automatica delle user stories è necessaria, quindi, per organizzazione che adottano lo sviluppo agile.
Il beneficio principale dell'estrazione delle user stories è il supporto che può fornire all'analista nell'identificazione dei requisiti e delineamento del dominio di interesse del software.
In generale, le fonti per la raccolta dei requisiti sono disponibili in linguaggio naturale. Tuttavia, l'approccio all'estrazione delle user stories dal linguaggio naturale è ancora limitato, per via dell'ambiguità intrinseca del linguaggio naturale.
A questo scopo, la tesi si propone di fornire un approccio per estrarre le user stories in maniera automatica e nel modo più accurato e preciso possibile.
Tramite tecniche di NLP, ovvero *natural language processing*, forniremo un approccio in grado di estrarre le user stories dal linguaggio naturale in automatico, riducendo così l'impegno umano nella progettazione e sviluppo del software, che riesce quindi a minimizzare gli errori o ambiguità delle fasi principali e fondamentali del ciclo di sviluppo del software.

Indice

Capitolo 1

Introduzione

1.1 Contesto applicativo

Nello sviluppo software agile [2], la capacità di tradurre rapidamente i requisiti raccolti dagli stakeholder in user stories chiare e utilizzabili è fondamentale per mantenere alta la velocità e la qualità del ciclo di sviluppo.
Le user stories sono infatti strumenti chiave nelle metodologie agili, perché descrivono le funzionalità dal punto di vista dell'utente finale, facilitando la comprensione delle esigenze di business e la priorità delle funzionalità da sviluppare.
Tuttavia, molti requisiti vengono inizialmente raccolti in linguaggio naturale, presentando quindi una sfida di interpretazione per i team di sviluppo. Questi requisiti possono contenere informazioni implicite, ambiguità, o dettagli non necessari, rendendo difficile la loro conversione manuale in user stories strutturate e chiare.
Esempio Applicativo: Immaginiamo un sistema di gestione ordini in un ristorante che coinvolge vari attori come camerieri, cuochi e manager. Uno degli stakeholder descrive il seguente requisito:

Requisito: "Il cameriere deve poter visualizzare lo stato degli ordini per sapere se sono pronti da servire, mentre il cuoco deve essere informato sugli ordini in attesa." L'estrazione automatica permetterebbe di generare due user stories chiare:

1. **User Story per il cameriere**: *Come cameriere, voglio visualizzare lo stato degli ordini per sapere se sono pronti da servire.*

2. **User Story per il cuoco**: *Come cuoco, voglio essere informato sugli ordini in attesa.*

1.2 Motivazioni

L'estrazione automatica delle user stories da requisiti in linguaggio naturale risponde a una necessità crescente nel contesto dello sviluppo software agile. Le metodologie agili richiedono infatti che i team di sviluppo possano trasformare rapidamente i requisiti degli stakeholder in user stories comprensibili e implementabili, tuttavia, i requisiti scritti in linguaggio naturale spesso risultano ambigui, informali e poco strutturati. Questa ambiguità può portare a incomprensioni, interpretazioni errate e, di conseguenza, a errori nello sviluppo del prodotto.
Di seguito alcune delle motivazioni all'estrazione automatica di user stories:

- **Velocizzare il Passaggio dai Requisiti alle User Stories**: La trasformazione manuale dei requisiti in user stories è un processo lento e soggetto a errori. Automatizzare questa trasformazione consente di risparmiare tempo e risorse, riducendo al minimo l'intervento umano.

- **Ridurre le ambiguità e aumentare la qualità delle User Stories**: I requisiti in linguaggio naturale sono spesso scritti in modo informale, e questa informalità introduce ambiguità. Utilizzare un sistema automatico permette di garantire coerenza e qualità, riducendo il rischio di fraintendimenti che possono manifestarsi nelle fasi successive dello sviluppo.

- **Facilitare la Comunicazione tra Stakeholder e Team di Sviluppo**: Una volta estratte, le user stories risultano più comprensibili e standardizzate rispetto ai requisiti scritti in linguaggio naturale. Questo favorisce la comunicazione e l'allineamento tra stakeholder e sviluppatori, migliorando la chiarezza degli obiettivi di progetto.

1.3 Obiettivi

L'estrazione automatica di user stories mira a risolvere queste problematiche, permettendo di passare da requisiti in linguaggio naturale e informale a user stories formalizzate.
Utilizzando tecniche di *NLP*, ovvero Natural Language Processing, si propone un approccio che identifica automaticamente attori, azioni, scomponendo i requisiti in elementi strutturati.
Nello specifico questa tesi si propone di **definire un approccio di estrazione di user stories** cioè di creare un approccio che identifichi con precisione le componenti principali di una user story: attore, azione e motivazione.

1.4 Risultati ottenuti

Index	Requirement	User Story
0	A waiter should be able to use a tablet to take customer orders and send them directly to the kitchen staff for faster order processing.	[AS A waiter I WANT TO be able to use a tablet to take customer orders and send them directly to the kitchen staff for faster order processing SO THAT to take customer orders and send them directly to the kitchen staff for faster order processing]
1	The waiter must be able to view the restaurant's current menu on their tablet, including prices and availability of items.	[AS The waiter I WANT TO be able to view the restaurant's current menu on SO THAT to view the restaurant's current menu on their tablet, including prices and availability of items]
2	A waiter should be able to use a tablet to take customer orders and also modify existing orders before they are finalized to accommodate special requests.	[AS A waiter I WANT TO be able to use a tablet to take customer orders and also modify existing orders before they are finalized to accommodate special requests SO THAT to accommodate special requests]

Table 1.1: Estratto delle user stories ottenute

Più dataset sono stati impiegati dall'approccio, ottenendo in output le user stories di ognuno.
Inoltre, dalla tabella possiamo vedere che da un singolo requisito è possibile estrarre più user stories.
Ottenute le use stories è stata eseguita la fase di *validazione*. Eseguita confrontando le user stories generate dall'approccio con due oracoli: uno creato manualmente, e uno generato da ChatGPT.
La fase di *validazione* ha misurato la somiglianza tra le user stories generate dall'approccio e quelle ottenute dai due oracoli, e **BLEU**, **METEOR** e **ROUGE** sono state le metriche utilizzate.

#	**Oracolo**	**Rouge**	**Bleu**	**Meteor**
3	ChatGPT	• Rouge1: precision=0.74, recall=0.84, fmeasure=0.78 • Rouge2: precision=0.53, recall=0.60, fmeasure=0.56 • RougeL: precision=0.61, recall=0.69, fmeasure=0.65	0.5685	0.4517
3	Manuale	• Rouge1: precision=0.60, recall=0.89, fmeasure=0.72 • Rouge2: precision=0.51, recall=0.76, fmeasure=0.61 • RougeL: precision=0.54, recall=0.80, fmeasure=0.64	0.5133	0.5511
49	ChatGPT	• Rouge1: precision=0.79, recall=0.52, fmeasure=0.63 • Rouge2: precision=0.48, recall=0.31, fmeasure=0.38 • RougeL: precision=0.63, recall=0.41, fmeasure=0.50	0.3864	0.3513

#	Oracolo	Rouge	Bleu	Meteor
49	Manuale	• Rouge1: precision=0.97, recall=0.72, fmeasure=0.83 • Rouge2: precision=0.83, recall=0.61, fmeasure=0.70 • RougeL: precision=0.60, recall=0.66, fmeasure=0.76	0.6231	0.5862
34	ChatGPT	• Rouge1: precision=0.83, recall=0.51, fmeasure=0.63 • Rouge2: precision=0.45, recall=0.28, fmeasure=0.34 • RougeL: precision=0.60, recall=0.37, fmeasure=0.46	0.3768	0.3008
34	Manuale	• Rouge1: precision=0.83, recall=0.81, fmeasure=0.82 • Rouge2: precision=0.63, recall=0.62, fmeasure=0.62 • RougeL: precision=0.74, recall=0.72, fmeasure=0.73	0.6933	0.6071

Table 1.2: Valutazioni per documento e per tipo oracolo

Per tutti i documenti notiamo che i valori delle metriche sono tendenzialmente più alti per l'oracolo manuale rispetto a quelli generati da ChatGPT. Questo accade per diversi motivi: **Affinità Semantica e Lessicale**: L'oracolo manuale è stato costruito con un maggiore allineamento lessicale e strutturale rispetto ai requisiti di partenza, ottenendo quindi punteggi BLEU,

METEOR e ROUGE più elevati. ChatGPT ha introdotto sinonimi o strutture variate che mantengono il significato ma riducono la similarità lessicale. **Uniformità di Stile**: Le user stories dell'oracolo manuale seguono uno stile uniforme, rendendole più facilmente comparabili alle user stories generate. ChatGPT, invece, può introdurre variazioni stilistiche che, pur mantenendo la correttezza semantica, riducono i punteggi di precisione e di recall.

1.5 Struttura della tesi

- **Capitolo 1 Introduzione**: si descrivono il contesto e le motivazioni.
- **Capitolo 2 Background**: si trova un' introduzione allo sviluppo *Agile, a NLP e metodi di language processing.*
- **Capitolo 3 Stato dell'arte**: descrive approccio e soluzioni già implementati in passato e la differenza dall'approccio proposto.
- **Capitolo 4 Metodo di Ricerca**: si descrivono le 7 parti di cui l'approccio proposto è composto.
- **Capitolo 5 Analisi dei Risultati**: si mostrano e si analizzano i risultati ottenuti.
- **Capitolo 6 Conclusioni**: si descrive ciò che si è sviluppato e si propongono gli sviluppi futuri.

Capitolo 2

Background

2.1 Sviluppo Agile

Lo sviluppo Agile fu ideato per la prima volta nel 2001, quando un gruppo di esperti si riunì per discutere di un nuovo modo di sviluppare che fosse più leggero, veloce e reattivo ai cambiamenti rispetto ai tradizionali metodi di sviluppo. Elaborarono così il manifesto **Agile per lo sviluppo del software** in cui furono descritti 4 principi fondamentali:

- **Gli individui e le interazioni più che i processi e gli strumenti**: Vi è un focus maggiore sulla comunicazione diretta tra gli individui interessati al progetto rispetto ai processi formali e strumenti.Per esempio, una conversazione diretta tra sviluppatori e clienti può risolvere un problema o chiarire un dubbio molto più velocemente di una lunga catena di email o documentazione.

- **Il software funzionante più che la documentazione esaustiva** : Vi è un maggior focus sul consegnare un software utilizzabile piuttosto che dedicare tempo e risorse alla creazione di una documentazione molto dettagliata che potrebbe essere ridondante o non utile.

- **La collaborazione col cliente più che la negoziazione dei contratti**: Si preferisce un approccio dinamico e collaborativo con il cliente, dove la fiducia e la comunicazione aperta permettono di rispondere meglio alle sue esigenze, piuttosto che focalizzarsi esclusivamente su accordi contrattuali.

- **Rispondere al cambiamento più che seguire un piano**: All'insegna della flessibilità e al cambiamento durante lo sviluppo di un progetto,

l'agile promuove una risposta attiva e veloce, garantendo che il progetto evolva in linea con le esigenze del cliente. Invece di attenersi rigidamente a un piano prestabilito che potrebbe diventare irrilevante.

In sostanza, lo sviluppo agile si basa su iterazioni brevi che rilascino una funzionalità, favorisce la continua evoluzione dei requisiti e la collaborazione tra team di sviluppo e stakeholder.
L'estrazione automatica di user stories dai requisiti è particolarmente utile nello sviluppo agile. Infatti proprio perché l'agile è basato sulle iterazioni brevi e di frequenti rilasci è significativo avere rapidamente le user stories, così da aumentare la velocità di consegna. Proprio perché l'agile favorisce la continua evoluzione dei requisiti, che, possono cambiare in base al feedback con il cliente, è necessario modificare le user stories in tempi rapidi.

2.2 NLP

Natural Language Processing (NLP), o elaborazione del linguaggio naturale, è un campo dell'intelligenza artificiale che si concentra sulla comprensione, interpretazione e generazione del linguaggio umano da parte delle macchine. NLP consente di trasformare il linguaggio naturale in una forma comprensibile per i computer, facilitando applicazioni come il riconoscimento delle entità, la traduzione automatica, e molto altro. Ecco alcuni dei compiti principali di NLP:

1. **Tokenizzazione**: suddivide il testo in unità più piccole, come parole o frasi, rendendo il linguaggio più gestibile per l'elaborazione.

2. **Tagging grammaticale (POS Tagging)**: identifica la funzione grammaticale delle parole (come sostantivi, verbi, aggettivi) all'interno di una frase.

3. **Riconoscimento delle Entità Nominate (NER)**: individua e classifica entità rilevanti, come persone, luoghi od organizzazioni.

4. **Dependency parsing**: analisi delle dipendenze, analizzare la struttura sintattica di una frase per comprendere le relazioni tra le parole di cui è composta.

2.2.1 Pos Tagging

La grammatica tradizionale considera 8 parti del discorso: *nomi, pronomi, aggettivi, verbi, avverbi, preposizioni, congiunzioni e interiezioni.*
Più recentemente sono state introdotte nuove classi morfologiche chiamate **Tagset**:

- 45 Penn Treebank tagset (Marcus 1993)
- 87 C5 tagset (Lacaster 1997)
- C7 tagset 146 (Leech 1994)

Le classi considerate possono essere *chiuse o aperte*

- **Classi chiuse** sono quelle che hanno dei membri fissi (es. preposizioni). Tendono a contenere le parole funzionali (di, e, che, da, in, ...) che sono brevi, frequenti e hanno un ruolo nella grammatica
- **Classi aperte** invece sono soggette all'aggiunta di nuovi termini (es. verbi e sostantivi).

Un **Pos Tagger** ha l'obiettivo di assegnare un tag a ciascuna parola di un documento.

Esempio:
the/*DET* **cat**/*NN* **jumps**/*VBZ* **on**/*IN* **the**/DET **table**/*NN*

Nel linguaggio naturale ci possono essere più tag compatibili con una parola (ambiguità).
Did/*MD* **you**/*PRN* **box**/*?* **your**/*DET* **belonging**/*NN*
La parola box deve essere etichettata con VB, verbo, e non NN, nome. Il POS tagger deve essere in grado di far fronte a quest'ambiguità.

Algoritmi POS tagging

- **Tagger basati su regole**: Prevedono la creazione "manuale" di un ampio database di regole che specificano per i casi ambigui le condizioni da verificare per l'assegnazione di ogni tag possibile.
  Esempio:
  una parola è un sostantivo se è preceduta da un articolo
- **Tagger probabilistici**: In genere risolvono le ambiguità stimando la probabilità che una data parola abbia un dato tag in un dato contesto usando un documento di riferimento.

Tag	Description	Tag	Description
CC	Coordinating Conjunction	PRP$	Possessive pronoun
CD	Cardinal Number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential there	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	To
JJR	Adjective, Comparative	UH	Interjection
JJS	Adjective, Superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund or present participle
NN	Noun, singular or mass	VBN	Verb, past participle
NNS	Noun, plural	VBP	Verb, non-3rd person singular present
NNP	Proper noun, singular	VBZ	Verb, 3rd person singular present
NNPS	Proper noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP$	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb

Figure 2.1: La figura mostra le parti del discorso dal **English Penn Treebank tagset** (45)

- **Tramite Machine learning**: Tramite l'addestramento di un modello su di un dataset etichettato *parola-tag*, si esegue la predizione del tag di una nuova parola. Quindi, classificazione di una parola a una classe, in questo caso, un pos tag.

2.2.2 NER

Ogni volta che leggiamo una frase di un testo, noi umani siamo in grado di categorizzare subito le parole in nomi di persone, aziende, luoghi, valori etc...
Le macchine però non possiedono questa abilità naturale, richiedono quindi di un metodo per categorizzare le parole, il NER.
L'obiettivo primario di NER è di processare *dati strutturati e non* e di clas-

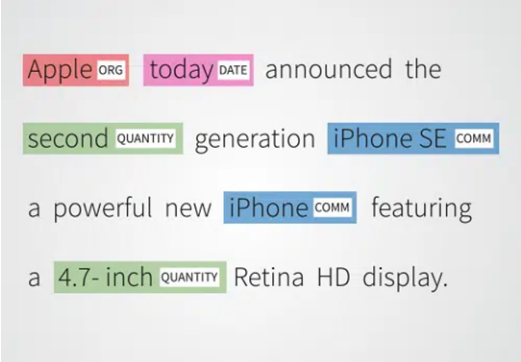

Figure 2.2: Esempio categorizzazione NER

sificare le *named entities* rilevate in categorie predefinite. Possiede, quindi, due fasi distinte:

- **NAMED ENTITY RECOGNITION** dove vengono identificate una o più parole dal documento, le named entities.
- **NAMED ENTITY CLASSIFICATION** dove vengono classificate le parole identificate in categorie predefinite.

La NAMED ENTITY RECOGNITION è la prima fondamentale parte del processo NER. Diversi algoritmi vengono adottati per questa fase:

- **Regole basate su pattern**, in cui il sistema sfrutta pattern lessicali e sintattici (come l'iniziale maiuscola per i nomi propri).
- **Modelli di Machine Learning**, che sono addestrati su dati annotati per riconoscere entità sulla base di caratteristiche.

Dopo aver riconosciuto le entità, il sistema passa alla classificazione cioè alla NAMED ENTITY CLASSIFICATION.
In questa fase, ogni entità identificata viene assegnata a una categoria specifica, come "Person", "Location", "Organization", ecc. La classificazione si

basa comunemente su **Modelli di Machine Learning classificatori**.

Esempio: "*Barack Obama ha visitato Parigi nel 2015 e ha incontrato i membri delle Nazioni Unite.*" Ner classificherà: **Barack Obama**/*Person*, **Parigi**/*Location*, **2015**/*Date* **Nazioni Unite**/*Organization*.

2.2.3 Dependency Parsing

L'obiettivo del *dependency Parsing*, ovvero analisi delle dipendenze è la costruzione di un albero, il **parse tree**, che rappresenta la relazione tra le dipendenze delle parole di una frase. **Esempio**: "Book that flight".

- "Book" è il verbo principale *ROOT*.
- "that" è l'oggetto diretto *dobj* del verbo "book".
- "flight" è l'oggetto del pronome *that*, con la relazione di determinante *det*.

L'algoritmo che meglio descrive il Dependency Parsing è il **Graph-based Dependency Parsing**.
L'algoritmo partendo da una **struttura a grafo** in cui ogni parola è un nodo nel grafo, e le relazioni grammaticali tra le parole sono rappresentate da archi diretti pesati che collegano i nodi, costruisce un **Maximum spanning Tree (MST)** ovvero un albero massimo di ricoprimento.
Restituendo quest'albero l'algoritmo di fatti restituisce l'assegnazione delle relazioni più probabile tra tutte le parole che sono presenti in una frase.
L'assegnazione dei pesi sugli archi del grafo iniziale è una fase preliminare, ed è eseguita da modelli *Machine Learning di regressione* addestrati su dataset contenenti migliaia di relazioni tra due parole e che adesso sono in grado di predire il peso, quindi un numero intero, da assegnare a ogni arco del grafo di partenza.

2.3 Estrazione delle user stories dal linguaggio naturale

Come accennato, lo sviluppo Agile Software Development è noto per la flessibilità nel supportare i cambiamenti aziendali/organizzativi.
La user story sono una notazione utile nello sviluppo software, in particolar modo Agile, per descrivere le funzionalità del software dal punto di vista dell'utente.
Le user stories si compongo di tre aspetti costitutivi:

1. **L'aspetto di chi**: rappresenta il ruolo/tipo dell'utente.
2. **L'aspetto del cosa**: rappresenta lo scopo/caratteristica.
3. **L'aspetto del perché**: rappresenta la ragione/scopo.

In ogni user story sono presenti i seguenti elementi:

1. **Ruoli**: I ruoli sono caratteristiche comportamentali astratte degli attori in contesti specifici.
2. **Tasks**: Gli obiettivi sono condizioni che le parti interessate desiderano raggiungere.
3. **Hard-goals**: Hard-goals sono condizioni che le parti interessate(stakeholders) devono raggiungere e possono essere misurati.
4. **Soft-goals**: sono condizioni che le parti interessate(stakeholders) devono raggiungere e non possono essere misurati.

Esempio:

User Story: *Come utente amministratore, voglio poter generare report settimanali sulle prestazioni del team, in modo da monitorare l'efficacia delle attività e intervenire tempestivamente in caso di necessità.*

Descrizione degli elementi

- **Ruoli**: In questa user story, il ruolo è quello di "utente amministratore," che è responsabile della supervisione e del monitoraggio delle prestazioni del team, quindi agisce come una figura di controllo e analisi.
- **Tasks**: Il task principale è la generazione di report settimanali, che consente all'amministratore di monitorare l'efficacia delle attività. L'obiettivo è quindi raccogliere informazioni regolari sulle prestazioni.
- **Hard-goals**: In questo caso, un hard-goal potrebbe essere "ottenere un report settimanale dettagliato," che permette di valutare quantitativamente la frequenza dei report e la completezza delle informazioni incluse.
- **Soft-goals**: In questa user story, un soft-goal potrebbe essere "monitorare l'efficacia delle attività" con l'intento di garantire un miglioramento qualitativo continuo. Non è un obiettivo quantificabile direttamente, ma è comunque un risultato desiderato che influenza il processo decisionale e la gestione del team.

Capitolo 3

Stato dell'arte

L'applicazione del Natural Language processing per l'estrazione di user stories trova luogo nel scovare difetti, generare modelli/artefatti, ridurre l'ambiguità intrinseca del linguaggio naturale.

Diversi studi si sono concentrati nell'estrarre le user stories da modelli o artefatti, come use case diagram [5][6][19], sequence diagram [4], class diagrams [3] [13]. Molti di meno invece hanno estratto user stories dal linguaggio naturale. Questo perché il linguaggio naturale ha una forma non strutturata che rende complesso estrarre gli aspetti del chi, del cosa e del perché.
La tecnica proposta nello studio proposto di Paige Rodeghero [17] estrae informazioni rilevanti per la creazioni di user stories da conversazioni tra clienti e sviluppatori.
Nello studio sono stati costruiti due modelli di machine learning che classificano una conversazione, trascritta, come contente informazioni rilevanti per la creazione di user stories. Lo studio ha buoni risultati ma non identifica gli aspetti del chi, cosa e perché in quanto etichetta una conversazione come rilevante, ovvero identifica se sono presenti oppure no informazioni di interesse per la fase di raccolta di requisiti dello sviluppo del software.

Nello studio Henriksson e Zdravkovic [7] è stato proposto una soluzione composta di quattro fasi che potrebbe consentire agli sviluppatori di software di identificare le user stories da artefatti prodotti da esseri umani (report, fogli di lavoro, ecc. . .) o da macchine (sensori e registri). Lo studio suggerisce che i meta dati dei testi raccolti potrebbero essere ricondotti a user stories, quindi, questi meta dati potrebbero utilizzati per creare nuove user stories. Tuttavia, questa soluzione utilizza fonti ben strutturate, che non sono sempre disponibili soprattutto in contesti aziendali dove i requisiti cambiano in modo veloce.

Il progetto **user-story-generator** su GitHub di [11] è un esempio di utilizzo di modelli di linguaggio di grandi dimensioni (LLM) per generare user stories. Utilizza un modello di linguaggio avanzato che è stato addestrato su un ampio dataset di storie utente. Il modello può generare storie utente coerenti e ben strutturate in base ai prompt forniti dagli utenti. Il progetto include un'API che espone questa funzionalità, permettendo agli sviluppatori di integrare facilmente il generatore di storie utente nei loro flussi di lavoro. Questo progetto differisce di molto dal nostro proprio perché fa uso di un LLM. Gli LLM sono addestrati su enormi quantità di dati e possono comprendere il contesto in modo molto più profondo rispetto ai modelli NLP tradizionali.
Gli LLM sono in grado di generare testo che è molto simile a quello scritto da un essere umano. Questo rende le storie utente generate più naturali e facili da comprendere. Gli approcci NLP tradizionali possono generare testo che è meno fluido e più meccanico, poiché spesso si basano su modelli di linguaggio meno sofisticati.
Gli approcci NLP tradizionali spesso si basano su regole predefinite e modelli più semplici.
I modelli NLP tradizionali possono generare testo che è meno fluido e più meccanico, poiché non hanno la stessa capacità di comprensione contestuale degli LLM.

Lo studio di C. A. dos Santos and K. Bouchard and B. Minetto Napoleão [18] ha proposto un'altra soluzione generazione automatica di user story, analizzando. Sono stati identificati dei set di dati poi utilizzati per i modelli di training e testing. Tecniche di revisione degli approcci di Natural Language Processing (NLP) e Machine Learning (ML) sono stati applicati. Valutazione: valutazione dei metodi per determinare la qualità delle user story generate. La ricerca ha inoltre evidenziato sfide come la scarsità di dati disponibili al pubblico e che la generazione di user stories rimane ancora un campo in cui fare ricerca. L' approccio proposto si affida per lo più alle tecniche di NLP per l'estrazione delle user stories. L'approccio proposto utilizza PROMISE un dataset standard invece la ricerca di dos Santons utilizza documenti disponibili più o meno standardizzati. Infine, l'approccio proposto propone un modo per valutare le user stories prodotte mentre la ricerca di dos Santons indica che non ci sono metodi standardizzati o rigorsi per la valutazione.

Lo studio analizza l'elicitazione dei requisiti tramite user stories, proponendo che coinvolgere gruppi, anziché individui, migliori il processo in termini di quantità, unicità e copertura delle storie generate. La metodologia si basa su un esperimento di laboratorio che confronta le user stories prodotte da partecipanti individuali con quelle generate da gruppi. L'efficacia è mis-

urata attraverso tre metriche: numero di storie prodotte, unicità delle storie e copertura dei requisiti. Lo studio, pur essendo parzialmente completato, suggerisce che ulteriori ricerche siano necessarie per trarre conclusioni definitive.

Lo studio di [14] propone che di coinvolgere gruppi, anziché individui, così da migliorare il processo in termini di quantità, unicità e copertura delle storie generate.
La metodologia si basa su un esperimento di laboratorio che confronta le user stories prodotte da partecipanti individuali con quelle generate da gruppi. L'efficacia è misurata attraverso tre metriche: numero di storie prodotte, unicità delle storie e copertura dei requisiti.
La differenza con l'approccio proposto è che lo studio di Nguyen enfatizza l'aspetto collaborativo dell'elicitazione tramite gruppi, invece l'approccio proposto si concentra sull'automatizzazione del processo con tecniche di NLP applicate a dataset come PROMISE. Inoltre, l'obiettivo dello studio è migliorare la qualità dell'elicitazione in una fase iniziale attraverso la partecipazione umana, mentre l'approccio che proponiamo mira a ottimizzare la generazione di user stories già derivate dai requisiti, riducendo l'intervento manuale e gli errori tramite modelli linguistici.

La ricerca di [16] è un altro esempio di utilizzo di LLM nello specifico il potente Chat GPT-4, per automatizzare la creazione di user stories dai documenti dei requisiti. Questo approccio mira a ridurre il carico di lavoro degli ingegneri software, che normalmente dedicano molto tempo a incontri e analisi per comprendere i requisiti e generare storie pertinenti. L'output prodotto dallo strumento è un formato **JSON**, che facilita l'integrazione nei principali strumenti di gestione del ciclo di vita del software. Rispetto all'approccio che proponiamo, lo studio utilizza un LLM un modello avanzato di linguaggio, per generare user stories.Invece il nostro approccio usa tecniche di NLP focalizzate sull'estrazione di storie. L'output è **JSON**, quindi facilmente integrabile in strumenti di gestione progetti come il nostro approccio invece fornisce l'output come testo.

Capitolo 4

Metodologia

4.1 Architettura dell'approccio

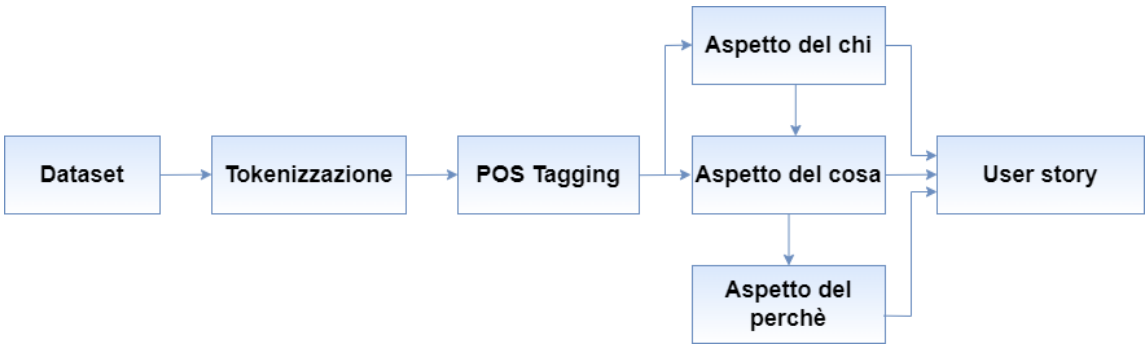

Figure 4.1: La Figura mostra l'architettura dell'approccio proposto.

L'architettura dell'approccio consiste di 7 parti:

1. **Ottenimento dataset**: descrizione del dataset utilizzato dall'approccio e di come è strutturato.

2. **Tokenizzazione**: Suddivisione delle frasi in token ovvero parti più piccole.

3. **Suddivisione della parte del discorso (POS)**: applicazione di un POS PATTERN per l'identificazione dei candidati degli aspetti del chi e del cosa.

4. **Identificazione degli aspetti del chi**: Applicazione della *verifica degli iperonimi, NER e dependency parsing* per filtrare il più precisamente i candidati degli aspetti del chi al fine di ottenere gli aspetti del chi finali.

5. **Identificazione dell'aspetto del cosa**: Applicazione della *verifica degli iperonimi, identificazione degli oggetti del cosa* per filtrare il più precisamente i candidati degli aspetti del cosa al fine di ottenere gli aspetti del cosa finali.

6. **Individuazione dell'aspetto del perché**: Gli aspetti del perché sono derivati direttamente dagli aspetti del cosa tramite il *dependency parsing.*

4.2 Dataset

Il dataset scelto è **PROMISE** (Predictor of Metrics for Software Engineering) è una raccolta di dati utilizzata principalmente per il *benchmarking* di modelli predittivi nel campo dell'ingegneria del software, con particolare focus sulla previsione delle metriche di qualità del software.
Il dataset PROMISE è suddiviso in **49 progetti software** distinti. Per ognuno conosciamo *l'indice del progetto, il requisito e il tipo di requisito.*

```
1,'The system shall allow modification of the display.',F
1,'The system shall offer a display of all the Events in the exercise.',F
1,'The system shall filter data by: Venues and Key Events.',F
1,'The system shall allow a user to define the time segments',F
1,'The system shall display the local and exercise time in separate clocks',F
1,'The system shall offer the ability to pause and resume the refresh of data.',F
1,'The system shall provide charts for the Activity or Event actual versus assumed time.',F
1,'The system shall provide a history report of changes made to the Activity or Event data',F
2,'The look and feel of the system shall conform to the user interface standards of the smart device.',LF
```

Figure 4.2: Estratto del dataset PROMISE

4.3 Applicazione tokenizzazione per ogni requisito

Ottenuto il dataset, per ogni requisito eseguiremo la fase di tokenizzazione. La tokenizzazione è il processo di suddividere un testo in unità più piccole chiamate *token*. Un *token* può essere una parola, una parte di parola (come un prefisso o suffisso), o anche un simbolo come punteggiatura. Per ogni requisito otterremo token che rappresentato parole che saranno soggetti al Pos pattern.

Ecco un esempio:

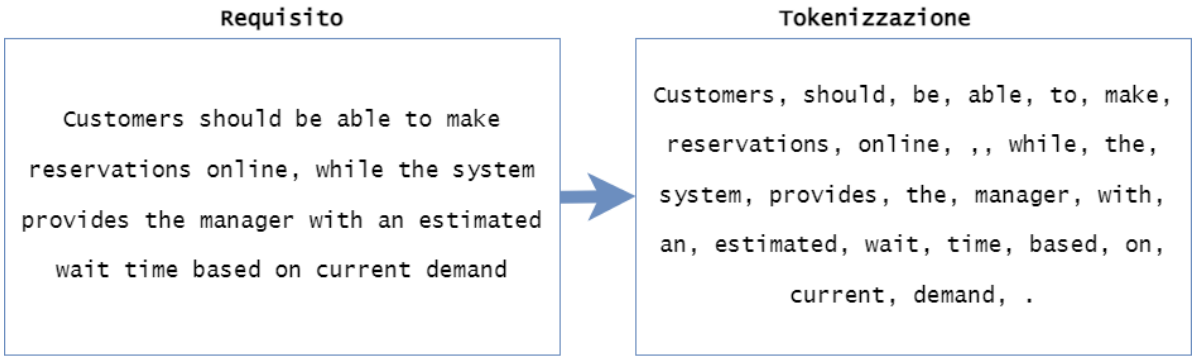

Figure 4.3: Esempio Tokenizzazione di un requisito

4.4 Applicazione POS Tagging per ogni requisito

Di ogni requisito vengono etichettate tutte le parole utilizzando le *parti del discorso* (POS). Ogni token ha un tag corrispettivo come sostantivi, verbi, aggettivi etc...
L'aspetto di chi è una *frase nominale (noun phrase)* ovvero persona, ruolo od organizzazione.
Invece, l'aspetto del cosa è un verbo seguito da una *frase nominale(noun phrase).*
Presi in input i token e i rispettivi tag viene applicato il POS Pattern secondo le regole:

- **ASPETTO CHI**: (DT)*((ADJ) or (NN) or (PRP))+

- **ASPETTO COSA**:(ADJ or VERB)((DT)*((ADJ) or (NN) or (PRP))+)+

Per esempio: "Program/NN Administrator/NN" è scelto come candidato dell'aspetto del chi per le fasi successive poiché ricade nella regola dell'aspetto del chi.

La fase di identificazione degli aspetti del chi e del cosa comincia applicando queste regole. L'applicazione delle regole darà in output i primi candidati, soggetti poi alle fasi successive.

Esempio:

Applicazione POS Tagging:

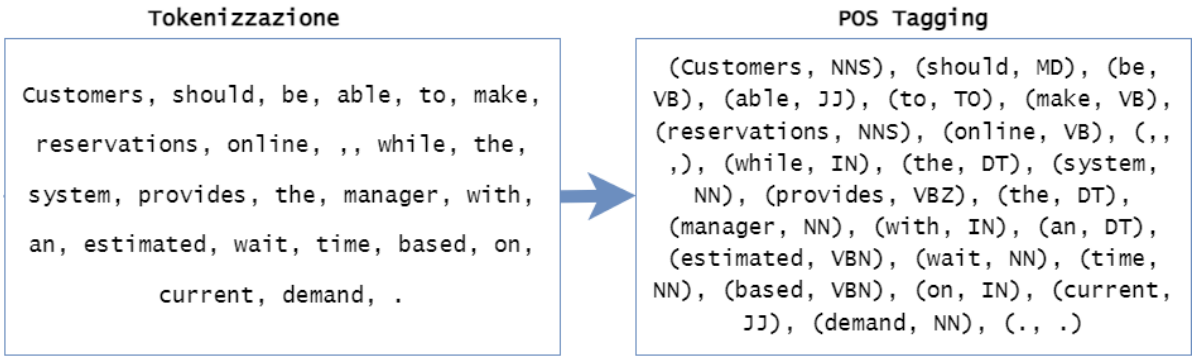

Figure 4.4: Esempio Pos Tagging di un requisito

Applicazione POS Pattern per gli aspetti del chi:

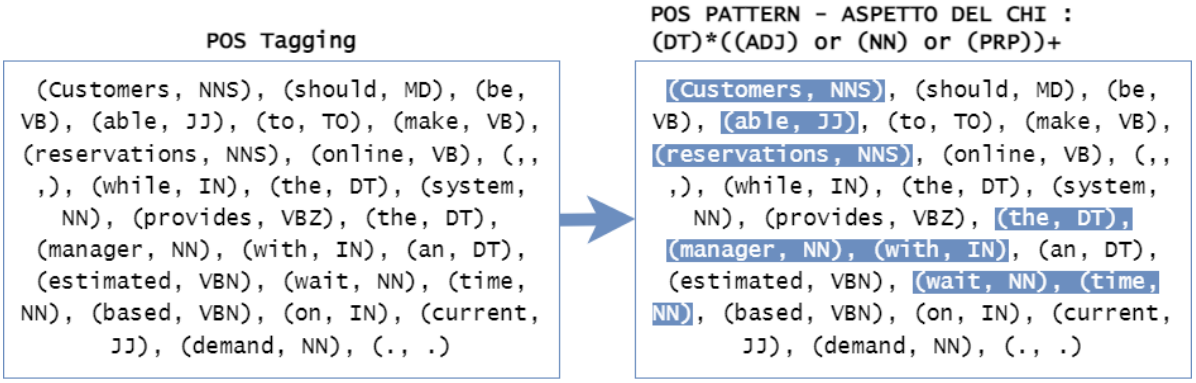

Figure 4.5: candidati del chi di un requisito

Evidenziati vi sono i candidati del chi, in questo caso: (Customers, NNS), (able, JJ), (reservations, NNS),(the, DT), (manager, NN), (with, IN), [(wait, NN), (time, NN).

Applicazione POS Pattern per gli aspetti del cosa:

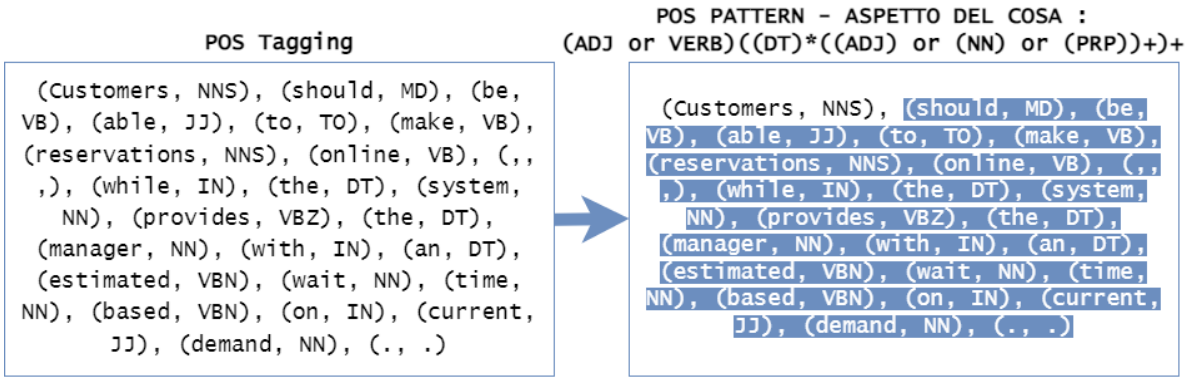

Figure 4.6: candidati del cosa di un requisito

similmente, tra le parentesi quadre vi sono i candidati del cosa, in questo caso: [(should, MD), (be, VB), (able, JJ), (to, TO), (make, VB), (reservations, NNS), (online, VB), (,, ,), (while, IN), (the, DT), (system, NN), (provides, VBZ), (the, DT), (manager, NN), (with, IN), (an, DT), (estimated, VBN), (wait, NN), (time, NN), (based, VBN), (on, IN), (current, JJ), (demand, NN)], (., .)] .

4.5 Identificazione dell'aspetto del chi

Identificare gli aspetti del chi è simile al chiedersi chi sia lo skateholder coinvolto in un requisito software. Molti tool [8] identificano l'aspetto del chi tramite il semplice pattern soggetto-predicato-oggetto di una frase. Ma questo è un approccio adatto per frasi semplici e ben strutturate, quando nella maggior parte dei casi i requisiti si presentano in modo non strutturato. Quindi si propone l'identificazione dell'aspetto del chi in una combinazione di diversi metodi:

- Verifica degli ipernomi da Wordnet [12].
- Applicazione del metodo NER.
- L'analisi delle dipendenze. [9]

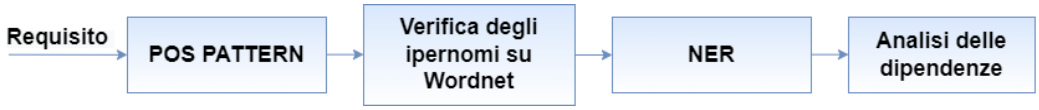

Figure 4.7: La figura mostra i passi che compongono l'identificazione dell'aspetto del chi

Per poter comprendere la prima fase ovvero verifica degli ipernomi da Wordnet è prima necessario capire cos'è un ipernome.
WordNet è un ampio database per la lingua inglese, sviluppato alla Princeton University. È strutturato in modo che le parole siano organizzate per significato, infatti, in WordNet, un **synset** (o "set di sinonimi") è un insieme di parole che condividono lo stesso significato. Un **ipernome**(o iperonimo) è un termine che rappresenta una categoria più generale a cui appartiene un synset specifico. Per esempio:

- "Essere vivente" è un ipernome di "animale", poiché "animale" è una categoria specifica di esseri viventi.

Presi in input i primi candidati dati dalla fase precedente ovvero applicazione del Pos pattern su di requisito, ricorsivamente esploreremo tutti quanti gli iperonimi di una data parola/token e, se si trovano ipernomi in cui è presente "person" o "group" allora la parola/token è considerata candidato del chi per le fasi successive.

La seconda fase è l'applicazione di NER(Named entity recognition).
Il NER è un metodo efficace per identificare gli aspetti del chi in frasi strutturate o complesse. In sostanza, NER è una tecnica per classificare entità. Solitamente, le entità denominate sono etichettate in nomi di persone, organizzazioni, luoghi, espressioni temporali e quantità.
Eseguiremo l'esclusione delle entità identificate da NER che sono state etichettate con etichette diverse da PERSON o GROUP.

Esempio:
Requisito:*"Google employees must be able to modify their personal information in the internal HR portal."*
POS Pattern : *"Google employees" , "able", "personal information"*
NER:*"Google"*
Risultato :*"Google employees"*

L'ultima fase è l'applicazione del *dependency parsing*, ovvero analisi delle dipendenze. Il *dependency parsing* consente di identificare il ruolo di ciascun elemento nella frase, come il soggetto, l'oggetto, il verbo principale, e come queste parole si relazionano. Ogni relazione tra parole è etichettata con un tipo specifico, come *nsubj* , ovvero soggetto nominale o *obj* per l'oggetto diretto.
Identificheremo relazioni del tipo:*'nsubjpass','nsubj"* , rispettivamente *soggetto nominale passivo e soggetto attivo* ovvero chi nella frase subisce l'azione e chi nella frase esegue l'azione.

Esempio:

```
Node (from)-->   Relation     -->Node (to)

be                nsubj          Customers
be                 aux             should
be                 ROOT                be
be                acomp              able
make               aux                to
able              xcomp              make
make               dobj       reservations
make              advmod            online
be                punct                 ,
provides           mark             while
system             det                the
provides          nsubj            system
be                advcl           provides
manager            det                the
provides           dobj           manager
provides           prep              with
time               det                an
time               amod          estimated
time           compound               wait
with               pobj               time
time               acl              based
based              prep                on
demand             amod           current
on                 pobj            demand
be                punct                 .
[Tree('candidate aspect of who', [('Customers', 'NNS')]),
 Tree('candidate aspect of who', [('the', 'DT'), ('manager', 'NN'), ('with', 'IN')])]
```

Figure 4.8: Esempio Esecuzione Dependency Parsing

Esempio:

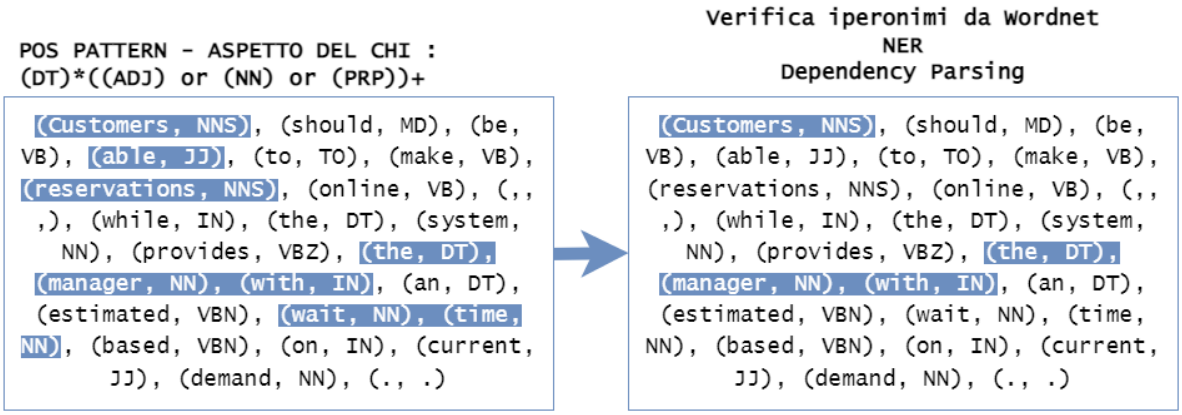

Figure 4.9: Esempio Risultato Aspetti del chi

Come possiamo vedere dai candidati iniziali ne sono stati rimossi diversi: *able, reservations, wait time.*

4.6 Identificazione dell'aspetto del cosa

Trovare gli aspetti cosa è equiparabile all'estrazione di funzionalità software perché l'obiettivo è ottenere requisiti software dal linguaggio naturale.
L'identificazione dell'aspetto del cosa è suddivisa in tre fasi:

- Verifica dei *lexname*, ovvero nomi lessicali da Wordnet [12].
- Identificazione degli oggetti del cosa.
- Combinazione candidati aspetti del cosa e oggetti del cosa.

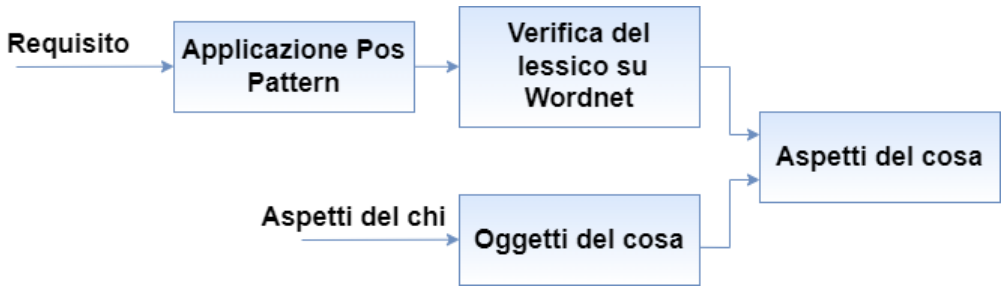

Figure 4.10: La figura mostra i passi che compongono l'identificazione dell'aspetto del cosa

Mentre un **Iperonimo** indica una relazione gerarchica specifica (es: cane è un tipo di animale), **Lexname**: è una categoria più ampia e generica senza gerarchie. Presi in input i primi candidati dati dalla fase precedente ovvero applicazione del Pos pattern, nella prima fase dell'identificazione dell'aspetto del cosa esploreremmo tutti quanti i nomi lessicali di una data parola/token e, se si trovano nomi lessicali in cui è presente:*cognition, communication, creation, contact, motion, perception, possession, assist* allora la parola è considerata candidato del cosa per le fasi successive.

La seconda fase ovvero identificazione degli oggetti del cosa. Gl'oggetti del cosa consistono in parole/token ricavate dalle relazioni che gli aspetti del chi hanno con i candidati dell'aspetto del cosa trovati al passo precedente.
Si utilizzano gli aspetti del chi trovati al passo precedente per controllare tramite il *dependency parsing* le relazioni head-dependent che essi hanno.

Navigando *il parse tree* restituito dal Depedency Parsing, leggeremo tutte le relazioni head-dependent tra tutti i token di una frase, se tali relazioni sono del tipo *"nsubj", "pobj", "agent"* allora viene si salverà la parte head della relazione in una lista di appoggio.
Tuttavia, salvare solo la parte head della relazione non restituisce un risultato apprezzabile, poiché in sostanza abbiamo salvato solamente un token. Per questo, una volta salvata la parte head della relazione , si itererà all'indietro dal token salvato fino al nodo ROOT dell' *albero*. Il risultato di questa fase è una lista degli oggetti del cosa.

L'ultima fase consiste nella combinazione degli oggetti del cosa e dei candidati aspetti del cosa. Per ogni requisito vogliamo filtrare i candidati aspetti del cosa per trovare solo quelli validi.
Per cui, filtriamo i candidati aspetti del cosa tramite la lista oggetti del cosa che abbiamo salvato per ogni requisito. Per un dato requisito si controlla se un token della lista degli oggetti del cosa è presente nella lista dei candidati degli aspetti del cosa, se ciò è vero allora il candidato in questione diventa un aspetto del cosa finale. Tuttavia, sono risultati alcuni requisiti con una lista oggetti del cosa vuota, per cui invece di avere una combinazione esclusiva gli aspetti del cosa ricavati tramite gli oggetti del cosa sono considerati come primi candidati ma non sono gli unici.

Esempio:

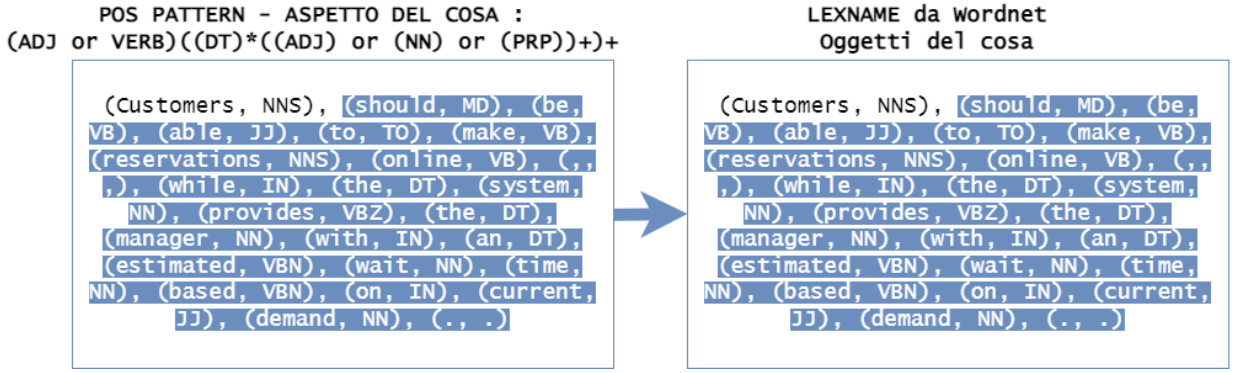

Figure 4.11: Esempio fase lexname from Wordnet per l'aspetto del cosa

4.7 Individuazione dell'aspetto del perché

L'aspetto del perché è ottenuto tramite il *dependency parsing*, che restituisce il *parse tree*; Per ogni requisito si cercano relazioni causali tra i suoi aspetti del cosa. Se un aspetto del cosa ha una relazione causale della forma di **advcl**, ovvero *modificatore della clausola avverbiale* o **xcomp** , ovvero *complemento della clausola con soggetto esterno*, viene considerato un aspetto del "perché". Esempio:

- Il sistema deve inviare una notifica **quando viene rilevata** una modifica nei dati. La clausola **quando viene rilevata una modifica nei dati** è di tipo *advcl*.
- L'utente può selezionare un documento **per visualizzare** il contenuto. In questo caso, **per visualizzare** non ha un soggetto proprio e si riferisce all'utente, soggetto della clausola principale. La clausola **per visualizzare il contenuto** è di tipo *xcomp*.

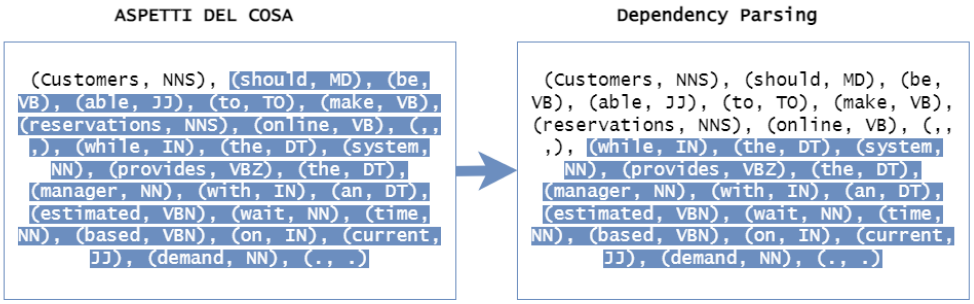

Figure 4.12: La figura mostra i passi che compongono l'identificazione dell'aspetto del perché

Viene estratto l'aspetto del perché dall'aspetto del cosa, in generale, l'aspetto del cosa contiene l'aspetto del perché. In requisti più semplici spesso non è presente.

4.8 Valutazione

La fase di valutazione è stata eseguita creando due oracoli per ognuno dei tre documenti utilizzati dal dataset PROMISE, rispettivamente il 3°, il 34° e il 49° documento.
Il primo dei due oracoli è stato scritto manualmente intabellando quindi il requisito, l'aspetto del chi, del cosa e del perché. Il secondo oracolo invece è stato generato da ChatGPT che, similmente, ha in tabellato il requisito, l'aspetto del chi, del cosa e del perché. Le metriche di valutazione utilizzate sono tre: **rouge[10], blue[15] e meteor [1]**

4.8.1 La metrica BLEU

La metrica BLEU (Bilingual Evaluation Understudy) si basa sul confronto tra la sequenza di *n-grammi* (sequenze di parole di lunghezza variabile)[15] del testo generato e quelli del testo di riferimento. BLEU calcola il numero di n-grammi condivisi tra il testo generato e quello di riferimento, attribuendo un punteggio compreso tra 0 e 1, dove un punteggio più alto indica una maggiore somiglianza.
Ecco un esempio:
Testo di riferimento:"Il gatto nero salta sopra il muro"
Testo generato:"Il gatto nero corre sopra il muro".

Estrazione dei bi-grammi, uni-grammi, tri-grammi

- **testo di riferimento**: "Il gatto", "gatto nero", "nero salta", "salta sopra", "sopra il", "il muro".
- **testo generato**: "Il gatto", "gatto nero", "nero corre", "corre sopra", "sopra il", "il muro".

$$\textbf{Precisione BLEU} = \frac{\textbf{bigrammi comuni}}{\textbf{bigrammi generati}} = \frac{\mathbf{4}}{\mathbf{6}} = 0,67$$

$$\textbf{Precisione BLEU} = \sqrt[3]{\textbf{unigrammi} * \textbf{bigrammi} * \textbf{trigrammi}} = 0.57$$

Applicazione penalità per testi brevi o per parole ricorrenti La penalità di brevità o di ricorrenza è applicata solo se il testo generato è significativamente più corto del testo di riferimento. In questo esempio, entrambi i testi sono della stessa lunghezza né ci sono parole ricorrenti, quindi non è necessaria la penalità.

Il punteggio BLEU per il testo generato rispetto al testo di riferimento, usando solo i bi-grammi, è del 57%.

BLEU è principalmente una metrica di *precisione*, misurando quanto il testo generato copre gli n-grammi del testo di riferimento. Non tiene conto della copertura complessiva del contenuto (cioè, del richiamo) e si concentra su quanti degli n-grammi generati sono corretti rispetto al riferimento.

4.8.2 La metrica Rouge

La metrica ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*) [10] è una serie di metriche utilizzate per valutare la qualità dei testi generati, confrontandoli con testi di riferimento. È spesso impiegata per misurare la somiglianza tra testi, specialmente in applicazioni di Natural Language Processing.
Esistono diversi tipi di Rouge: **ROUGE-N**: misura la sovrapposizione di n-grammi (sequenze di parole di lunghezza variabile) tra il testo generato e il testo di riferimento. I valori più comuni di N sono:

- **ROUGE-1**: basato sui singoli termini (*uni-grammi*)
- **ROUGE-2**: basato sui bi-grammi (coppie di parole consecutive)
- **ROUGE-3** e così via, per n-grammi più lunghi.

ROUGE-L: misura la **Longest Common Subsequence** (LCS), ovvero la sequenza più lunga di parole in comune, mantenendo l'ordine. È utile perché non richiede che le parole siano consecutive, riflettendo meglio il contenuto simile. **ROUGE-W**: variante di ROUGE-L, dà più peso alle sotto sequenze più lunghe, attribuendo punteggi più alti a contenuti che riescono a mantenere una struttura simile al testo di riferimento.

Ecco un esempio :
Testo di riferimento: "Il gatto nero salta sopra il muro alto"
Testo generato: "Il gatto salta sopra il muro"
Uni grammi del testo di riferimento: "Il", "gatto", "nero", "salta", "sopra", "il", "muro", "alto"
Uni grammi del testo generato: "Il", "gatto", "salta", "sopra", "il", "muro"

- **Recall** = $\frac{\textbf{unigrammi comuni}}{\textbf{unigrammi nel testo di riferimento}} = \frac{\textbf{8}}{\textbf{6}} = 0.75$
  Il richiamo misura la proporzione di unigrammi del testo di riferimento che sono presenti nel testo generato.

- **Precision** = $\frac{\textbf{unigrammi comuni}}{\textbf{unigrammi nel testo generato}} = \frac{\textbf{6}}{\textbf{6}} = 1.0$
  La precisione misura la proporzione di unigrammi nel testo generato che coincidono con quelli del testo di riferimento.

- $F1 = 2 * \frac{\textbf{Precision*Recall}}{\textbf{Precision + Recall}} = 2 * \frac{\textbf{1.0}\times\textbf{0.75}}{\textbf{2.1.0+0.75}} = 0.857$
  L'F1-score è la media armonica tra richiamo e precisione, e rappresenta un equilibrio tra i due valori.

ROUGE è più orientata al *richiamo*, misurando quanto bene il testo di riferimento è coperto dal testo generato. A seconda della variante, ROUGE può anche incorporare misure di precisione e di **F1-score** per dare una valutazione più bilanciata tra copertura e accuratezza.

4.8.3 La metrica Meteor

METEOR *(Metric for Evaluation of Translation with Explicit ORdering)* [1] è una metrica sviluppata per valutare automaticamente la qualità dei testi generati, confrontandoli con testi di riferimento. È stata ideata per superare alcune limitazioni di metriche come BLEU, offrendo un'analisi più dettagliata e flessibile della somiglianza tra testo generato e riferimento.

- **Allineamento flessibile**: METEOR esegue un *allineamento* tra il testo generato e quello di riferimento, confrontando le parole in modo più preciso e considerando diversi tipi di corrispondenze tra le parole, non solo gli n-grammi esatti. Utilizza **sinonimi** e **steli delle parole** (radici comuni), considerando come corrispondenti anche parole simili o coniugate, come "corre" e "correre". Questo rende METEOR più flessibile e tollerante alle variazioni linguistiche rispetto a metriche rigide come BLEU.

  Testo di riferimento: "Il gatto nero salta sopra il muro." **Testo generato**: "Il felino nero balza sul muro."
  Sinonimi: "gatto" e "felino" sono sinonimi, quindi METEOR li allinea come corrispondenti, anche "salta" e "balza" sono parole simili.

- **Stemming**: riduce le parole alle loro radici comuni (per esempio, "amato" e "amare" diventano "ama"), migliorando l'allineamento quando le parole sono coniugate o declinate diversamente.

- **Penalità per la discontinuità**: METEOR penalizza l'ordine delle parole diverso rispetto al testo di riferimento. Se la sequenza delle parole nel testo generato non corrisponde, la metrica applica una penalità, mantenendo una preferenza per le frasi che rispecchiano la struttura del testo di riferimento.

- **Calcolo di Precision, Recall e F1-score**:METEOR calcola sia **precisione** che **richiamo** e dà particolare importanza al **richiamo** (ossia alla copertura delle informazioni del riferimento nel testo generato). Utilizza un **F1-score** pesato, favorendo il richiamo, per stimare un punteggio finale che rappresenti un equilibrio tra accuratezza e completezza del contenuto.

Abbiamo concatenato tutte le stringhe ottenute dell'approccio in unica grande stringa, così è stato fatto anche per i due oracoli, ottenendo così per ogni documento tre grandi stringhe. Le grandi stringhe sono state poi valutate secondo le metriche.

Capitolo 5

Analisi dei Risultati

5.1 Aspetto del chi

Di seguito mostriamo un piccolo estratto degli aspetti del chi finali ottenuti affiancati dai requisiti in linguaggio naturale. Per ogni requisito è stato estratto un solo aspetto del chi, anche se in generale per ogni requisito è possibile estrarre più aspetti del chi.

Index	Requirement	Aspect of Who
0	A waiter should be able to use a tablet to take customer orders and send them directly to the kitchen staff for faster order processing.	A waiter
1	Customers should be able to make reservations online, while the system provides the manager with an estimated wait time based on current demand.	Customers
2	The kitchen staff should automatically receive notifications when a new order is placed, and the waiter should be able to track the status of the order on their tablet.	The kitchen staff

Table 5.1: Tabella mostrante requisiti e i rispettivi aspetti del chi individuati

5.2 Aspetto del cosa

Di seguito mostriamo un piccolo estratto degli aspetti del cosa finali ottenuti affiancati dai requisiti in linguaggio naturale. In generale è più probabile estrarre da un requisito un solo aspetto del cosa, essendo che ogni requisito tende a descrivere una sola feature o azione che deve essere presente nel prodotto finale.

Index	**Requirement**	**Aspect of What**
0	A waiter should be able to use a tablet to take customer orders and send them directly to the kitchen staff for faster order processing.	should be able to use a tablet to take customer orders and send them directly to the kitchen staff for faster order processing
1	Customers should be able to make reservations online, while the system provides the manager with an estimated wait time based on current demand.	should be able to make reservations online/while the system provides the manager with an estimated wait time based on current demand
2	The kitchen staff should automatically receive notifications when a new order is placed, and the waiter should be able to track the status of the order on their tablet.	should automatically receive notifications, should be able to track the status of the order on tablet

Table 5.2: Tabella mostrante requisiti e i rispettivi aspetti del cosa individuati

5.3 Aspetto del perchè

Di seguito mostriamo un piccolo estratto degli aspetti del perché finali ottenuti affiancati dai requisiti in linguaggio naturale.Anche se presente in tutte le user stories presentate, in generale, da un requisito non è possibile estrarre l'aspetto del perché direttamente perché non è presente nel requisito.

Index	**Requirement**	**Aspect of Why**
0	A waiter should be able to use a tablet to take customer orders and send them directly to the kitchen staff for faster order processing.	to take customer orders and send them directly to the kitchen staff for faster order processing
1	Customers should be able to make reservations online, while the system provides the manager with an estimated wait time based on current demand.	while the system provides the manager with an estimated wait time based on current demand
2	The kitchen staff should automatically receive notifications when a new order is placed, and the waiter should be able to track the status of the order on their tablet.	to track the status of the order on their tablet

Table 5.3: Tabella mostrante requisiti e i rispettivi aspetti del perchè individuati

5.4 User Stories

Di seguito mostriamo un piccolo estratto delle user stories finali ottenute affiancate dai requisiti in linguaggio naturale. Notiamo due particolari: tutte quelle presentate hanno l'aspetto del perché, solitamente invece non è esplicitamente presente nel requisito iniziale e che dall'ultimo requisito sono state estratte due user stories, questo fa comprendere che da un requisito è possibile estrarre più user stories.

Index	**Requirement**	**User Story**
0	A waiter should be able to use a tablet to take customer orders and send them directly to the kitchen staff for faster order processing.	[AS A waiter I WANT TO be able to use a tablet to take customer orders and send them directly to the kitchen staff for faster order processing SO THAT to take customer orders and send them directly to the kitchen staff for faster order processing]
1	Customers should be able to make reservations online, while the system provides the manager with an estimated wait time based on current demand.	[AS Customers I WANT TO be able to make reservations online, while the system provides the manager with an estimated wait time based on current demand SO THAT while the system provides the manager with an estimated wait time based on current demand]
2	The kitchen staff should automatically receive notifications when a new order is placed, and the waiter should be able to track the status of the order on their tablet.	[AS The kitchen staff I WANT TO automatically receive notifications SO THAT to track the status of the order on their tablet, AS notifications I WANT TO be able to track the status of the order on SO THAT to track the status of the order on their tablet]

Table 5.4: Tabella mostrante requisiti e le rispettive user stories individuate

Come accennato nella sezione di *valutazione* sono stati impiegati tre documenti dal dataset PROMISE. Rispettivamente il 3°, il 34° e il 49°. Per ognuno sono stati creati due oracoli uno manuale, in cui presi i requisiti sono stati annotati gli aspetti del chi, cosa e perchè ed uno generato da ChatGPT.

Terzo documento

3° documento confrontato con l'oracolo generato da ChatGPT ha questi risultati.

- **Rouge1**: *Misura il match di unigrammi (parole singole) tra il testo generato e quello di riferimento.*
  - **Precisione (0.74):** Il 74% delle parole generate si trovano nel riferimento.
  - **Richiamo (0.84)**: L'84% delle parole del riferimento si trova nel testo generato.
  - **F1-score (0.78):** Un buon equilibrio tra precisione e richiamo.
  - Rouge1 ha dei buoi risultati questo ci fa capire che l'approccio e l'oracolo generato da ChatGPT sono ben coperti, ovvero hanno una gran mole di parole uguali.
- **Rouge2**: *Misura il match di bigrammi (coppie di parole consecutive) tra i testi.*
  - **Precisione (0.53):** Circa il 53% dei bigrammi generati si trovano nel riferimento.
  - **Richiamo (0.60):** Circa il 60% dei bigrammi del riferimento è stato catturato.
  - **F1-score (0.56):** Il punteggio è più basso rispetto a ROUGE-1, il che indica che il modello cattura meno informazioni contestuali.
  - Rouge2 ha risultati più scarsi rispetto a Rouge1 quindi modello è meno efficace nel catturare la struttura sintattica o le relazioni tra le parole rispetto al contenuto lessicale.
- **RougeL**: *Misura il match della* ***longest common subsequence (LCS)****, tenendo conto della struttura sequenziale.*
-
  - **Precisione (0.61):** Il 61% degli elementi generati fa parte della sequenza comune più lunga con il riferimento.

  - **Richiamo (0.69):** Il 69% della sequenza più lunga del riferimento è presente nel testo generato.
  - **F1-score (0.65):** Indica che la coerenza strutturale è accettabile.
  - RougeL ha sufficenti risultati, le user stories generate mantengono una buona somiglianza con testo di riferimento, ma c'è ancora spazio per migliorare la fedeltà strutturale.

- **Bleu 57%** : *misura la somiglianza tra il testo generato e il riferimento, concentrandosi su sequenze di parole (n-grammi) e penalizzando le generazioni troppo corte.* Bleu non tiene conto del significato, ma si basa solo sulla sovrapposizione esatta di n-grammi. Avendo un putenggo di 0.57 vuol dire che nel 57% dei casi le user stories generate hanno n-grammi identici a quelle dell'oracolo. In generale valori più alti sono difficili da raggiungere senza un dataset estremamente specifico.

- **Meteor 45%** : *valuta la qualità del testo generato rispetto a un riferimento, tenendo conto di aspetti come la corrispondenza di parole (sinonimi inclusi), il significato e l'ordine.* **0.4517** indica che il modello cattura circa il **45%** della somiglianza complessiva con il testo di riferimento.

Per il 3° documento ma tramite l'oracolo manuale:

- **Rouge1**: *Misura il match di uni-grammi (parole singole) tra il testo generato e quello di riferimento.*
  - **Precisione (0.60):** Il 60% delle parole generate si trovano nel riferimento.
  - **Richiamo (0.89)**: L'89% delle parole del riferimento si trova nel testo generato.
  - **F1-score (0.72):** Un buon equilibrio tra precisione e richiamo.
  - Rouge1 ha dei buoi risultati simili a quelli ottenuti con l'oracolo di ChatGPT, ma migliora per il richiamo. Essendo l'oracolo manuale è più vicino all'input mentre ChatGpt potrebbe aver modificato la struttura del testo ma mantenuto il significato.

- **Rouge2**: *Misura il match di bi-grammi (coppie di parole consecutive) tra i testi.*
  - **Precisione (0.51):** Circa il 51% dei bi-grammi generati si trovano nel riferimento.

  - **Richiamo (0.76):** Circa il 76% dei bi-grammi del riferimento è stato catturato.
  - **F1-score (0.61):** Un buon equilibrio tra precisione e richiamo.
  - Rouge2 peggiora rispetto a Rouge1 anche con l'oracolo di ChatGPT succedeva, tuttavia i valori sono più altri rispetto al confronto con l'oracolo di ChatGPT.

- **RougeL**: *Misura il match della* ***longest common subsequence (LCS)****, tenendo conto della struttura sequenziale.*
  - **Precisione (0.51):** Il 51% degli elementi generati fa parte della sequenza comune più lunga con il riferimento.
  - **Richiamo (0.80):** L' 80% della sequenza più lunga del riferimento è presente nel testo generato.
  - **F1-score (0.64):** Indica che la coerenza strutturale è accettabile, ma non eccellente.
  - RougeL ha sufficienti risultati, le user stories generate mantengono una buona somiglianza con testo di riferimento, ma c'è ancora spazio per migliorare la fedeltà strutturale.

- **Bleu 51%** : *misura la somiglianza tra il testo generato e il riferimento, concentrandosi su sequenze di parole (n-grammi) e penalizzando le generazioni troppo corte.* Rispetto all'oracolo al confronto con l'oracolo di ChatGPT è diminuita, anche se di poco.

- **Meteor 55%** : *valuta la qualità del testo generato rispetto a un riferimento, tenendo conto di aspetti come la corrispondenza di parole (sinonimi inclusi), il significato e l'ordine.* **0.5511** indica che il modello cattura circa il **55%** della somiglianza complessiva con il testo di riferimento. Rispetto all'oracolo al confronto con l'oracolo di ChatGPT è aumentata.

Complessivamente, tra i due oracoli Rouge ha valutazioni simili, Bleu è più alta per ChatGPT mentre Meteor è più alta per l'oracolo manuale. **Un aumento di METEOR** rispetto all'oracolo ChatGPT indica che l'approccio riesce a produrre output che riflettono meglio la qualità linguistica dell'oracolo manuale, anche se potrebbe non essere perfetto in termini di corrispondenza letterale. Quindi, anche se di poco, possiamo affermare che l' approccio è più adatto a replicare user stories realistiche, come quelle che scriverebbe un esperto.

Trentaquattresimo documento

34° documento confrontato con l'oracolo generato da ChatGPT ha questi risultati.

- **Rouge1**: *Misura il match di uni-grammi (parole singole) tra il testo generato e quello di riferimento.*
  - **Precisione (0.83)**, **Richiamo (0.51)**, **F1-score (0.63)**
  - Rouge1 ha degli ottimi risultati. questo ci fa capire che l'approccio e l'oracolo generato da ChatGPT sono ben coperti, ovvero hanno una gran mole di parole uguali.
- **Rouge2**: *Misura il match di bi-grammi (coppie di parole consecutive) tra i testi.*
  - **Precisione (0.45)**, **Richiamo (0.28)**, **F1-score (0.34)**
  - Rouge2 ha risultati più scarsi rispetto a Rouge1 quindi modello è meno efficace nel catturare la struttura sintattica o le relazioni tra le parole rispetto al contenuto lessicale.
- **RougeL**: *Misura il match della* ***longest common subsequence (LCS)****, tenendo conto della struttura sequenziale.*
-
  - **Precisione (0.60)**, **Richiamo (0.37)**, **F1-score (0.47)**
  - RougeL ha sufficienti risultati, le user stories generate mantengono una buona somiglianza con testo di riferimento, ma c'è ancora spazio per migliorare la fedeltà strutturale.
- **Bleu 37%**
- **Meteor 30%**

Per il 34° documento ma tramite l'oracolo manuale:

- **Rouge1**
  - **Precisione (0.83)**, **Richiamo (0.81)**, **F1-score (0.82)**
- **Rouge2**
  - **Precisione (0.63)** , **Richiamo (0.62)**, **F1-score (0.62)**

- **RougeL**
  - **Precisione (0.74)** , **Richiamo (0.72)**, **F1-score (0.73)**
- **Bleu 70%**
- **Meteor 60%**

Rouge risulta molto migliore per l'oracolo manuale, in tutte le tre versioni analizzate, rispetto all'oracolo generato da ChatGPT. Probabilmente perché è più somigliante al dataset rispetto all'oracolo generato da ChatGPT. Infatti l'oracolo generato da ChatGPT può introdurre variazioni nello stile o nel contenuto che lo rendono meno allineato al modello generatore. **Sia Bleu che Meteor risultano molto migliori** per l'oracolo manuale rispetto all'oracolo generato da ChatGPT.

Quarantanovesimo documento

49° documento confrontato con l'oracolo generato da ChatGPT ha questi risultati.

- **Rouge1**: *Misura il match di uni grammi (parole singole) tra il testo generato e quello di riferimento.*
  - **Precisione (0.79)**, **Richiamo (0.52)**, **F1-score (0.63)**
- **Rouge2**: *Misura il match di bi grammi (coppie di parole consecutive) tra i testi.*
  - **Precisione (0.48)**, **Richiamo (0.31)** , **F1-score (0.38)**
- **RougeL**: *Misura il match della* ***longest common subsequence (LCS)****, tenendo conto della struttura sequenziale.*
  - **Precisione (0.63)**. **Richiamo (0.41)**, **F1-score (0.50)**
- **Bleu 38%**
- **Meteor 35%**

Per il 49° documento ma tramite l'oracolo manuale:

- **Rouge1**: *Misura il match di uni grammi (parole singole) tra il testo generato e quello di riferimento.*
  - **Precisione (0.97)**, **Richiamo (0.72)** , **F1-score (0.83)**
- **Rouge2**: *Misura il match di bi grammi (coppie di parole consecutive) tra i testi.*
  - **Precisione (0.83)** , **Richiamo (0.61)** , **F1-score (0.70)**
- **RougeL**: *Misura il match della* ***longest common subsequence (LCS)****, tenendo conto della struttura sequenziale.*
  - **Precisione (0.60)**, **Richiamo (0.66)**, **F1-score (0.76)**
- **Bleu 62%**
- **Meteor 58%**

Rouge risulta migliore di poco per l'oracolo manuale, in tutte le tre versioni analizzate, rispetto all'oracolo generato da ChatGPT. Da notare il valore 0.97% di Rouge1 nell'oracolo manuale, che indica che il 97% delle parole dell'approccio sono presenti nell'oracolo manuale. **Sia Bleu che Meteor risultano migliori** per l'oracolo manuale rispetto all'oracolo generato da ChatGPT. Ma non distanziano di molto come invece è successo nel 34° documento l'oracolo di ChatGPT.

Infine chiariamo perché il valore delle metriche degl'oracoli manuale sono migliori rispetto a quelli degl'oracolo generati da ChatGPT.
Ragioni legate alla sua aderenza al contesto specifico delle user stories
Essendo redatto da un umano, l'oracolo manuale segue lo stile, il contenuto e la struttura delle user stories presenti nel dataset, rendendolo più rappresentativo e fedele al dominio di riferimento.

L'oracolo manuale garantisce un punteggio BLEU più alto grazie alla maggiore sovrapposizione di n-grammi. In quanto ChatGPT può introdurre variazioni stilistiche o lessicali meno pertinenti.
METEOR mostra un miglioramento con l'oracolo manuale poiché quest'ultimo cattura con maggiore accuratezza il significato generale e le varianti linguistiche rilevanti per il dominio. Tuttavia, Questa è una delle metriche che pressoché rimane uguale per l'oracolo manuale che per l'oracolo generato da ChatGPT.
ROUGE risulta più alto per l'oracolo manuale, riflettendo una migliore copertura del contenuto. L'oracolo manuale rispecchia il modo in cui un esperto umano scriverebbe una user story.

In generale, quello che possiamo concludere è che le user stories generate dall'approccio hanno valori delle metriche più alti rispetto a quelle dell'oracolo create manualmente, risultano inferiori se confrontate con le use stories generate da ChatGPT. Questo succede per svariati motivi: **Affinità Semantica e Lessicale**: L'oracolo manuale è stato costruito con un maggiore allineamento lessicale e strutturale rispetto ai requisiti di partenza, ottenendo quindi punteggi BLEU, METEOR e ROUGE più elevati. ChatGPT ha introdotto sinonimi o strutture variate che mantengono il significato ma riducono la similarità lessicale. **Uniformità di Stile**: Le user stories dell'oracolo manuale seguono uno stile uniforme, rendendole più facilmente comparabili alle user stories generate. ChatGPT, invece, può introdurre variazioni stilistiche che, pur mantenendo la correttezza semantica, riducono i punteggi di precisione e di recall.

Capitolo 6

Conclusioni

Il lavoro propone un approccio in grado di generare user stories partendo da requisiti scritti in linguaggio naturale.
Sono stati utilizzati diversi dataset e diverse tecniche di *NLP* per ottenere risultati migliori possibili. Al cuore l'approccio propone una combinazione di metodi, *NER, Dependency Parsing, lexname from WordNet* **per gl'aspetti del chi e del cosa**, invece per **l'aspetto del perché** l'utilizzo del *Dependecy Parsing* partendo dall'aspetto del cosa.
La valutazione è eseguita creando due oracoli per ogni dataset utilizzato uno creato manualmente e uno generato da ChatGPT.
I risultati si aggirano al di sopra del **70%** per le metriche di *Bleu e Meteor* e attorno al **75%** per la metrica *Rouge*. Tuttavia come sviluppi futuri si propone:

- **Utilizzo di dataset ulteriori**: per avere la maggiore diversificazione di dati possibile e anche la maggiore quantità di dati possibili.
- **Analisi della soggettività**: mira a identificare e comprendere le opinioni, le emozioni e i punti di vista personali che influenzano l'espressione e l'interpretazione delle informazioni. Che può essere utile nell'identificare l'aspetto del perché.
- **L'integrazione di euristiche**: informazioni problem-specific per migliorare l'estrazione di user stories la precisione e l'affidabilità dei risultati.

Questo link conduce al *repository* **Github** dove sono racchiuse tutte le informazioni principali relative all'approccio proposto.

`https://github.com/AlfonsoAnzelmo35/userStoriesGenerator`

Bibliography

[1] S. Banerjee and A. Lavie. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. 2005. URL: https://aclanthology.org/W05-0909.pdf.

[2] Kent Beck et al. *Manifesto for Agile Software Development*. https://agilemanifesto.org. (citato a pagina 7). 2001.

[3] W. Dahhane et al. *An Automated Object-Based Approach to Transforming Requirements to Class Diagrams*. 2014 2nd World Conference on Complex Systems (WCCS). 2015. DOI: 10.1109/ICoCS.2014.7060906. URL: https://doi.org/10.1109/ICoCS.2014.7060906.

[4] M. Elallaoui, K. Nafil, and R. Touahni. *Automatic Generation of UML Sequence Diagrams from User Stories in Scrum Process*. 2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA). 2015. DOI: 10.1109/SITA.2015.7358415. URL: https://doi.org/10.1109/SITA.2015.7358415.

[5] M. Elallaoui, K. Nafil, and R. Touahni. *Automatic Transformation of User Stories into UML Use Case Diagrams Using NLP Techniques*. Procedia Computer Science. 2018. DOI: 10.1016/j.procs.2018.04.010. URL: https://doi.org/10.1016/j.procs.2018.04.010.

[6] A. Gupta, G. Poels, and P. Bera. *Creation of Multiple Conceptual Models from User Stories – A Natural Language Processing Approach*. Advances in Conceptual Modeling. ER 2019. Lecture Notes in Computer Science. 2019. DOI: 10.1007/978-3-030-34146-6_5. URL: https://doi.org/10.1007/978-3-030-34146-6_5.

[7] A. Henriksson and J. Zdravkovic. *A Data-Driven Framework for Automated Requirements Elicitation from Heterogeneous Digital Sources*. Lecture Notes in Business Information Processing. 2020. DOI: 10.1007/978-3-030-63479-7_24. URL: https://doi.org/10.1007/978-3-030-63479-7_24.

[8] P. Jin et al. *News Feature Extraction for Events on Social Network Platforms*. 26th International World Wide Web Conference 2017, WWW 2017 Companion. 2019. DOI: 10.1145/3041021.3054151. URL: https://doi.org/10.1145/3041021.3054151.

[9] D. Jurafsky and James H. Martin. *Dependency Parsing*. Speech and Language Processing, 3rd ed., 2020 draft. 2020.

[10] C.-Y. Lin. *ROUGE: A Package for Automatic Evaluation of Summaries*. Proceedings of the Workshop on Text Summarization Branches Out, 2004. 2004. URL: https://aclanthology.org/W04-1013.pdf.

[11] Meandor. *User Story Generator: A LLM-Based Approach*. GitHub Repository. 2024. URL: https://github.com/meandor/user-story-generator.

[12] George A. Miller. *WordNet: A Lexical Database for English*. Communications of the ACM. 1969.

[13] S. Nasiri et al. *Towards a Generation of Class Diagram from User Stories in Agile Methods*. Procedia Computer Science. 2020. DOI: 10.1016/j.procs.2020.03.148. URL: https://doi.org/10.1016/j.procs.2020.03.148.

[14] Cuong D. Nguyen et al. *Generating User Stories in Groups*. Proceedings of the 15th International Workshop on Groupware: Design, Implementation, and Use, CRIWG 2009, Peso da Régua, Douro, Portugal, September 13-17, 2009. Sept. 2024. DOI: 10.1007/978-3-642-04216-4_30. URL: https://doi.org/10.1007/978-3-642-04216-4_30.

[15] K. Papineni et al. *BLEU: a Method for Automatic Evaluation of Machine Translation*. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), 2002. 2002. URL: https://aclanthology.org/P02-1040.pdf.

[16] Tajmilur Rahman and Yuecai Zhu. *Automated User Story Generation with Test Case Specification Using Large Language Model*. Apr. 2024.

[17] P. Rodeghero et al. *Detecting User Story Information in Developer-Client Conversations to Generate Extractive Summaries*. Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). 2017. DOI: 10.1109/ICSE.2017.13. URL: https://doi.org/10.1109/ICSE.2017.13.

[18] C. A. dos Santos, K. Bouchard, and B. Minetto Napoleão. *Automatic User Story Generation: A Comprehensive Systematic Literature Review*. International Journal of Data Science and Analytics. 2024. DOI: 10.1007/s41060-024-00567-0. URL: https://doi.org/10.1007/s41060-024-00567-0.

[19] Y. Wautelet, S. Heng, D. Hintea, et al. *Bridging User Story Sets with the Use Case Model*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2016. DOI: 10.1007/978-3-319-47717-6_11. URL: https://doi.org/10.1007/978-3-319-47717-6_11.

Ringraziamenti

Desidero esprimere la mia profonda gratitudine a tutte le persone che hanno contribuito alla realizzazione di questa tesi. In primo luogo, un ringraziamento va *al mio relatore, Carmine Gravino e il correlatore Francesco Casillo* per il loro supporto, la guida costante e la disponibilità durante tutto il percorso.
I consigli forniti sono stati fondamentali per lo sviluppo del mio lavoro.

Ringrazio *mio padre*, per il supporto incondizionato e l'incoraggiamento che mi ha sempre dato, anche nei momenti più difficili.

Un grazie di cuore anche *ai miei amici*, che mi hanno sostenuto e spronato, e che hanno condiviso con me tanto le gioie quanto le sfide di questo percorso universitario. Il loro appoggio è stato per me di grande conforto e motivazione.

Grazie a *me stesso*, per aver creduto in questo progetto dall'inizio fino alla fine, per aver affrontato le sfide con determinazione, per non essermi arreso nei momenti difficili. Questo lavoro è anche il risultato di tutto il tempo, l'energia e la passione che ho investito.

A tutti, grazie di cuore.