**Moduli di Intelligenza Artificiale – Code4Code**

**Introduzione**Per assolvere in maniera completa alle proprie funzionalità, la piattaforma Code4Code dovrà integrare al proprio interno dei moduli di Intelligenza Artificiale il cui scopo è quello di assistere l’utente in diverse fasi.  
In primo luogo, quando l’utente si registra, dovrà fornire al sistema un insieme di informazioni circa le proprie conoscenze in termini di Linguaggi, Librerie e Framework (come chiarito nel *Problem Statement*). Queste informazioni potranno essere fornite sia in modo *esplicito* (nelle modalità indicate nel *Problem Statement*) che in modo *implicito*, grazie all’utilizzo dell’Intelligenza Artificiale: nel seguente documento ci si soffermerà proprio su quest’ultima modalità.  
Una volta che la piattaforma avrà raccolto abbastanza informazioni sull’utente potrà consigliare a quest’ultimo degli utenti candidati per cominciare una collaborazione, tenendo conto di diversi fattori (come già accennato nel *Problem Statement*), in particolar modo prestando attenzione al fatto che per quanto un utente possa essere più esperto rispetto ad altri nell’insegnamento di una determinata tecnologia (ciò sarebbe testimoniato dalle ottime recensioni), c’è un limite “umano” che gli impedirebbe di avere il tempo materiale di collaborare con più di un certo numero di utenti: è proprio in tale ambito che entra in gioco l’Intelligenza Artificiale. Nelle prossime sezioni verranno analizzati singolarmente i problemi.

**Ottenimento implicito delle skill e delle necessità dell’utente**  
Subito dopo aver completato la registrazione alla piattaforma, l’utente sarà messo di fronte ad una scelta: potrà compilare una sequenza di form contenenti le informazioni necessarie al sistema, oppure associare al proprio account gli account di *GitHub* e/o di *Stack Overflow*. Grazie alle *API* pubbliche esposte da queste piattaforme è possibile, previa autenticazione, ottenere una serie di informazioni utili:

* Nel caso di *GitHub* è possibile ottenere tutte le repository (sia pubbliche che private) di cui l’utente risulta *owner* o *contributor* e in particolar modo è possibile ottenere la lista dei linguaggi utilizzati in quella repository. Tale lista contiene, di certo, ottimi candidati ad essere linguaggi che l’utente padroneggia.
* Nel caso di *Stack Overflow* è possibile ottenere la lista dei tag assegnati alle domande effettuate dall’utente e a cui l’utente ha risposto. I tag assegnati alle domande potrebbero comprendere Linguaggi, Framework o Librerie; nel caso delle domande effettuate dall’utente, i tag potrebbero essere indicativi per le tecnologie a cui l’utente è interessato (sotto l’assunzione che se l’utente ha fatto una domanda che coinvolge una certa tecnologia, probabilmente è interessato ad approfondirla), nel caso di domande a cui l’utente ha risposto, i tag potrebbero essere indicativi per le tecnologie che l’utente conosce.

Non sfugge all’attenzione il fatto che nella deduzione delle tecnologie padroneggiate e di quelle di cui l’utente ha bisogno, vengono effettuate delle assunzioni che in alcuni casi potrebbero non essere del tutto corrette. L’obiettivo di tale *tool* è quello di effettuare una proposta di tecnologie conosciute e desiderate che potrà essere modificata dall’utente in qualsiasi momento. Dopo che il tool avrà dedotto grazie a GitHub e a StackOverflow le tecnologie che l’utente padroneggia, gli chiederà conferma dandogli la possibilità di modificarle prima di iniziare a consigliargli nuove tecnologie da imparare. La proposta relativa alle tecnologie da imparare si baserà anche sull’affinità tra linguaggi conosciuti e altri linguaggi; a tale scopo è necessario comprendere cosa si intenda per *affinità* tra linguaggi. Per risolvere tale problema risulta necessario poter effettuare una rappresentazione delle tecnologie strutturandole in modo che possano essere confrontate tra di loro. L’idea più intuitiva potrebbe essere quella di indicare per ciascun linguaggio i relativi paradigmi e ritenere affini tra loro linguaggi che presentano un certo numero di paradigmi in comune. Ogni linguaggio potrebbe essere considerato come appartenente ad una determinata categoria relativa al suo ambito di utilizzo (e.g.: HTML e CSS appartengono all’ambito dello sviluppo Web e quindi hanno un grado di affinità elevato).

**Obiettivi**

* È necessario fornire una schematizzazione delle tecnologie che consenta di considerare, per ciascuna di esse, una serie di parametri in base ai quali affermare che una tecnologia sia più o meno affine ad altre.
* Dopo aver costruito la schematizzazione di cui sopra, è necessario scegliere e costruire un modello di Intelligenza Artificiale che dato in input un insieme di tecnologie restituisce un insieme di tecnologie che siano quanto più compatibili possibile con le tecnologie specificate in input.

**Rappresentazione delle tecnologie**  
Le tecnologie di cui si occuperà la piattaforma sono di tre tipi: *Linguaggi*, *Framework* e *Librerie*. Per i linguaggi verrà utilizzato un tipo di schematizzazione, mentre per Framework e Librerie ne verrà utilizzato un altro che sarà fortemente legato al Linguaggio a cui fanno riferimento.

**Linguaggi**Un linguaggio è descritto dai seguenti elementi:

* Tipo: fa riferimento alla tipologia del linguaggio (e.g.: programmazione, scripting, markup…)
* Paradigmi: fa riferimento ai paradigmi utilizzati dal linguaggio (e.g.: imperativo, procedurale, a oggetti, funzionale…)
* Utilizzi: fa riferimento agli ambiti di utilizzo del linguaggio (e.g.: general purpose, sviluppo web, sistemi embedded…)

Sulla base di tali caratteristiche sarà possibile confrontare tra loro i diversi linguaggi in base al fatto che siano dello stesso tipo, abbiano un certo numero di paradigmi e casi d’utilizzo in comune.

**Framework e librerie**Framework e librerie sono descritti dai seguenti elementi:

* Linguaggio di riferimento: indica il linguaggio a cui il Framework o la libreria fa riferimento
* Scopo: fa riferimento allo scopo del Framework/libreria (e.g.: Intelligenza Artificiale, creazione Interfacce Utente, sviluppo applicazioni Web)

**Modello di Intelligenza Artificiale**Riflettendo sulla struttura del problema è possibile passare a rassegna un insieme di modelli di Intelligenza Artificiale e scegliere quello che meglio si sposa con la struttura stessa. Il problema si presta molto bene ad un modello di ricerca in quanto si può immaginare che a partire da un input (un insieme di tecnologie padroneggiate dall’utente) si esplora lo spazio di ricerca (insieme di tecnologie che l’utente non padroneggia) al fine di trovare un certo numero di tecnologie affini all’input. Un modello di apprendimento sarebbe ugualmente utilizzabile, tuttavia richiederebbe un addestramento e la disponibilità di un dataset di una certa dimensione per scongiurare diversi problemi che potrebbero presentarsi durante l’addestramento stesso.   
Bisogna, a questo punto, scegliere un modello di ricerca che possa lavorare al meglio con la rappresentazione delle tecnologie precedentemente definita.   
Un problema di ottimizzazione di vincoli potrebbe adattarsi alla situazione. A tal proposito:

* Le variabili che descrivono i Linguaggi sono *Tipo, Paradigmi* e *Utilizzo*; i vincoli sono rappresentati dal fatto che i linguaggi presenti abbiano determinate combinazioni *Tipo*-*Paradigmi*-*Utilizzo*.
* Le variabili che descrivono Librerie e Framework sono *Linguaggio di riferimento* e *Scopo*; i vincoli sono rappresentati dal fatto che i linguaggi presenti abbiano determinate combinazioni *Linguaggio di riferimento-Scopo*.

La funzione da ottimizzare prende in input i valori delle variabili della tecnologia considerata in quel momento nello spazio di ricerca, e in base ai valori delle variabili della tecnologia con cui si sta cercando di ottenere affinità, restituisce un certo punteggio che deve essere massimizzato; tale punteggio dipende dalla similarità tra i valori delle variabili.   
  
*Esempio 1: Considerando come input il linguaggio* *Java descritto nel seguente modo: {Tipo=Programmazione, Paradigmi=Orientato agli oggetti, Utilizzi=General Purpose}, l’agente esplorerà lo spazio di ricerca dei Linguaggi in modo da trovarne uno che abbia valori delle variabili quanto più simili possibile ai valori del linguaggio Java. Quando l’agente si troverà a valutare l’affinità tra Java e HTML descritto da {Tipo=Markup, Paradigmi=Dichiarativo, Utilizzi=Sviluppo Web} otterrà un punteggio molto basso, mentre quando dovrà valutare l’affinità tra Java e Kotlin descritto da {Tipo=Programmazione, Paradigmi=Orientato agli oggetti, Utilizzi=General Purpose} otterrà un punteggio molto alto dal momento che tutti i valori delle variabili di Kotlin coincidono alla perfezione con quelle di Java.   
Esempio 2: Considerando come input la libreria Python TensorFlow descritta nel seguente modo: {Linguaggio di riferimento=Python, Scopo=Intelligenza Artificiale}, la funzione restituirà punteggi alti in corrispondenza di altre librerie di Intelligenza Artificiale da usare con Python (ad esempio Keras).*

Per quanto illustrato finora, il sistema assegnerà ad ogni utente un Linguaggio per ogni Linguaggio che padroneggia e un Framework/Libreria per ogni Framework/Libreria che padroneggia. Potrebbero, tuttavia, presentarsi dei casi limite. Supponiamo che l’utente non conosca alcun Framework/Libreria: in quel caso l’agente non gliene consiglierebbe nessuno in quanto il matching, per com’è stato presentato finora, avviene sempre tra tecnologie dello stesso tipo. A tal proposito si potrebbe pensare ad una soluzione in cui se ad un linguaggio *X* padroneggiato dall’utente vengono associati solo linguaggi con punteggi “bassi” (bisogna comprendere cosa si intende per “punteggi bassi”), al posto di assegnargli uno di questi, si esplora lo spazio di ricerca dei Framework/Librerie al fine di cercarne uno che abbia come linguaggio di riferimento proprio *X*. Dopo che l’agente ha trovato un certo numero di tecnologie da proporre all’utente, gli dà la possibilità di modificare il risultato in modo che l’utente possa avere una base da cui partire e possa modificare la proposta a proprio piacimento.