



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..."*

- John Nash

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	ii
Elenco delle figure	iv
1 Introduzione	1
1.1 Contesto applicativo	1
1.2 Obiettivi e risultati	2
1.3 Struttura della tesi	4
2 Piattaforme collaborative e di apprendimento	5
2.1 Differenza tra piattaforme collaborative e di apprendimento	6
2.2 Esempi di piattaforme collaborative	6
2.3 Esempi di piattaforme di apprendimento	7
2.4 Tandem: una piattaforma che unisce collaborazione ed apprendimento	7
2.5 Considerazioni sulle piattaforme analizzate	8
3 Background sull'Intelligenza Artificiale utilizzata	9
3.1 Motivazioni all'utilizzo dell'Intelligenza Artificiale	10
3.2 Suggerimento di Tecnologie Software	10
3.2.1 Suggerimento di Linguaggi	11
3.2.2 Suggerimento di Frameworks e Librerie: Natural Language Processing	15
3.3 Suggerimento di utenti con cui collaborare	18

4	Code4Code	20
4.1	Funzionalità della piattaforma	21
4.2	Architettura del sistema	25
4.3	Framework utilizzati	26
5	Conclusioni	27
5.1	Sviluppi futuri	28
5.2	Decontestualizzazione del lavoro di Intelligenza Artificiale	29
	Bibliografia	31
	Ringraziamenti	33

Elenco delle figure

4.1	Form per l’inserimento dei Linguaggi padroneggiati dall’utente	21
4.2	Form per l’inserimento dei Frameworks e delle Librerie padroneggiate dall’utente	22
4.3	Risultato dell’Algoritmo che suggerisce Tecnologie Software	23
4.4	Mockup del calendario che tiene traccia degli schedule delle lezioni	24
4.5	Architettura della piattaforma CODE4CODE	25

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. I benefici che possono essere ottenuti dalla collaborazione nell'ambito di progetti di qualsiasi tipo sono innumerevoli: si può pensare all'aumento della produttività, alla suddivisione del lavoro, alla possibilità di valutare diverse idee evitando di focalizzarsi solo su un numero ristretto di opzioni, alla crescita personale e a tanti altri vantaggi. Se poi si considera l'ambito dello sviluppo Software, ci si rende conto del fatto che la collaborazione non sia un optional in quanto, al giorno d'oggi, un qualsiasi tipo di progetto Software richiede conoscenze di ambiti molto diversi fra loro, non solo dal punto di vista tecnologico. È certamente vero che nell'ambito di un progetto Software sono presenti diversi Team che lavorano con tipi di tecnologie tra loro diversi (ad esempio c'è chi si occupa del *DataBase*, chi si occupa del *Front-End*, eccetera), ma è altrettanto vero che la differenza tra le persone coinvolte in un progetto Software risiede anche nel settore in cui sono specializzate, che non sempre è quello informatico. Dunque, anche se si volessero ignorare i vantaggi derivanti dalla collaborazione che riguardano la produttività e la suddivisione del lavoro, le competenze che sono richieste al giorno d'oggi per lavorare ad un progetto sono difficilmente conciliabili in un unico individuo in quanto riguardano innumerevoli discipline.

Risulta, dunque, di fondamentale importanza maturare un approccio alla collaborazione e soprattutto alla comunicazione in quanto non è scontato che persone aventi background diversi riescano a comunicare in maniera agevole ed efficiente. Collaborare insieme ad altre persone ad un progetto non implica necessariamente riuscire a beneficiare di tutti i vantaggi della collaborazione: ci sono casi in cui scarse capacità di comunicazione hanno contribuito al fallimento di progetti. Una recente ricerca [1] finalizzata all'analisi delle motivazioni per le quali la piattaforma *SourceForge* è stata abbandonata, ha evidenziato vari problemi derivanti da diversi fattori, tra cui i cosiddetti *community smells* ossia

"caratteristiche organizzative e socio-tecniche sub-ottimali nell'ambito di una community software che può portare a costi aggiuntivi dovuti a problemi di comunicazione, rage-quitting e così via" (Tradotto) [2]

sintomo di una forte instabilità dal punto di vista organizzativo e comunicativo. I community smells sono stati solo una delle cause che ha portato problemi alla piattaforma, tuttavia non vanno assolutamente sottovalutati in quanto sono comportamenti più comuni di quanto si possa pensare e possono portare a conseguenze peggiori di quelle previste.

Il lavoro di tesi svolto, porta il concetto di collaborazione nell'ambito dell'apprendimento di tecnologie Software; tale iniziativa nasce dalla convinzione che anche in quest'ambito sia possibile sviluppare importanti capacità collaborative e soprattutto comunicative da poter sfruttare nell'ambito di progetti Software. 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 ¹, Frameworks e Librerie da studiare sulla base di quelli conosciuti dall'utente che ne usufruisce. Il primo micro-agente consiglia Linguaggi a partire dai Linguaggi conosciuti dall'utente, mentre il secondo micro-agente si occupa di suggerire Frameworks e Librerie sulla base dei Linguaggi, 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*. In particolar modo:

- Le *Association Rules* sono state impiegate allo scopo di ricavare, dato un insieme di Linguaggi conosciuti dall'utente, altri Linguaggi spesso utilizzati insieme ad essi nell'ambito di progetti Software, mediante semplici regole di tipo statistico applicate ai dati prelevati dalle Repositories pubbliche di *GitHub*. Quest'approccio ha lo scopo di consigliare agli utenti Linguaggi spesso utilizzati insieme a quelli che già conosce, in modo da completare la sua formazione.
- Il *Natural Language Processing* è stato impiegato allo scopo di analizzare un insieme di descrizioni di Frameworks e Librerie ed un insieme di tag di Repositories prelevate da *GitHub* per comprendere la correlazione esistente tra i Frameworks e le Librerie attualmente in voga. Tale analisi consente di stabilire, a partire da un insieme di Linguaggi, Frameworks e Librerie conosciuti da un utente, quali siano i Frameworks e le Librerie da consigliargli.

L'utilizzo dell'Intelligenza Artificiale verrà debitamente trattato ed approfondito 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

¹Nel 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 [...]

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.

Piattaforme collaborative e di apprendimento

Prima di progettare la piattaforma CODE4CODE è stato svolto un lavoro di ricerca al fine di comprendere in maniera approfondita la situazione attuale dal punto di vista dell'apprendimento online. Questo capitolo fa una breve panoramica sui concetti di piattaforme collaborative e di apprendimento, evidenziando le differenze tra le due tipologie di piattaforme. Vengono, successivamente analizzate alcune piattaforme collaborative e di apprendimento considerate affini alla piattaforma CODE4CODE ed in particolare delle suddette piattaforme vengono messe in luce le caratteristiche considerate rilevanti nella trattazione effettuata, con particolare attenzione a quelle che possono fare la differenza sulla qualità di una piattaforma dal punto di vista della collaborazione. Al termine del capitolo, viene presentata una piattaforma il cui approccio all'apprendimento e alla collaborazione si basa su un principio cardine che è alla base della piattaforma CODE4CODE, ne vengono analizzati gli aspetti positivi e ne vengono riconosciuti i limiti intrinseci. 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'insegnante 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* [3] 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 educativo, 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.

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. Nel presente capitolo, in primo luogo si motiva l'impiego dell'Intelligenza Artificiale mediante una panoramica generale e ad alto livello sulle problematiche risolvibili con essa. Successivamente vengono analizzate singolarmente le problematiche risolte, introducendo, con esse, gli strumenti di Intelligenza Artificiale impiegati con i relativi fondamenti teorici. Dopo l'infarinatura teorica che serve a comprendere in modo più agevole il lavoro svolto, vengono illustrate le modalità tramite le quali sono stati prelevati i dati sui quali lavorano i moduli di Intelligenza Artificiale sviluppati e vengono fornite specifiche motivazioni riguardanti alcune scelte effettuate in fase implementativa. Il presente capitolo risulta, dunque, essere il più denso di tecnicismi ed il cuore dell'intero lavoro di tesi. La sua stesura ha richiesto uno studio da diverse fonti, indicate nell'apposita sezione, che possono essere consultate per ottenere ulteriori informazioni sui fondamenti teorici e matematici degli strumenti di Intelligenza Artificiale utilizzati. Il capitolo si chiude con la proposta di progettazione di un altro Agente di Intelligenza Artificiale che potrebbe fornire un ulteriore supporto agli utenti della piattaforma.

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.

Tra queste due problematiche è stata debitamente affrontata, progettata ed implementata solo la prima, mentre della seconda verrà fornita 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, 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*.

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 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* [4] 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 commits e 100 stars. Il Tool sopracitato ha prodotto un file output in formato *JSON*, di cui 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* [5], 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 con 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:

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

- 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:

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

- 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:

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

L'algoritmo Apriori sfrutta queste tre misure di probabilità al fine di generare le Association Rules. 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:

$$\{L_1, L_2, \dots, L_i\} \rightarrow \{L_{i+1}, \dots, L_k\} \quad (3.2.4)$$

Ad ogni Association Rule sono associate la Confidence e il Lift relativi ai Linguaggi $\{L_{i+1}, \dots, L_k\}$ (tale termine della Regola è detto *conseguente*) considerando le Repository contenenti i Linguaggi $\{L_1, L_2, \dots, L_i\}$ (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.

Per generare la Association Rules, l'algoritmo Apriori deve seguire una serie di passi in maniera iterativa. Per comprendere l'algoritmo è necessario conoscere il concetto di *k-itemset*: un *k-itemset* è un insieme di k elementi che vengono associati tra loro; nel caso in esame un

k-itemset è un insieme di *k* Linguaggi. L'algoritmo, a partire dal DataSet dei Linguaggi delle Repositories, genera iterativamente gli *itemset* partendo da un *1-itemset* fino ad arrivare ad una condizione di terminazione, per cui la generazione arriverà fino ad un *k-itemset*. Ad ogni *itemset* generato, vengono esclusi gli item che non raggiungono la soglia *s* di Support minima che viene presa in input dall'algoritmo. Per generare un *1-itemset*, viene seguita una procedura molto semplice: si conteggia ogni singolo Linguaggio ottenendo il numero di volte in cui compare nelle Repositories, si escludono i Linguaggi che compaiono nelle Repositories con una percentuale inferiore al Support *s* indicato. Tale lista di Linguaggi rappresenta l'*1-itemset* e sarà la base per costruire il *2-itemset*. Infatti, per la generazione del *2-itemset* vengono considerati i Linguaggi ottenuti dall'*1-itemset* e ad ognuno di essi viene associato ogni altro Linguaggio possibile, ottenendo un *itemset* fatto da coppie (ossia un *2-itemset*). Dall'*itemset* così ottenuto vengono eliminati gli elementi (le coppie di Linguaggi) il cui Support è al di sotto della soglia *s* citata in precedenza. Si procede iterativamente nel modo descritto, tuttavia dal *3-itemset* in poi diventa evidente una specifica proprietà dell'algoritmo: ogni possibile sottoinsieme di un *k-itemset* deve essere incluso in uno dei precedenti *itemset*. In altri termini, ogni sottoinsieme di un *k-itemset* deve avere un Support superiore alla soglia *s*. Tale proprietà dell'algoritmo porta alla condizione di terminazione in quanto l'algoritmo si ferma nel momento in cui non è più possibile generare un *itemset* di cui tutti i sottoinsiemi appartengono ad un *itemset* generato precedentemente. Arrivati all'ultimo *k-itemset*, l'algoritmo riesce a generare tutte le possibili Association Rules relative ai Linguaggi, andando a considerare tutte le possibili combinazioni di Linguaggi nel termine antecedente e nel termine conseguente delle regole. Verranno scartate le regole aventi una Confidence o un Lift al di sotto delle soglie prese in input dall'algoritmo.

L'algoritmo Apriori non è stato implementato da zero, ma è stata utilizzata una sua implementazione in *Python* denominata *Apyori* [6] 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 $\frac{n}{2}$ Linguaggi a cui è associata la Confidence più elevata, se supera 20 vengono conservati solo gli $\frac{n}{4}$ 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 Framework 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* contenente i Topic 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 interpretare e dare un senso ai dati ed estrarre informazioni utili sulla correlazione tra i Framework, sulla correlazione tra le Librerie e sulla correlazione tra Framework e Librerie.

L'algoritmo di NLP utilizzato è *Word2Vec* [7] 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. In particolare, esistono due versioni distinte del modello *Word2Vec*:

- Il modello *Skip-grams* [8], che ha come obiettivo quello di calcolare la distribuzione di probabilità delle parole che si trovano intorno ad una determinata parola input. Per la risoluzione del problema in questione è stato utilizzato questo tipo di modello.
- Il modello *Continuous-bag-of-words*, che ha come obiettivo quello di predire la "parola centrale" dato un certo numero di parole di contesto.

Quando si parla di *Word2Vec* 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* [9] 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.

3.3 Suggerimento di utenti con cui collaborare

La questione relativa al suggerimento di utenti con cui collaborare nell'ambito della piattaforma, non è stata interamente progettata ed implementata com'è evidentemente accaduto nel caso della questione del suggerimento delle tecnologie Software, tuttavia è possibile fornire un accenno di progettazione in merito. È chiaro che due utenti possano collaborare se uno conosce tecnologie che interessano all'altro, tuttavia questo non è l'unico fattore in gioco che determina la possibilità di due utenti di collaborare; si possono considerare, in astratto, un insieme di fattori in gioco e si può immaginare di disporre di un algoritmo che, a partire da un insieme di utenti, tenendo conto di tali fattori consiglia all'utente un insieme di partner




















con cui collaborare. L'algoritmo in questione dovrebbe massimizzare (o minimizzare) una funzione di fitness che rispecchia un qualche tipo di combinazione matematica dei parametri. I parametri da considerare dipendono strettamente dalla piattaforma; alcuni di questi riguardano il numero di partner di un determinato utente e le recensioni acquisite nell'insegnamento di una determinata tecnologia. Tra i diversi algoritmi utilizzabili, sono stati valutati gli algoritmi genetici, ossia algoritmi di ricerca che si ispirano alla genetica e si prestano bene alla risoluzione di questo problema sia per via della tipologia della funzione obiettivo che è basata sull'ottimizzazione di diversi parametri, che per la struttura del problema che sarebbe facilmente traslabile sulle strutture di un algoritmo genetico. Tale tipo di algoritmo (che sarebbe più corretto definire meta-euristica) effettua una ricerca "portando avanti" un insieme di soluzioni anziché una singola e conserva quelle considerate più promettenti dal punto di vista della funzione di fitness. Come anticipato in precedenza, gli algoritmi genetici hanno diverse analogie con la genetica, in particolare le soluzioni che vengono "portate avanti" sono definite "individui" e le componenti che formano tali soluzioni sono i "geni" degli individui. Per mostrare la similarità tra le strutture dell'algoritmo genetico e quelle del problema in questione, è possibile effettuare un parallelismo tra i due contesti: un individuo è un gruppo di utenti candidati per essere consigliati e un gene è un singolo utente. La trattazione appena effettuata ha lo scopo di dare una semplice intuizione su quella che può essere l'idea alla base della risoluzione di tale problema. Se si volesse approfondire anche soltanto il discorso degli algoritmi genetici, sarebbe doveroso approfondire diversi aspetti che sono stati tralasciati, come ad esempio gli operatori genetici che vengono utilizzati per far mutare ed evolvere le soluzioni.

La piattaforma CODE4CODE non è stata implementata interamente, ma solo nella parte relativa all'Intelligenza Artificiale. Nonostante ciò l'idea della piattaforma è concreta ed il messaggio che vuole comunicare è chiaro e preciso e per farlo sono necessari vari strumenti che sono stati ideati e dei quali si ha una parziale idea di implementazione. In questo capitolo vengono ripetuti gli obiettivi della piattaforma mediante una panoramica generale della stessa e vengono presentate le funzionalità principali che assolvono allo scopo di supportare la collaborazione tra gli utenti. È presente un breve workflow dell'utilizzo della piattaforma che chiarisce il modo in cui gli strumenti di Intelligenza Artificiale presentati nei precedenti capitoli vengono utilizzati nel momento in cui l'utente usufruisce della piattaforma. Il capitolo si conclude con la trattazione di aspetti relativi all'architettura del sistema e ai Framework utilizzati.

4.1 Funzionalità della piattaforma

Seppur implementata solo nella parte relativa all'Intelligenza Artificiale, la piattaforma CODE4CODE è stata ideata per assolvere ad un certo compito e per avere determinate funzionalità. Come specificato nei precedenti capitoli, la piattaforma CODE4CODE si basa su uno scambio reciproco di conoscenze tra gli utenti facendo sì che non vi sia necessità di scambi di denaro. In altri termini, se l'utente *A* desidera imparare una certa tecnologia *X* non deve disporre di una somma di denaro da offrire all'utente *B* che gli insegna ad usare quella tecnologia, in quanto l'utente *B* avrà bisogno di imparare una tecnologia *Y* che *A* potrà insegnargli. I punti focali dell'applicazione risultano essere, dunque, la comunicazione e il matching tra i diversi utenti che deve avvenire secondo parametri ben precisi, oltre che gli strumenti di supporto che la piattaforma stessa deve fornire agli utenti per l'apprendimento. Per comprendere la rilevanza di questi fattori è conveniente seguire il workflow dell'utente registrato all'applicazione (verranno saltati i passaggi relativi alla registrazione e all'autenticazione in quanto sono passaggi "classici"). Dopo che l'utente effettua la registrazione e fa il Log In per la prima volta viene sottoposto ad alcune semplici domande inerenti al mondo della programmazione che verranno usate per "conoscerlo meglio" e per comprendere le sue esigenze e quello che ha da offrire agli altri utenti della piattaforma. La figura 4.1 mostra l'interfaccia utente per l'inserimento dei Linguaggi padroneggiati, mentre la figura 4.2 mostra l'interfaccia per l'inserimento dei Frameworks e delle Librerie padroneggiate.





























Seleziona i linguaggi che conosci

 Assembly	 Batch	 C#	 C++
 C	 CoffeeScript	 CSS	 Go
 Groovy	 Hack	 HTML	 Java
 JavaScript	 Kotlin	 Lua	 Objective-C
 Perl	 PHP	 PowerShell	 Python
 Ruby	 Shell	 SQL	 Swift
 LaTeX	 TypeScript		

Avanti

Figura 4.1: Form per l'inserimento dei Linguaggi padroneggiati dall'utente

Seleziona i framework e le librerie che conosci

 .NET	 AJAX	 Amp	 Android
 Angular	 ASP.NET	 Bootstrap	 Django
 Electron	 Ember	 Express	 Firebase
 Flask	 jQuery	 Keras	 Koa
 Node	 Ruby On Rails	 Ratchet	 ReactJS
 React Native	 ReactiveUI	 Scikit Learn	 Spring Boot
 Symfony	 TensorFlow	 Thymeleaf	 Vue

Conferma

Figura 4.2: Form per l’inserimento dei Frameworks e delle Librerie padroneggiate dall’utente

È possibile evitare che l’utente debba inserire manualmente le tecnologie che conosce, dandogli la possibilità di effettuare il Log In con *GitHub* e chiedendogli di concedere l’autorizzazione alla lettura delle sue Repositories pubbliche e private. In questo modo è possibile prelevare i dati relativi alle tecnologie utilizzate nelle sue Repositories. A tal proposito è necessario fare due precisazioni:

- In primo luogo ottenere le tecnologie utilizzate nella Repositories significa ottenere i Linguaggi, i Frameworks e le Librerie. I Linguaggi sono facilmente ottenibili grazie alle API pubbliche messe a disposizione da *GitHub* (un semplice modulo che effettua tale tipo di lavoro è stato facilmente implementato), mentre per quanto riguarda i Frameworks e le Librerie ci si affiderà ai tag delle Repositories (che non sempre contengono questo tipo di informazioni) e verranno utilizzate opportune tecniche per ricondurre il nome del tag al nome del Framework o della Libreria corrispondente, eventualmente andando ad utilizzare tecniche simili o del tutto equivalenti a quelle di *NLP* presentate nel precedente capitolo.
- Il fatto che una determinata tecnologia sia presente in una Repository di cui l’utente risulta essere contributor, non significa necessariamente che l’utente padroneggi quella tecnologia; per tale ragione è opportuno che le tecnologie prelevate da *GitHub* vengano proposte all’utente e che gli venga, dunque, data la possibilità di modificarle opportunamente.

Giunti a questo punto, la piattaforma suggerisce all'utente un insieme di Linguaggi, Frameworks e Librerie consigliati utilizzando le metodologie trattate nel capitolo precedente, come illustrato nella figura 4.3.

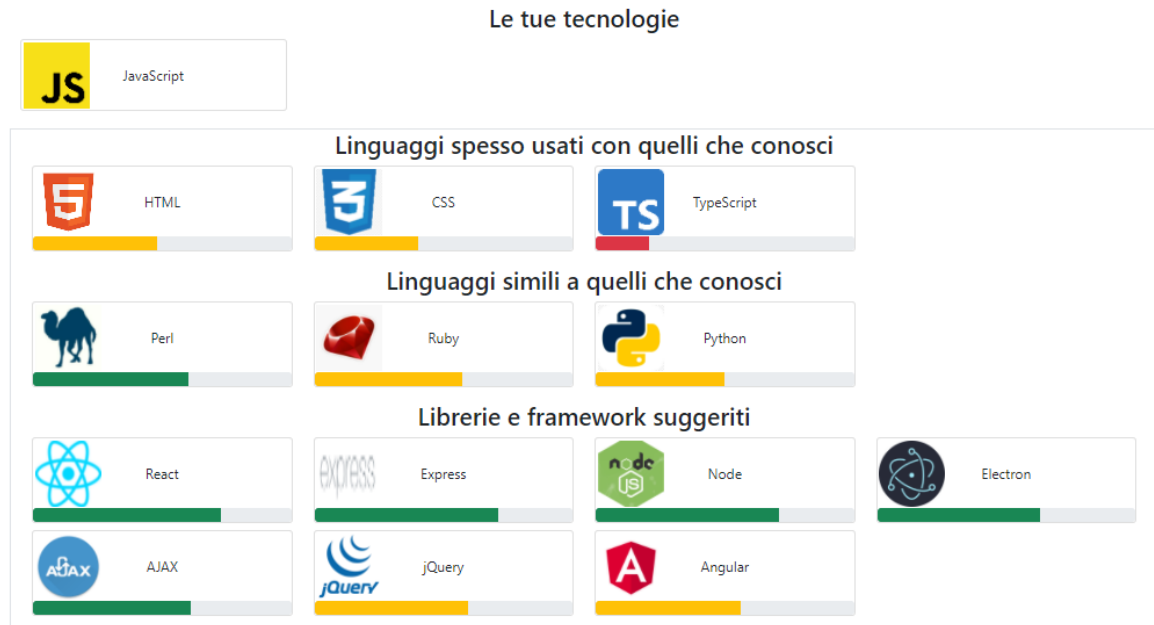


Figura 4.3: Risultato dell'Algoritmo che suggerisce Tecnologie Software

L'utente può scegliere una o più tecnologie da imparare e la piattaforma provvederà a suggerirgli un insieme di utenti con cui iniziare una collaborazione mediante un algoritmo di Intelligenza Artificiale di cui è stato fornito un accenno di progettazione nel capitolo precedente. Due utenti si mettono in contatto tramite una richiesta di collaborazione che potrà essere effettuata solo se l'utente destinatario della richiesta padroneggia almeno una tecnologia che l'utente mittente vuole imparare e vuole imparare almeno una tecnologia padroneggiata dall'utente mittente. Se l'utente destinatario accetterà la richiesta, si creerà una collaborazione tra i due utenti; a questo punto la piattaforma deve fornire un insieme di strumenti che consentono ai due utenti di collaborare in maniera agevole ed efficiente. Di seguito verranno proposte e descritte diverse funzionalità:

- Dovrebbe essere previsto un sistema di chat immediato che consenta agli utenti di comunicare in tempo reale per accordarsi sulle sessioni di "apprendimento"
- Le sessioni di apprendimento dovrebbero avvenire tramite lezioni in tempo reale, svolte mediante chiamate o videochiamate, in cui sia possibile condividere lo schermo per far sì che l'utente che insegna possa mostrare in tempo reale il codice che scrive e l'esecuzione dello stesso. Tali lezioni dovrebbero essere schedate e la piattaforma

dovrebbe tenere traccia dello schedule stesso mediante un calendario, che possa essere usato sia per tracciare lo schedule delle lezioni, che per aggiungerne di nuove. Un utente può aggiungere una lezione in cui lui sarà insegnante, potrà scegliere l'allievo e la tecnologia da insegnare e la lezione aggiunta verrà subito mostrata sul calendario dell'utente allievo. (figura 4.4).

lun	mar	mer	gio	ven	sab	dom
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	<div>Lez. 5 Insegnante: Mario Rossi Allievo: Luigi Verdi Orario: 15:00-16:00 Lezione: Java <div>+ Agg. una nuova lez.</div></div>		21
22	23	24	25			28

Figura 4.4: Mockup del calendario che tiene traccia degli schedule delle lezioni

- Il miglior modo per imparare è "provare sul campo", quindi dopo le sessioni di apprendimento è necessario che l'utente che sta imparando provi praticamente quanto ha appreso durante le lezioni. A tale scopo la piattaforma deve mettere a disposizione un sistema di assegnamento di task tramite il quale l'utente che sta insegnando può assegnare uno o più task all'utente che sta imparando. Il tutto potrebbe essere semplificato tramite un'integrazione con *GitHub*, in modo tale che la piattaforma tenga traccia dei task (testo dell'esercizio), di un eventuale schedule dei task (tramite un calendario) e del link alla Repository di *GitHub* contenente una proposta di soluzione svolta dall'utente che sta imparando, che verrà poi corretta dall'utente che insegna.

È importante sottolineare che i ruoli di "utente che insegna" e di "utente che impara" sono assolutamente dinamici in quanto nell'ambito di diversi meeting questi due ruoli potrebbero invertirsi (idealmente vi sarà un numero di meeting in cui l'utente A è insegnante e B è allievo, uguale al numero di meeting in cui l'utente A è allievo e B è insegnante).

4.2 Architettura del sistema

La piattaforma CODE4CODE si concretizza in una *WebApp* fruibile mediante un semplice Web Browser. Vi è un *Web Server* scritto in *Java* che serve le richieste del Client mediante delle *REST API*. Durante l'utilizzo della piattaforma si interagisce con tre ulteriori Server:

- Il primo Server, scritto in *Python*, fornisce i risultati del modulo di Intelligenza Artificiale che suggerisce Frameworks e Librerie all'utente.
- Il secondo Server è quello di *Mattermost* [10], strumento pensato per risolvere il problema delle chiamate e delle videochiamate con condivisione schermo. *Mattermost* è una piattaforma *open source* e *self-hosted* a sé stante, ideata per la collaborazione nell'ambito di un'organizzazione e che fornisce servizi di chat in tempo reale con tanto di condivisione di file, chiamate, videochiamate, creazione di canali di comunicazione, funzionalità di condivisione schermo, il tutto accessibile mediante Web Browser come il resto delle funzionalità della piattaforma CODE4CODE. Dal momento che *Mattermost* è *open source*, è possibile ricompilare i sorgenti per modificare sia il *Front-End* che il *Back-End* della piattaforma al fine di adattarla ulteriormente al contesto di utilizzo evitando gli utilizzi impropri e decontestualizzati da parte dell'utente.
- Il terzo Server è il *DataBase Server* che mediante un apposito *DBMS* gestisce le entità persistenti della piattaforma.

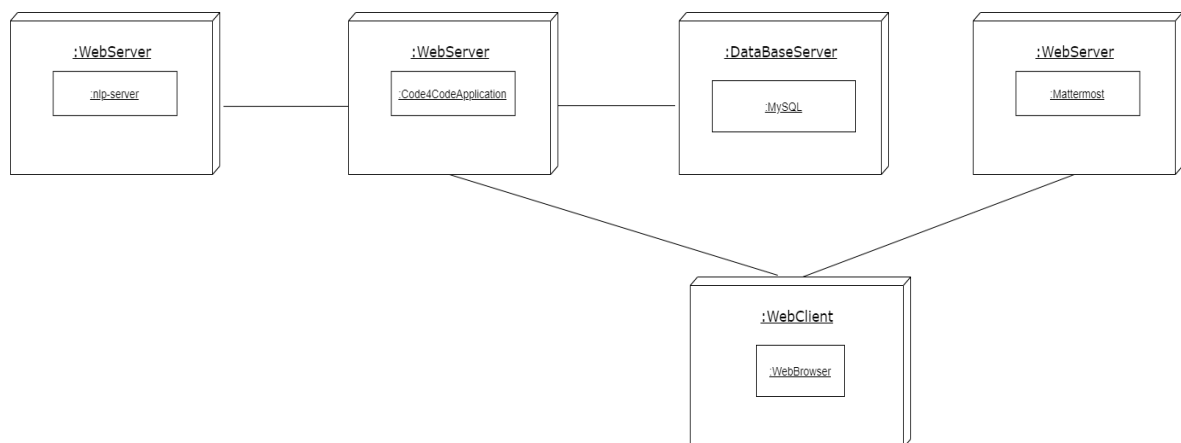


Figura 4.5: Architettura della piattaforma CODE4CODE

4.3 Framework utilizzati

Per la realizzazione della piattaforma CODE4CODE sono stati impiegati vari Frameworks e Librerie.

- Per il *Web Server* in *Java* è stato utilizzato *Spring* che ha reso agevole e veloce lo sviluppo delle *REST API* implementate. Inoltre è stato utilizzato il motore di template *Thymeleaf* per uno sviluppo moderno delle pagine *HTML*, la cui parte grafica è stata facilmente curata grazie al Framework *Bootstrap*.
- Per la parte di *Natural Language Processing* in *Python* è stata usata la Libreria *Gensim* che ha fornito un'implementazione dell'algoritmo *Word2Vec* per l'addestramento della Rete Neurale.
- Per le *Association Rules*, utilizzate per il suggerimento dei Linguaggi, è stata utilizzata la Libreria *Apyori* grazie alla quale è possibile usufruire dell'implementazione dell'algoritmo *Apriori* per la generazione delle *Association Rules*.
- Per il *Web Server* in *Python* è stato utilizzato *Flask*, che ha consentito di configurare il Server in maniera agevole con poche righe di codice.

Conclusioni

Quest'ultimo capitolo conclude la trattazione del lavoro di tesi attraverso una riflessione su quanto sviluppato, ma soprattutto su quanto non è stato ancora sviluppato. Viene posta molta attenzione su quelli che saranno gli sviluppi futuri della piattaforma CODE4CODE e dei relativi strumenti di supporto che essa dovrà fornire, e vengono, inoltre, valutate diverse opzioni di fronte a scelte implementative future analizzando gli aspetti positivi e negativi di ognuna di esse. Pensare al futuro della piattaforma, infatti, è parte integrante del lavoro di tesi in quanto l'implementazione risulta essere parziale e lascia spazio ancora a tanto lavoro, sebbene gran parte della piattaforma sia stata ideata per intero. Per concludere, vengono effettuate considerazioni sulla generalizzabilità di quanto effettuato dal punto di vista dell'Intelligenza Artificiale; è stato, infatti, proposto un utilizzo alternativo dell'Agente Intelligente sviluppato, che non è contestualizzato nella piattaforma CODE4CODE.

5.1 Sviluppi futuri

Come puntualizzato diverse volte nell'ambito della presente trattazione, il lavoro su cui ci si è soffermati maggiormente riguarda lo sviluppo dei moduli di Intelligenza Artificiale. Per tale ragione gli Agenti sviluppati sono stati testati in modo da verificare quanto sviluppato e, laddove sono stati rilevati risultati non esattamente conformi alla realtà, sono stati analizzati i motivi che hanno causato determinati problemi per comprendere se il risultato in questione fosse un semplice *outlier* o se ci fossero gravi errori nell'impostazione dei parametri di addestramento del modello utilizzato. Diverse volte è risultato opportuno modificare radicalmente i DataSet utilizzati, mentre altre volte sono stati integrati quelli preesistenti con DataSet di tipologie diverse; tali tipi di operazioni, insieme alla correzione dei parametri di addestramento, hanno migliorato la qualità dei risultati ottenuti. Allo stato attuale, dunque, si dispone di un modulo di Intelligenza Artificiale completo e funzionante, testato con un discreto numero di tecnologie Software ed integrato in un'implementazione primordiale della piattaforma CODE4CODE. Tra gli sviluppi futuri sarà, sicuramente, prevista l'implementazione integrale della piattaforma CODE4CODE con tutti i relativi strumenti di supporto descritti nel capitolo precedente. Tale implementazione proseguirà, presumibilmente, sulla scia di quanto attualmente realizzato, anche dal punto di vista di tecnologie e Frameworks utilizzati. Per quanto attualmente implementato, infatti, l'utilizzo di *Spring* ha portato alla realizzazione di un prototipo in maniera molto agevole e veloce, tuttavia vi sono numerose potenzialità di *Spring* non ancora sfruttate, come ad esempio la possibilità di utilizzare *JPA* (*Java Persistence API*) [11] per la gestione dei dati persistenti della piattaforma. Sarà, inoltre, necessario prendere una decisione definitiva riguardante il sistema di comunicazione utilizzato dagli utenti:

- La prima opzione è l'utilizzo di *Mattermost*: in tal caso sarebbe necessario apportare opportune modifiche al codice sorgente del Server sia lato *Front-End* che lato *Back-End* per evitare che gli utenti possano usufruire di *Mattermost* in maniera impropria ed indipendente dalla piattaforma. Tale problema si pone in quanto *Mattermost* è una piattaforma a sé stante e che prevede funzionalità che prescindono da quelle della piattaforma nella quale può, eventualmente, essere integrata.
- La seconda opzione è la creazione di un sistema di comunicazione integrato ad-hoc per la piattaforma. Ciò non comporterebbe la risoluzione dei problemi di integrazione che si hanno nel caso di *Mattermost*, ma sarebbe necessario riprogettare da zero un

sistema di comunicazione in tempo reale che sia agevole nell'utilizzo e abbia tutte le caratteristiche e le funzionalità esposte in precedenza.

L'ultimo aspetto da trattare riguarda l'algoritmo di Intelligenza Artificiale che consiglia agli utenti dei partner con cui collaborare. Nel capitolo sull'Intelligenza Artificiale è stata proposta un'idea di progettazione ed implementazione dell'algoritmo, tuttavia affrontare concretamente un problema del genere richiede un lavoro non indifferente dal momento che è necessario scegliere, progettare ed implementare tutto ciò di cui un algoritmo genetico necessita per funzionare; ad esempio stabilire quale algoritmo adottare per gli operatori genetici che l'algoritmo dovrà utilizzare, può portare a grosse differenze nei risultati finali.

5.2 Decontestualizzazione del lavoro di Intelligenza Artificiale

Sarebbe del tutto lecito chiedersi il motivo per il quale si sia deciso di sviluppare il modulo di Intelligenza Artificiale relativo al suggerimento di tecnologie Software, piuttosto che quello che suggerisce partner con cui collaborare. Il motivo di tale scelta risiede nel fatto che il modulo di Intelligenza Artificiale che consiglia tecnologie Software ha una duplice valenza, in quanto può essere decontestualizzato dalla piattaforma CODE4CODE ed utilizzato per altri scopi, mentre l'algoritmo che consiglia partner con cui collaborare è strettamente legato alla piattaforma e dipende da parametri ad essa relativi. L'algoritmo implementato può essere utilizzato al fine di effettuare studi statistici per comprendere quali Linguaggi e quali combinazioni di Frameworks e Librerie vengono spesso utilizzate insieme, il tutto in maniera assolutamente automatizzata basandosi su progetti reali ottenuti da *GitHub*. Tali studi potrebbero essere utilizzati per molteplici scopi, come ad esempio il rilevamento di cambi di tendenza nel panorama di utilizzo di Frameworks e Librerie; a tale scopo bisognerebbe, chiaramente, prelevare le nuove Repositories periodicamente da *GitHub* e assicurarsi di interpretare correttamente le variazioni che si ottengono nel tempo evitando di confondere differenze insignificanti con cambi di scenario che in realtà in quel momento non si stanno verificando.

Un altro tipo di applicazione dell'Agente implementato potrebbe consistere nel suggerimento dell'utilizzo di un determinato Framework o di una determinata Libreria nell'ambito di un progetto, sapendo che verranno impiegati determinati Linguaggi. Infatti, il DataSet di addestramento per il suggerimento di Frameworks è in larga parte composto da una lista di tag di Repositories, dalle quali è possibile evincere, con buona approssimazione, se in una determinata Repository venga utilizzato un certo Linguaggio insieme ad un certo Framework.

La logica alla base di tale approccio risiede nel fatto che se un Linguaggio ed un Framework sono strettamente correlati tra loro, molto probabilmente sono stati utilizzati insieme un buon numero di volte, ragion per cui è lecito pensare che la scelta di quel determinato Framework sia sensata (a patto che il Framework da utilizzare assolva effettivamente allo scopo che si vuole conseguire).

- [1] Damian Andrew Tamburri, Kelly Blincoe, Fabio Palomba, and Rick Kazman. "the canary in the coal mine..." a cautionary tale from the decline of sourceforge. (Citato a pagina 2)
- [2] Damian A. Tamburri, Fabio Palomba, and Rick Kazman. Exploring community smells in open-source: An automated approach. 08 2017. (Citato a pagina 2)
- [3] Wikipedia contributors. Tandem language learning — Wikipedia, the free encyclopedia, 2021. [Online; accessed 11-August-2021]. (Citato a pagina 7)
- [4] 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 12)
- [5] Lorenzo Govoni. Come l'algoritmo apriori misura le regole di associazione. (Citato a pagina 13)
- [6] ymoch. Apyori: A simple implementation of apriori algorithm by python. <https://github.com/ymoch/apyori>, 2016. (Citato a pagina 14)
- [7] Wikipedia contributors. Word2vec — Wikipedia, the free encyclopedia, 2021. [Online; accessed 5-September-2021]. (Citato a pagina 16)
- [8] Chris McCormick. Word2vec tutorial - the skip-gram model. <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>, 2016. (Citato a pagina 16)

- [9] 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 18)
- [10] Mattermost. (Citato a pagina 25)
- [11] Wikipedia. Java persistence api — wikipedia, l'enciclopedia libera, 2020. [Online; in data 2-settembre-2021]. (Citato a pagina 28)

Siti Web consultati

- Wikipedia – www.wikipedia.org
- Lorenzo Govoni – www.lorenzogovoni.com
- Seart GitHub Search – <https://seart-ghs.si.usi.ch/>
- GitHub Topics – <https://github.com/topics>
- Gensim topic modelling for humans – <https://radimrehurek.com/gensim/index.html>

Ringraziamenti

Tutto ciò che ho scritto in queste pagine è frutto di un indimenticabile viaggio che mi ha insegnato molto più di quanto io sia riuscito anche solo lontanamente a comunicare. Diverse persone hanno arricchito e reso entusiasmante il mio viaggio, ed in questo momento sento di volerle ringraziare.

In primo luogo voglio porre i miei più sentiti ringraziamenti al Prof. Fabio Palomba e al Dott. Emanuele Iannone, che dal primo momento hanno creduto nel mio progetto sostenendomi e dandomi consigli sul modo in cui approcciare i problemi che mi sono ritrovato a fronteggiare durante il mio cammino. Grazie a loro il mio percorso di tesi è stato un continuo esplorare e valutare nuove idee, talvolta nate da semplici osservazioni e poi diventate realtà.

Poco fa ho definito il mio percorso universitario un viaggio. Voglio ringraziare i preziosissimi compagni con cui ho condiviso questo viaggio: Lorenzo, Hermann, Felice, Mattia, Alfonso e Antonio. Ognuno di loro ha lasciato un segno indelebile su questo viaggio e senza di loro non sarei ciò che sono adesso. Tutto quello che abbiamo condiviso, soprattutto durante i momenti più bui di questi ultimi periodi, mi ha insegnato molto ed ha reso questo viaggio il più bello della mia vita.

Un ringraziamento particolare va al mio carissimo amico Vincenzopio che durante questi mesi si è costantemente interessato al mio lavoro di tesi, ed ha sempre ascoltato le mie idee e "sopportato" il mio irrefrenabile euforismo in alcuni momenti particolari. È sempre stato un punto di riferimento su cui poter contare ed un sostegno fondamentale durante questo viaggio.

Voglio ringraziare di cuore anche la mia amica Alessia, che mi ha sempre sostenuto ed ascoltato e che considero una delle persone più empatiche e sensibili che conosco. È sempre riuscita a comprendere i miei timori e le mie ansie ed ha sempre creduto in me dal primo giorno di quest'avventura.

Un ringraziamento speciale va al mio saggio mentore Mimmo Cioffi, che mi ha sempre stimolato e spronato a dare il meglio, credendo in me e nelle mie capacità dal giorno in cui mi ha conosciuto, soprattutto nei momenti in cui ero io a non credere più in me stesso. Se sono riuscito ad affrontare con positività e determinazione questo viaggio, è in gran parte merito suo e della fiducia che ha sempre riposto in me.

Quest'importante traguardo voglio dedicarlo ai miei genitori, che mi hanno cresciuto e sostenuto donandomi incondizionatamente una parte di loro stessi. Un doveroso ringraziamento va a loro e alla loro capacità di consigliare, senza mai avere la presunzione di imporre le loro idee; voglio ringraziarli per avermi dato anche la possibilità di sbagliare e di imparare dai miei errori, di cadere e di imparare a rialzarmi.

Un ultimo, commosso, ringraziamento va alla mia dolcissima zia Rina, che avrebbe tanto voluto condividere con me la gioia e l'emozione di questo traguardo.