UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Code4Code: tecniche di Intelligenza Artificiale per il suggerimento di Tecnologie Software

RELATORE

Prof. **Fabio Palomba**

Università degli studi di Salerno

CANDIDATO

Vincenzo Emanuele Martone

Matricola: 0512105758

Anno Accademico 2020-2021

"Perché il miglior risultato si ottiene quando ogni componente del gruppo farà ciò che è meglio per sé...

e per il gruppo..."

Sommario

Il contesto applicativo della Tesi è incentrato sullo studio e sull'applicazione di metodologie di Intelligenza Artificiale nell'ambito di una piattaforma di collaborazione. L'obiettivo principale della tesi sviluppata è quello di fornire un insieme di strumenti di Intelligenza Artificiale da integrare in un'ipotetica piattaforma di collaborazione. Lo scopo di tale piattaforma è quello di mettere in contatto persone intenzionate ad imparare nuove tecnologie in ambito informatico, in modo che possano iniziare una collaborazione che consiste nello scambio di conoscenze relative alle tecnologie che desiderano apprendere. Lo sviluppo della Tesi non si concentra sulla creazione della piattaforma completa, bensì sullo studio, sul confronto, sulla progettazione e sull'implementazione di algoritmi di Intelligenza Artificiale che possano assistere gli utenti nell'utilizzo della piattaforma stessa. La Tesi fornisce, dunque, un prototipo denominato Code4Code che mostra il funzionamento degli agenti di Intelligenza Artificiale sviluppati. Il problema principale che è stato risolto mediante l'utilizzo dell'Intelligenza Artificiale è quello relativo al suggerimento di tecnologie da imparare sulla base di quelle già conosciute dall'utente; a tale scopo all'utente vengono consigliate sia tecnologie simili a quelle che già conosce, sia tecnologie spesso utilizzate con quelle che già conosce.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto applicativo

Il lavoro proposto nell'ambito della presente Tesi si inserisce nel contesto applicativo delle piattaforme collaborative ed in particolar modo nell'ambito delle piattaforme che mirano all'acquisizione e al miglioramento delle competenze informatiche degli utenti ad esse iscritti mediante una collaborazione tra gli stessi. Al fine di contestualizzare il lavoro svolto è stato in parte progettato e sviluppato un prototipo di piattaforma collaborativa, denominato *Code4Code*, che mette a disposizione degli utenti un insieme di strumenti, tra cui l'Intelligenza Artificiale, che li assiste sia in fase di primo approccio alla piattaforma che in fasi più avanzate.

1.2 Obiettivi e risultati

Sebbene l'idea iniziale del lavoro si focalizzasse sulla creazione completa della piattaforma *Code4Code*, nel corso dello sviluppo è stata posta una maggiore attenzione verso gli strumenti di supporto che essa dovrebbe fornire agli utenti che ne usufruiscono, in particolar modo alle tecniche di Intelligenza Artificiale utili a migliorare l'esperienza degli utenti. L'obiettivo finale del lavoro svolto è lo sviluppo di due micro-agenti di Intelligenza Artificiale che compongono un unico macro-agente che consiglia Linguaggi di Programmazione, Frameworks e Librerie da studiare sulla base di quelli conosciuti dall'utente che ne usufruisce. Il primo micro-

agente consiglia Linguaggi di Programmazione a partire dai Linguaggi di Programmazione conosciuti dall'utente, mentre il secondo micro-agente si occupa di suggerire Frameworks e Librerie sulla base dei Linguaggi di Programmazione, dei Frameworks e delle Librerie conosciuti dall'utente. La suddivisione del macro-agente in due micro-agenti, nonostante la loro apparente similitudine, è stata dettata dalle differenze tra le tecniche di Intelligenza Artificiale impiegate nella realizzazione dei due micro-agenti: il primo micro-agente, infatti, sfrutta le *Association Rules*, mentre il secondo si serve di tecniche di *Natural Language Processing*. L'utilizzo dell'Intelligenza Artificiale verrà debitamente trattato nel corso dei successivi capitoli.
Dopo aver esaminato il dominio del problema e stabilito quali tecniche di Intelligenza Artificiale utilizzare, sono stati studiati i fondamenti teorici di tali tecniche che hanno reso possibile, in fase implementativa, un utilizzo consapevole delle Librerie adoperate per la creazione degli Agenti Intelligenti di interesse. Durante la fase implementativa ci si è prima di tutto soffermati sulla creazione dei singoli micro-agenti, successivamente è stato effettuato un lavoro di integrazione tra i micro-agenti stessi ed infine è stata aggiunta un'interfaccia utente per rendere possibile un utilizzo agevole di quanto implementato.
La fase implementativa ha richiesto numerose revisioni del DataSet e dei parametri di addestramento dei micro-agenti fino al raggiungimento di un risultato che, rapportato alle caratteristiche dei DataSet in esame e alla conoscenza dell'ambito delle attuali Tecnologie Software utilizzate, fosse intuitivamente sensato e contenesse il minor numero possibile di *outliers*. Il risultato finale di tale lavoro di Tesi risulta, dunque, essere un prototipo della piattaforma Web *Code4Code* tramite il quale si può interagire con l'Intelligenza Artificiale sviluppata.

1.3 Struttura della Tesi

La trattazione del lavoro di Tesi verrà suddivisa in cinque capitoli di cui viene data, di seguito, una breve descrizione:

- Il Primo Capitolo introduce il contesto applicativo, gli obiettivi e i risultati del lavoro di Tesi svolto
- Il Secondo Capitolo analizza le caratteristiche fondamentali di alcune piattaforme collaborative considerate particolarmente rilevanti e affini alla piattaforma *Code4Code*

- Il Terzo Capitolo rappresenta il cuore della Tesi in quanto descrive nel dettaglio le tecniche di Intelligenza Artificiale adoperate motivando le scelte che sono state effettuate a tal proposito
- Il Quarto Capitolo illustra la progettazione e l'architettura della piattaforma Web *Code4Code* presentando le tecnologie adoperate per l'effettiva creazione del prototipo implementato
- Il Quinto Capitolo conclude la trattazione del lavoro di Tesi presentando considerazioni di carattere generale sul lavoro svolto e sugli eventuali sviluppi futuri

CAPITOLO 2

Piattaforme collaborative e di apprendimento

Questo capitolo fa una breve panoramica sulle piattaforme collaborative e di apprendimento considerate affini alla piattaforma *Code4Code*. Tale trattazione fornisce un'idea della motivazione per la quale si è ritenuto utile progettare la piattaforma e soprattutto formulare e risolvere problemi ad essa inerenti.

2.1 Differenza tra piattaforme collaborative e di apprendimento

I concetti di piattaforma collaborativa e di piattaforma di apprendimento sono ben distinti. Una piattaforma collaborativa è, in generale, un tipo di piattaforma in cui gli utenti collaborano tra loro al fine di raggiungere uno scopo comune, mentre una piattaforma di apprendimento è un tipo di piattaforma che mira ad accrescere le conoscenze e le competenze di chi ne usufruisce. La natura del secondo tipo di piattaforma non è intrinsecamente legata ad un approccio basato sulla cooperazione tra gli utenti che vi partecipano, infatti nella maggior parte dei casi l'apprendimento risulta essere unidirezionale da insegnante ad allievo. Nell'ambito delle piattaforme di apprendimento vengono, spesso, inserite funzionalità di comunicazione tra allievo ed insegnante al fine di rendere agevole l'apprendimento tramite un confronto grazie al quale l'allievo può esporre i propri dubbi all'insegnante. Nonostante ciò, difficilmente il confronto tra le due parti avviene in tempo reale, e spesso si ricorre ad una sezione dedicata alle domande degli allievi a cui l'insegnante può rispondere in differita. Una piattaforma di collaborazione, d'altro canto, non è necessariamente legata all'apprendimento ma generalmente fornisce un insieme di strumenti per facilitare la comunicazione e la collaborazione in tempo reale.

2.2 Esempi di piattaforme collaborative

Al giorno d'oggi esistono diverse piattaforme collaborative che consentono a chi ne usufruisce di collaborare insieme ad altri utenti per raggiungere uno scopo comune. Di seguito verranno elencate alcune piattaforme e ne verranno brevemente analizzate le caratteristiche fondamentali.

- **GitHub**: oltre a fungere da hosting per i progetti software degli utenti che ne usufruiscono, fornisce strumenti per la gestione del *Version Control* che risulta essere di fondamentale importanza soprattutto nel caso di progetti di gruppo in cui gli sviluppatori devono collaborare tra loro.

- **Slack**: progettata per la collaborazione aziendale, fornisce un insieme di strumenti utili alla comunicazione istantanea tra gli utenti stessi. In particolare grazie a Slack è possibile avviare chiamate e videochiamate con funzionalità di condivisione schermo, usufruire di una chat in tempo reale e condividere file grazie all'integrazione con alcune piattaforme di condivisione come ad esempio *Google Drive*, *Dropbox e Microsoft OneDrive*.

2.3 Esempi di piattaforme di apprendimento

La progettazione di una piattaforma di apprendimento è un lavoro che determina in gran parte l'efficacia dell'apprendimento degli utenti che ne usufruiscono. Di seguito verranno esaminate alcune piattaforme con i relativi approcci alla progettazione dell'apprendimento.

- **MasterClass**: mediante il pagamento di un abbonamento, consente agli utenti di accedere ad un insieme di corsi che abbracciano diversi ambiti, tra cui il business, la musica, la scrittura e la tecnologia. I corsi sono tenuti da esperti del settore, grazie ai quali la qualità dei corsi risulta essere elevata, tuttavia la piattaforma non garantisce un'interazione tra studenti ed insegnanti, sebbene non manchino alcune occasioni in cui gli studenti possono sottoporre il proprio lavoro ad una revisione da parte degli insegnanti.

- **Udemy**: consente agli utenti di usufruire di corsi (sia gratuiti che a pagamento) erogati da altri utenti della piattaforma. Ogni corso è suddiviso in lezioni, in corrispondenza di ognuna delle quali è indicato l'argomento affrontato mediante un breve titolo. È, inoltre, disponibile una sezione Q&A in cui gli studenti del corso possono porre domande all'insegnate e chiunque può partecipare alla discussione. È disponibile una funzionalità di Messaggi Diretti tra studenti ed insegnanti solo per i corsi a pagamento al fine di ottenere delucidazioni sugli argomenti trattati nei corsi e per dare feedback. Infine gli insegnanti possono pubblicare degli annunci volti alla distribuzione di materiali gratuiti relativi ai loro corsi.

2.4 Tandem: una piattaforma che unisce collaborazione ed apprendimento

La piattaforma *Code4Code* si ispira concettualmente ad una piattaforma che unisce collaborazione ed apprendimento: **Tandem**. Tandem è un'App sviluppata per Android e iOS che serve a mettere in contatto utenti che vogliono imparare nuove lingue. La piattaforma associa ad un utente che vuole imparare una lingua A e conosce una lingua B un utente che conosce la lingua A e vuole imparare la lingua B. I due utenti, una volta in contatto, possono dialogare effettuando un lavoro di mutua correzione, senza instaurare un rapporto insegnante-studente. Tale approccio è detto *Tandem Language Learning* [1] ed è il principio cardine della piattaforma *Code4Code*.

Tandem fornisce come strumenti di supporto per la comunicazione chiamate, videochiamate e chat in tempo reale, in cui è possibile anche inviare messaggi vocali (fondamentali per esercitare la pronuncia della lingua).

2.5 Considerazioni sulle piattaforme analizzate

Dalla breve analisi effettuata, risulta evidente che in molti casi ci sia un compromesso tra qualità dei corsi erogati ed interazione con gli insegnanti. L'approccio utilizzato da *Tandem* fa emergere, in maniera evidente, tale compromesso, in quanto il generico interlocutore potrebbe non avere un background educazionale, per cui l'esperienza dal punto di vista pratico potrebbe risultare meno scorrevole di un'esperienza di insegnamento tradizionale come quella proposta da alcune delle piattaforme sopracitate in cui si partecipa a corsi pre-registrati. La problematica appena trattata rappresenta uno degli aspetti critici del metodo *Tandem Language Learning*. D'altro canto tale approccio favorisce al massimo l'interazione tra le parti che vogliono apprendere e non rende necessario il pagamento di una somma di denaro, in quanto la moneta di scambio per la conoscenza è la conoscenza stessa. La piattaforma *Code4Code* propone un tipo di approccio all'apprendimento basato sulla collaborazione: se un utente X conosce la Tecnologia A e vuole studiare la Tecnologia B, mentre un utente Y conosce la Tecnologia B e vuole studiare la Tecnologia A, potranno mettersi in contatto e, tramite la piattaforma, scambiarsi conoscenze relative alle Tecnologie mediante lezioni in tempo reale tenute dagli utenti stessi, chat ed esercizi.

CAPITOLO 3

Background sull'Intelligenza Artificiale utilizzata

Come anticipato nei precedenti capitoli, è stata posta particolare attenzione alla progettazione e all'implementazione degli strumenti di Intelligenza Artificiale che assistono gli utenti nell'utilizzo della piattaforma. Il presente capitolo risulta, dunque, essere il cuore del lavoro di Tesi in quanto tratta e approfondisce diversi aspetti relativi all'Intelligenza Artificiale.

3.1 Motivazioni all'utilizzo dell'Intelligenza Artificiale

L'utilizzo dell'Intelligenza Artificiale apre le porte all'implementazione di funzionalità che arricchiscono la piattaforma all'interno della quale vengono integrate. Nel caso della piattaforma *Code4Code* ci sono due motivazioni principali che portano all'esigenza di utilizzare l'Intelligenza Artificiale:

- **Suggerimento di Tecnologie Software**: Quando un utente si iscrive alla piattaforma potrebbe non avere le idee chiare sulle Tecnologie da apprendere: un apposito modulo di Intelligenza Artificiale lo assisterà consigliandogli Tecnologie da imparare sulla base di quelle che già conosce.
- **Suggerimento di utenti con cui collaborare**: L'anima della piattaforma è la collaborazione, ma la disponibilità di un utente può variare in base a diversi fattori come il numero di utenti con cui collabora in un certo momento. Mediante un algoritmo di Intelligenza Artificiale è possibile individuare un insieme di utenti con cui risulta più conveniente iniziare una collaborazione evitando lunghe ed inconcludenti ricerche nel momento in cui si è in procinto di iniziare una collaborazione.

Tra queste due problematiche è stata debitamente affrontata, progettata ed implementata solo la prima, mentre della seconda verrà fornita solo un'idea di progettazione ed implementazione.

3.2 Suggerimento di Tecnologie Software

Quando un utente si iscrive alla piattaforma seleziona le Tecnologie Software conosciute, sulla base delle quali gliene vengono suggerite altre da imparare. Una Tecnologia Software, in questo specifico caso, può essere un Linguaggio [1], un Framework o una Libreria, pertanto il problema in questione è stato suddiviso in due parti:

- Suggerimento di Linguaggi a partire da quelli conosciuti, mediante *Association Rules*.
- Suggerimento di Librerie e Frameworks a partire dai Linguaggi, dalle Librerie e dai Frameworks conosciuti, mediante *Natural Language Processing*.

[1] Nel corso della trattazione, a scopo esemplificativo, si utilizzerà il termine *Linguaggio* per riferirsi, in generale, alla classe dei Linguaggi formali che comprende Linguaggi di Programmazione, Markup, Scripting, Stylesheet [...]

L'esigenza di suddividere l'Agente di Intelligenza Artificiale in due micro-agenti è stata dettata dalla eterogeneità dei DataSet disponibili a causa dei quali non è stato possibile pensare ad una soluzione unificata per risolvere il problema. Nei paragrafi successivi verranno trattati gli aspetti teorici ed implementativi delle tecniche di Intelligenza Artificiale utilizzate con riferimenti ai relativi DataSet.

3.2.1 Suggerimento di Linguaggi

Suggerire un Linguaggio di Programmazione ad un utente in base a quelli che conosce, significa proporre all'utente Linguaggi che siano in qualche modo correlati a quelli di sua conoscenza. È sensato considerare due principali tipi di correlazione: *similarità* e *complementarietà*.

- Due Linguaggi vengono considerati simili se hanno un certo numero di caratteristiche in comune.
- Due Linguaggi vengono considerati complementari se vengono spesso utilizzati insieme nell'ambito di Progetti Software.

Similarità tra Linguaggi

Stabiliti i parametri secondo i quali due Linguaggi sono definibili simili, risulta semplice sviluppare un Algoritmo che, presi in input i Linguaggi conosciuti dall'utente, restituisca quelli che hanno più caratteristiche possibili in comune con i Linguaggi input. Le caratteristiche in esame sono: tipo del Linguaggio (Programmazione, Markup, Scripting, Query, Stylesheet [...]), paradigmi (Object Oriented, funzionale, procedurale [...]) e sistema di tipizzazione (nessuna tipizzazione, tipizzazione forte, tipizzazione debole). Supponendo, ad esempio, che il Linguaggio input sia *Java* che è un Linguaggio di Programmazione, il cui paradigma è *Object Oriented*, con sistema di tipizzazione forte, l'Algoritmo suggerirà sicuramente *Kotlin* (le cui caratteristiche corrispondono quasi esattamente a quelle di *Java*) in misura maggiore rispetto ad un Linguaggio di Markup come ad esempio *HTML* o ad un Linguaggio di Scripting come ad esempio *Lua*. La metrica adottata per quantificare quanto un Linguaggio sia effettivamente consigliato ad un utente è esattamente il numero di caratteristiche che il Linguaggio input e gli altri Linguaggi hanno in comune. L'Algoritmo non restituisce un unico Linguaggio, bensì un insieme di Linguaggi aventi un certo numero di caratteristiche in comune con almeno uno dei Linguaggi input. Il numero massimo di Linguaggi da consigliare all'utente è stato scelto in seguito ad alcuni test per individuare la

soluzione migliore: se la dimensione n dell'insieme dei Linguaggi restituiti dall'Algoritmo supera le 5 unità, si considerano solo i primi $\frac{n}{2}$ Linguaggi più simili, in caso contrario si considerano tutti i Linguaggi restituiti.
Dal momento che i dati da esaminare riguardano le caratteristiche dei vari Linguaggi, è stato ottenuto un semplice DataSet prelevando i dati da siti Web come *Wikipedia*. L'Algoritmo vero e proprio è stato sviluppato in *Java* senza l'ausilio di particolari Framework o Librerie.
L'approccio appena descritto non rappresenta una vera e propria applicazione di Intelligenza Artificiale, ma solo una base esemplificativa da cui partire per poter stabilire, in maniera intuitiva, la similarità tra due Linguaggi. L'approccio basato sulla complementarietà, descritto di seguito, costituisce un esempio di applicazione dell'Intelligenza Artificiale sicuramente più articolato rispetto a quello appena presentato.

Complementarietà tra Linguaggi

Focalizzarsi sulla complementarietà tra i Linguaggi consente, a chi vuole imparare, di acquisire competenze a 360 gradi. Come anticipato in precedenza, due Linguaggi sono considerati complementari se vengono spesso utilizzati insieme nell'ambito di Progetti Software; per indagare sulla complementarietà tra i Linguaggi, è necessario disporre sia di un insieme di Progetti Software di dimensione sufficientemente elevata, al fine di poter effettuare considerazioni reali sull'associazione tra i Linguaggi, che di una tecnica per interpretare e dare un senso ai dati relativi ai progetti. Al fine di ottenere un insieme considerevole di Progetti Software è stato utilizzato *GitHub* ed in particolar modo è stato fatto riferimento alle Repositories pubbliche. *GitHub* mette a disposizione API pubbliche e gratuite che assolvono a diversi scopi, tra cui cercare i progetti ed ottenere informazioni relative ad essi. Tuttavia, per semplificare il lavoro e migliorare la qualità del DataSet è stato utilizzato un tool online denominato *SEART GitHub Search Engine* [2] che consente, mediante un'interfaccia molto intuitiva, di filtrare i risultati secondo determinate caratteristiche della Repository come: numero di commits, numero di stars, numero di releases, range di data. Effettuare un filtraggio del genere consente di evitare Repositories non rilevanti (Repositories di prova, piccoli progetti di esempio, esercizi didattici) focalizzando l'attenzione solo su quelle che possano in qualche modo essere rilevanti ai fini della trattazione del problema in questione. I parametri utilizzati per filtrare le Repositories sono stati scelti in maniera intuitiva e sono stati effettuati dei test per confrontarli tra loro. Il risultato migliore è stato ottenuto prelevando tutte le Repositories con almeno 50 commit e 100 stars. Il Tool sopracitato ha prodotto un file output in formato *JSON*, del quale è stato effettuato un parsing utilizzando uno script scritto in *Python* tramite il

quale sono state estratte le informazioni rilevanti per la risoluzione del problema in questione, ossia i Linguaggi impiegati nell'ambito delle Repositories con la relativa quantità di byte di codice utilizzato. È stato, dunque, creato un file in cui su ogni riga è riportata la lista di Linguaggi utilizzati in una determinata Repository, ciascuno dei quali è seguito dalla quantità di byte di codice scritto in quel Linguaggio. Ogni riga è, dunque, associata ad una specifica Repository ed in totale sono stati estratti i dati relativi a esattamente 51135 Repositories.
Il file appena descritto funge da DataSet per l'Algoritmo che cerca la correlazione tra i Linguaggi basandosi sulla complementarietà tra gli stessi. I dati vengono interpretati mediante le *Association Rules* [3], un metodo che serve ad estrarre relazioni nascoste da una grande quantità di dati. Nel caso in esame, la relazione da ricercare nei dati è la presenza simultanea di Linguaggi nella stessa Repository; le Association Rules fanno proprio questo tipo di lavoro considerando tre misure di probabilità denominate *Support, Confidence e Lift*.

- Il **Support** è una misura che indica la frequenza in cui un determinato Linguaggio compare nei dati in esame. Il Support di un linguaggio L in un DataSet contenente N Repositories si calcola nel seguente modo:

$$Support(L) = \frac{Freq(L)}{N} \tag{3.2.1}$$

- La **Confidence** è una misura che indica la frequenza con cui un Linguaggio L_2 è presente nelle Repositories che contengono un Linguaggio L_1. La formula per calcolare la Confidence è la seguente:

$$Confidence(L_1 \rightarrow L_2) = \frac{Support(L_1 \cap L_2)}{Support(L_1)} \tag{3.2.2}$$

- Il **Lift** è una misura simile alla *Confidence*, in quanto indica la frequenza con cui un Linguaggio L_2 è presente nelle Repositories che contengono un Linguaggio L_1, tuttavia tiene conto anche della popolarità del Linguaggio L_2. Dal punto di vista matematico, si calcola nel seguente modo:

$$Lift(L_2) = \frac{Support(L_1 \cap L_2)}{Support(L_1) \times Support(L_2)} \tag{3.2.3}$$

L'algoritmo Apriori sfrutta queste tre misure di probabilità al fine di generare le Association Rules. Tralasciando i dettagli implementativi dell'Algoritmo Aprori (che possono essere approfonditi alla fonte indicata [3]), se si considera un sottoinsieme di k Linguaggi del DataSet sarà possibile generare $2^k - 2$ Association Rules, ciascuna delle quali avrà la seguente forma:

$$\{L_1, L_2, \ldots, L_i\} \rightarrow \{L_{i+1}, \ldots, L_k\} \tag{3.2.4}$$

Ad ogni Association Rule sono associate la Confidence e il Lift relativi ai Linguaggi $\{L_{i+1}, \ldots, L_k\}$ (tale termine della Regola è detto *conseguente*) considerando le Repository contenenti i Linguaggi $\{L_1, L_2, \ldots, L_i\}$ (tale termine della Regola è detto *antecedente*), e ad ogni gruppo di Association Rules relativo ad un sottoinsieme di k Linguaggi è associato il Support dei Linguaggi stessi. L'Algoritmo Apriori non è stato implementato da zero, ma è stata utilizzata una sua implementazione in Python denominata *Apyori* [4] che consente di eseguire l'algoritmo in maniera agevole a partire dal DataSet di Repositories descritto in precedenza e di impostare i parametri di funzionamento dell'Algoritmo come la soglia di Support, Confidence e Lift minime dei Linguaggi che l'Algoritmo deve considerare e la dimensione massima dei sottoinsiemi che devono essere usati dall'Algoritmo per generare le Association Rules. A tal proposito, i parametri che sono stati scelti riguardano la soglia di Support e Confidence poste rispettivamente a 0.0045 e 0.10.
Quanto descritto fino ad adesso non dipende dall'input dell'utente e fa, dunque, parte di un lavoro che viene svolto offline e che funge da base per l'Algoritmo vero e proprio che a partire dai Linguaggi conosciuti dall'utente ne suggerisce altri. Tale Algoritmo, infatti, preleva tutte le Association Rules che come antecedente contengono i Linguaggi input e considera i Linguaggi contenuti nel termine conseguente; la lista di questi Linguaggi verrà chiamata *Lista di Linguaggi candidati*. Ad ogni Linguaggio facente parte della *Lista di Linguaggi candidati*, l'Algoritmo associa la Confidence della Regola a cui appartiene (in caso di Linguaggi duplicati considera quello appartenente alla Regola con Confidence maggiore) e consiglia alcuni di questi Linguaggi agli utenti. Non vengono consigliati tutti i Linguaggi candidati per due motivi principali:

- Il primo motivo riguarda aspetti puramente legati all'usabilità della piattaforma, per cui non si consigliano troppi Linguaggi all'utente per evitare di confonderlo. A tale scopo dopo aver ottenuto la lista dei Linguaggi candidati, l'Algoritmo controlla la sua dimensione n e se questa è inferiore o uguale a 10, non scarta alcun Linguaggio, se è compresa tra 10 e 19, conserva solo gli $n/2$ Linguaggi a cui è associata la Confidence più elevata, se supera 20 vengono conservati solo gli $n/4$ Linguaggi con la Confidence più elevata.

- Il secondo motivo riguarda la presenza di Linguaggi nelle Repositories che pur essendo molto frequenti, non risultano rilevanti. Ad esempio, in molti Progetti sono presenti brevi script scritti in Bash utilizzati per motivi riguardanti la configurazione e l'esecuzione del Progetto; non sarebbe, dunque, corretto consigliare Bash unicamente perché

molto usato insieme ad un determinato Linguaggio input. Risulta, dunque, necessario svolgere un lavoro di tipo statistico su ogni Linguaggio candidato per determinare se, nelle Repositories in cui viene utilizzato insieme ad almeno uno dei Linguaggi input dell'utente, ha una percentuale di utilizzo almeno pari ad una determinata soglia. La percentuale di utilizzo di un Linguaggio in una Repository può essere facilmente ricavata a partire dalla quantità di byte di codice scritto in quel Linguaggio; ai fini dell'Algoritmo viene effettuata una media delle percentuali di codice scritto nel Linguaggio candidato considerato: verranno scartati tutti i Linguaggi la cui media di percentuale di codice impiegato nelle Repositories risulta essere inferiore al 3.5% (tale soglia è stata scelta dopo aver effettuato diversi test e confrontato tra loro i risultati).

3.2.2 Suggerimento di Frameworks e Librerie: Natural Language Processing

Suggerire Frameworks e Librerie ad un utente a partire dai Linguaggi, dai Frameworks e dalle Librerie che conosce, si è rivelato essere un lavoro meno agevole del precedente, in quanto le API di GitHub consentono di ricavare i Linguaggi adoperati in una Repository, ma non i Frameworks e le Librerie. È stata, dunque, adottata una tecnica di Intelligenza Artificiale diversa dalla precedente, al fine di sfruttare al meglio il DataSet in possesso; tale DataSet contiene:

- Descrizioni in lingua inglese di Linguaggi, Framework e Librerie estratti tramite uno scraper scritto in *JavaScript* dalla pagina di GitHub relativa ai Topic (tra cui Frameworks e Librerie) relativi alle Repositories.
- Lista di Tag relativi a diverse Repositories estratti da GitHub tramite uno scraper appositamente scritto in *JavaScript*. A differenza del caso dei Linguaggi, tuttavia, i tag sono inseriti dagli utenti e non sempre alla stessa Libreria/Framework corrisponde la stessa stringa, in quanto spesso vengono utilizzati nomi alternativi ed abbreviazioni (ciò non accade nel caso dei Linguaggi in quanto vengono automaticamente impostati da GitHub).

Vista l'irregolarità e l'informalità del DataSet in esame, si è deciso di ricorrere a tecniche di *Natural Language Processing* al fine di poter interpretare e dare un senso ai dati ed estrarre informazioni utili sulla correlazione tra i Frameworks, sulla correlazione tra le Librerie e sulla correlazione tra Frameworks e Librerie. L'Algoritmo di NLP utilizzato è *Word2Vec* [5] che è basato su una Rete Neurale con un singolo livello nascosto il cui scopo è quello di calcolare un insieme di vettori al fine di valutare la similarità tra le parole facenti parte

del DataSet. Quando si parla di *Doc2Vec* è necessario dettagliare il concetto di DataSet mediante termini specifici: in primo luogo l'unità di base del DataSet è il *documento*, definito come testo di qualsiasi tipo (una breve descrizione, un paper, un libro...) e che nel caso in questione risulta essere la descrizione di una Libreria/Framework oppure l'insieme di tag relativi ad una singola Repository. Un insieme di *documenti* forma un *Corpus*, che funge da input per l'Algoritmo e dal quale si estraggono le parole al fine di costruire un dizionario. Come anticipato, l'Algoritmo *Word2Vec* fa uso di una rete neurale, nella quale è possibile individuare:

- Un livello input che trasforma le parole in vettori: in primo luogo l'Algoritmo crea un dizionario contenente tutte le n parole del *Corpus* e alla parola i-esima associa un vettore di dimensione $1 \times n$ composto da valori tutti pari a 0 e da un 1 in posizione i-esima; tali vettori sono chiamati *one-hot vector*.

- Un livello nascosto che contiene n vettori, ciascuno di dimensione m; la scelta di m è, generalmente, problem specific e rappresenta il numero di *features* delle parole nel dizionario considerato. Lo scopo della fase di addestramento è quello di apprendere i valori delle componenti dei vettori dello strato nascosto della rete, in quanto rappresentano le caratteristiche vere e proprie delle parole del dizionario; tali vettori sono denominati *embeddings* e il livello nascosto della rete contiene una matrice $n \times m$ di tutti gli *embeddings* relativi alle parole del dizionario.

- Un livello output nel quale viene effettuato il prodotto tra l'*embedding* della parola input e la matrice che contiene tutti gli *embeddings* trasposta, e successivamente al risultato viene applicata la funzione di regressione Softmax, ottenendo un vettore contenente la probabilità che ogni parola del dizionario si trovi nel contesto della parola input.

L'addestramento della rete neurale consiste in diverse fasi: in primo luogo vi è la definizione di un valore intero detto *window size* che rappresenta il numero di parole da considerare a sinistra e a destra di una parola w all'interno di una frase del *Corpus*. Tali parole saranno considerate appartenenti allo stesso contesto di w e sono dette *parole di contesto*. Vengono, a questo punto, costruiti dei campioni di addestramento ossia delle coppie *(w, x)* dove w è una parola del dizionario e x è una sua parola di contesto; generate tutte queste coppie, viene data in input alla rete neurale la parola w, ottenendo, in questo modo, un vettore output v contenente le probabilità che ogni parola del dizionario si trovi nel contesto di w. A questo punto si confronta il vettore v con il vettore x (che essendo un *one-hot vector* conterrà un 1 nella

posizione relativa alla parola di contesto) e si aggiornano gli *embeddings* del livello nascosto della rete utilizzando il metodo della discesa del gradiente. Tale tipo di addestramento fa sì che i pesi del livello intermedio vadano a modificarsi gradualmente sulla base delle coppie di parole vicine che vengono prelevate dal *Corpus*: al termine dell'addestramento, le parole che vengono considerate appartenenti allo stesso contesto, avranno i relativi *embeddings* simili. Dal momento che un *embedding* non è altro che un vettore, il risultato dell'addestramento può essere immaginato in uno spazio a m dimensioni in cui ogni parola è rappresentata da un punto le cui coordinate sono descritte dal corrispondente *embedding*, per cui parole usate in contesti simili, e quindi con *embeddings* simili si tradurranno in punti molto vicini tra loro nello spazio a m dimensioni. In questo modo, la Rete Neurale riesce ad apprendere il contesto in cui un Framework o una Libreria vengono usati e ad affiancare tra loro Frameworks e Librerie in base al contesto appreso.

L'Algoritmo appena descritto non è stato implementato da zero, ma è stata utilizzata una Libreria scritta in Python denominata *Gensim* [6] che offre l'implementazione dell'Algoritmo *Word2Vec* con la possibilità di impostare i parametri per l'addestramento della Rete Neurale; a tale scopo è stata impostata la dimensione dello spazio vettoriale a 200 e il numero di epoche della rete a 50. Prima di sottoporre il Corpus all'Algoritmo, è stata effettuata una fase di *preprocessing* che ha rimosso le *stopwords* dai *documenti* del *Corpus*, in particolare dalle descrizioni dei Frameworks e delle Librerie. Il lavoro descritto fino a questo momento viene svolto *una tantum* in quanto funge da addestramento per la rete neurale, al seguito del quale viene salvato il modello addestrato in maniera persistente. Una volta salvato il modello addestrato, è possibile testarlo su diversi input al fine di stabilire le parole più simili, in termini di contesto, alla parola input. Ponendo nuovamente l'attenzione sulla piattaforma *Code4Code*, la parola input sarà un Linguaggio, un Framework o una Libreria, così come dalle parole output verranno selezionati solo i Frameworks e le Librerie. In particolare verranno selezionati solo quelli la cui probabilità di trovarsi nel contesto della parola input è superiore al 50%. L'utente, dunque, seleziona i Linguaggi e i Framework che conosce, e la Rete Neurale processerà ognuna di queste Tecnologie per produrre un insieme di Frameworks e Librerie che gli consiglierà.

CAPITOLO 4

Code4Code

BREVE SPIEGAZIONE CONTENUTO CAPITOLO

4.1 Introduzione

4.2 Architettura del sistema

4.3 Framework utilizzati

4.4 Implementazione

CAPITOLO 5

Conclusioni

BREVE SPIEGAZIONE CONTENUTO CAPITOLO

Bibliografia

[1] Wikipedia contributors. Tandem language learning — Wikipedia, the free encyclopedia, 2021. [Online; accessed 11-August-2021]. (Citato a pagina 6)

[2] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. Sampling projects in github for MSR studies. In *Proceedings of the 18th International Conference on Mining Software Repositories*, MSR'21, page To appear, 2021. (Citato a pagina 11)

[3] Lorenzo Govoni. Come l'algoritmo apriori misura le regole di associazione. (Citato a pagina 12)

[4] ymoch. Apyori: A simple implementation of apriori algorithm by python. `https://github.com/ymoch/apyori`, 2016. (Citato a pagina 13)

[5] Chris McCormick. Word2vec tutorial - the skip-gram model. `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/`, 2016. (Citato a pagina 14)

[6] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`. (Citato a pagina 16)

Siti Web consultati

- Wikipedia – `www.wikipedia.org`
- Lorenzo Govoni – `www.lorenzogovoni.com`
- Seart GitHub Search – `https://seart-ghs.si.usi.ch/`

Ringraziamenti

INSERIRE RINGRAZIAMENTI QUI