UNIVERSITÀ DI PISA

Laurea Magistrale in Artificial Intelligence and Data Engineering

Documentazione del Progetto di Gestione dell'Innovazione

JOB DESCRIPTION AND CV COMPARISON

Domenico D'Orsi, Denny Meini, Matteo Razzai

A.A. 2023/2024

GitHub repository:
https://github.com/dominicofthebears/JobDescription_and_CVComparison

Sommario

1. Introduzione

Il crescente aumento di modelli e applicazioni basate sull'intelligenza artificiale è ormai una tematica estremamente importante nel panorama tecnologico mondiale, in quanto tali strumenti stanno proiettando il genere umano in un nuovo paradigma non solo tecnico-scientifico, ma si tratta di un radicale cambiamento nei nostri stili di vita: ciò include, naturalmente, l'aspetto lavorativo di ciascuno di noi.

Sono ormai numerose le aziende che hanno integrato tali sistemi all'interno del proprio modello di business a causa della forte riduzione dei costi che ciò implica, e la previsione è che esse tendano, con il progresso dei modelli, ad aumentare sempre di più. Basti pensare a quanto più efficiente ed economica sia, in campo di assistenza al cliente, utilizzare un chatbot o simili piuttosto che un referente umano, e tale ragionamento è facilmente estendibile a diversi task ripetitivi e che non richiedono un alto sforzo cognitivo. In sintesi, il concetto di human-in-the-loop per questa tipologia di impieghi andrà man mano svanendo a favore di tali tecnologie.

Abbiamo, per tali ragioni, voluto provare ad applicare tali strumenti al primo step che concerne il processo di ricerca di un lavoro, per verificare se allo stato attuale sia effettivamente funzionale, anche in questo ambito, supportare questa attività con l'Intelligenza Artificiale: tale step è quello della lettura di un'offerta di lavoro, suddivisa in varie sottosezioni, e sottomissione del proprio curriculum per poterne valutare l'aderenza all'offerta stessa.

2. Obiettivi del Progetto

Per fare ciò, abbiamo preso come riferimento la nota piattaforma online statunitense ***O*Net,*** la quale raccoglie al suo interno circa un migliaio di offerte di lavoro, ciascuna delle quali suddivisa nel dettaglio tra varie sottocategorie: quelle da noi considerate fanno riferimento alle *work activities*, ai *tasks* e alle *skills* richieste.

Esse sono presentate sotto forma di lista di elementi, per ciascuno dei quali è definito un grado di priorità per l'offerta di lavoro sotto forma di valore numerico in 100esimi. Tale rappresentazione potrebbe apparire molto asettica e poco invitante su una piattaforma quale ***LinkedIn*** ove solitamente si preferisce una maggior discorsività.

Gli obiettivi di tale progetto, dunque, sono stati quelli di raggruppare le varie offerte di lavoro per macrocategoria e per ciascuna offerta andare a generare, tramite l'uso di appositi modelli di AI, una sintesi discorsiva delle tre sottocategorie relative a work activities, tasks e skills nonché una descrizione complessiva del lavoro.

Lo step successivo, invece, è quello di permettere all'utente di caricare il proprio curriculum in formato .pdf da cui estrarne il contenuto e verificarne la somiglianza con la descrizione testuale dell'offerta stessa, restituendo un valore numerico da 0 a 1 che rappresenti proprio questa somiglianza.

3. Architettura

In questo capitolo andremo a presentare l'architettura del nostro progetto, ripercorrendo i modelli utilizzati e le principali operazioni che sono state svolte per poter raggiungere la versione finale della nostra applicazione.

3.1 Modelli e chiamate

Per la nostra applicazione abbiamo deciso di usare Python come linguaggio di programmazione, perché è un linguaggio molto potente e che si sposa benissimo con operazioni di scraping e clustering e con le chiamate API, che sono centrali nel nostro progetto.

Fondamentale per la nostra applicazione è stato **all-MiniLM-L6-v2** che è un modello di embedding facente parte di **sentence-transformers**, un modulo Python specializzato nella costruzione e nella manipolazione di embedding.

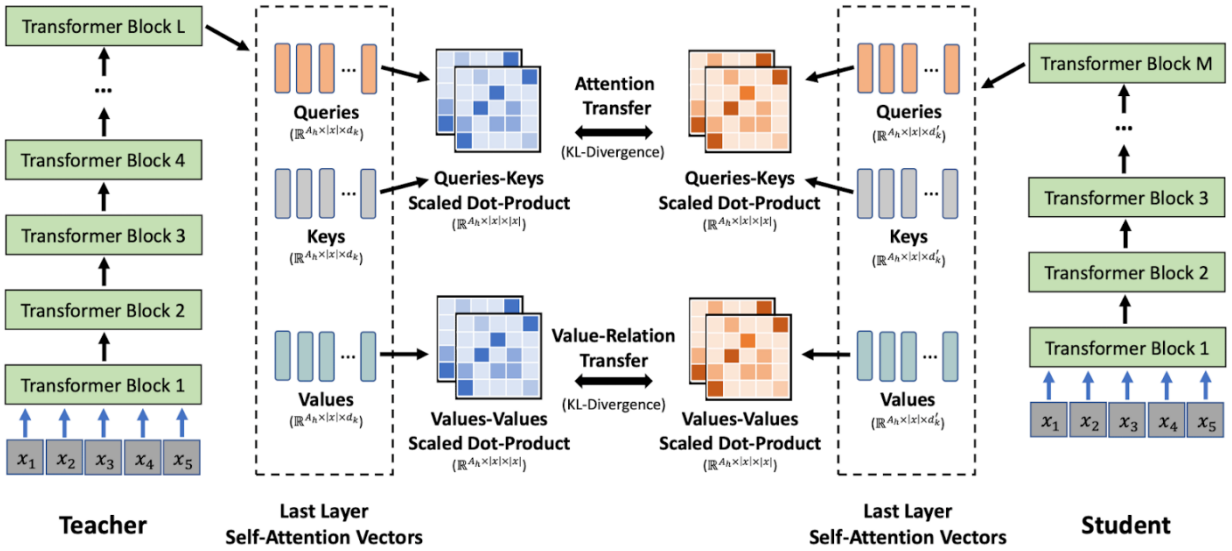


Figura 1: Architettura di MiniLM

Il modello è stato utilizzato per generare degli embedding, ovvero delle strutture per rappresentare un'informazione in modo numerico. Questa operazione è stata necessaria per poter clusterizzare i lavori in base alle loro descrizioni e per poter valutare l'affinità dell'utente con il lavoro scelto in base al curriculum.

Come modello di Intelligenza Artificiale abbiamo utilizzato **LLaMA 70B**, sviluppato da Meta e hostato dall'azienda americana **Groq** nata nel 2016. Groq ci permette di utilizzare una serie di modelli tramite le sue API, che è proprio quello che abbiamo fatto. Avevamo inizialmente pensato di usare la più famosa ChatGPT di OpenAI, ma abbiamo dovuto cambiare obiettivo dato che quest'ultima ha deciso di aumentare i costi.

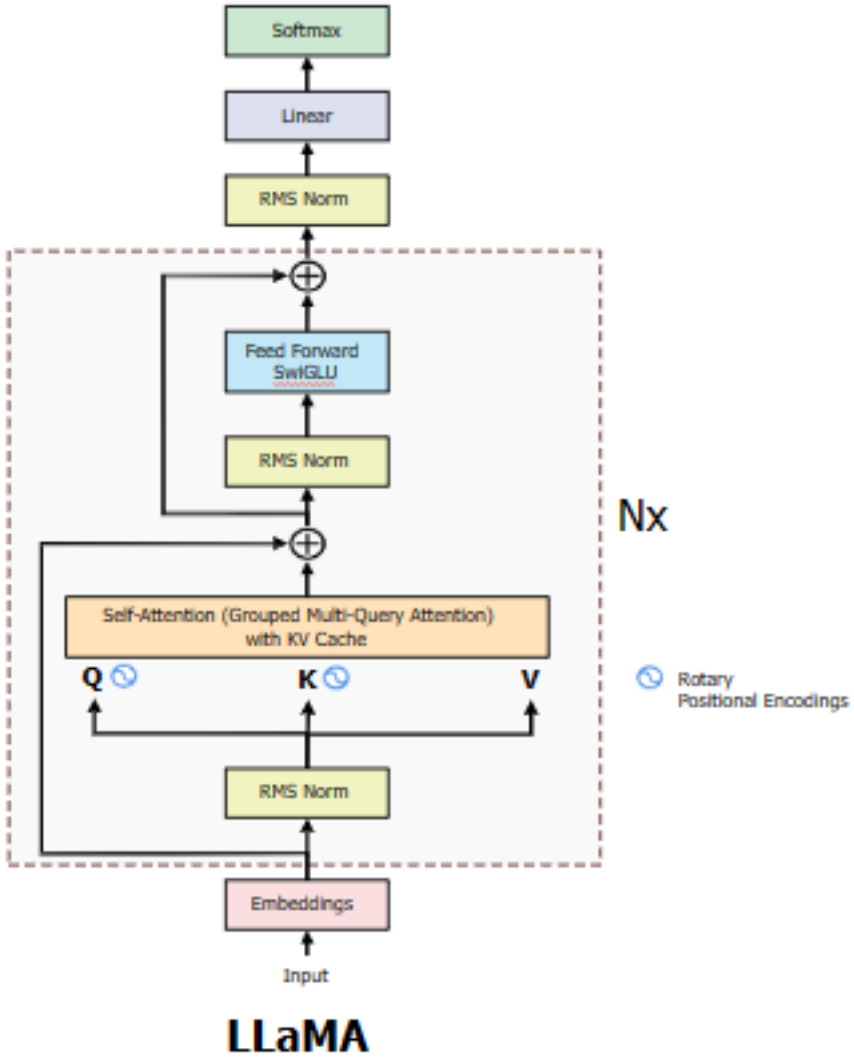


Figura 2: Architettura di LLaMA

LLaMA viene utilizzato per generare delle job description relative ai vari lavori disponibili, le descrizioni sono sia generali che divise in sezioni (work activities, skills e tasks). La stessa operazione verrà poi effettuata a partire dal CV dell'utente, in modo da poter verificare la compatibilità col lavoro scelto.

3.2 Database vettoriale

L'altra componente dell'applicativo che è necessario citare è quella del database vettoriale utilizzato: esso è stato necessario per poter migliorare il prompt sottoposto al modello per generare la job description tramite ***RAG* (retrieval-augmented generation)**. Essa consiste nell'aggiungere al prompt che si fornisce come input alcuni esempi di come andrebbe svolto il task che si sta richiedendo.

Per poter fare ciò, ci siamo serviti di un database con la possibilità di salvare al proprio interno vettori di grandi dimensioni, come gli embeddings da noi utilizzati: trattasi si ***Supabase***, per il quale è stato sufficiente andare a creare una tabella popolata in seguito con i nostri dati.

Per tale ragione, siamo partiti da un dataset abbastanza semplice ma completo di Job description (https://www.kaggle.com/datasets/ravindrasinghrana/job-description-dataset), e per ciascun record di tale dataset siamo andati a salvare nel database vettoriale la posizione lavorativa come chiave, l'embedding della descrizione come vettore e la descrizione in formato testuale come metadata.

Strutturando in tal modo i nostri dati, è stato possibile in fase di sottomissione del prompt andare a trovare le tre professioni più simili a quella in esame, tramite similarità del coseno dei relativi embeddings, ed aggiungere le suddette al prompt.

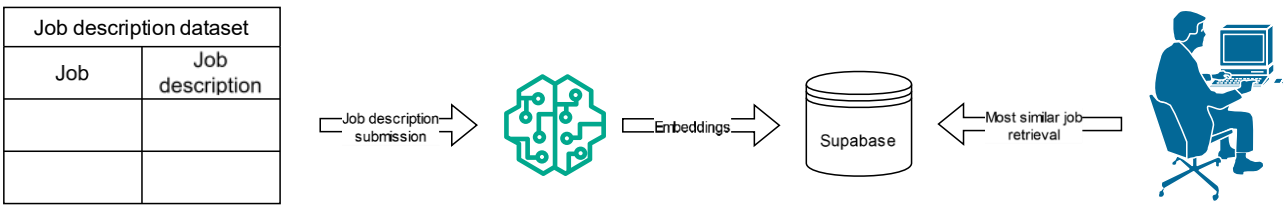


Figura 3: Interazione con il database vettoriale

3.3 Clustering

La prima operazione svolta è stata effettuare lo scraping dal sito di **O*NET** (https://www.onetonline.org/), ne parleremo meglio nel capitolo 3.4. Con il risultato dello scraping abbiamo costruito un file csv che, una volta convertito in DataFrame, è stato passato al modello **all-MiniLM-L6-v2** che ha così generato un embedding per ogni occupazione.

Successivamente abbiamo clusterizzato i lavori basandoci sugli embedding generati dal modello, con l'intento di suddividere i lavori in categorie. Abbiamo deciso di utilizzare **KMeans** come algoritmo di clustering perché è un algoritmo molto semplice ma al contempo estremamente efficace.

Questa fase ha richiesto un po' di tempo poiché abbiamo provato ad effettuare il clustering con un numero variabile di clustering desiderati. Un ulteriore problema derivava dal fatto che gli embedding generati avessero un numero molto alto di feature (circa 300) ed è un fattore che può creare problemi abbastanza importanti all'algoritmo. Quindi abbiamo tentato di mitigare questa problematica applicando un'operazione di riduzione delle feature (**Principal Component Analysis, PCA**), anche qui provando con una serie di valori per cercare di ottenere il miglior risultato possibile.

Dopo numerosi test abbiamo generato 10 cluster dopo una riduzione degli embedding a 50 feature ciascuno, ottenendo quindi le seguenti categorie:

- Industrial Machinery and Equipment Operators
- Service and Maintenance
- Administration and Support
- Science and Research
- Manual Operators and Technicians
- Health and Wellness
- Communication, Entertainment and Creative Professions
- Engineering and Technology
- Management and Supervision
- Education and Teaching

Alla fine di questo procedimento abbiamo ottenuto un file consistente in una lista di lavori con il relativo codice identificativo e la categoria di appartenenza.

3.4 Generazione della Job description e delle sottosezioni

Come detto in precedenza, per questo primo step ci siamo serviti del modello conversazionale di Meta noto come LLaMa al quale però, naturalmente, era necessario sottoporre una richiesta/comando sotto forma di prompt per poter ottenere il risultato desiderato. Per tale ragione, l'intera procedura nasce in primis dallo *scraping* dei relativi file .csv relativi alle sottosezioni di nostro interesse presenti nella specifica pagina di ciascuna offerta di lavoro, tramite apposita richiesta HTTP possibile grazie alla preventiva costruzione dell'apposito URL, partendo dal codice del lavoro stesso.

Di ciascun file, siamo andati a considerare le prime dieci voci, per poi andare ad estrapolarne e formattarne il testo, in quanto esso stesso sarebbe stato la prima componente del prompt relativo alle sottosezioni, seguito subito dopo da una spiegazione di cosa tali valori e descrizioni indicassero. A

ciò, siamo andati ad unire una descrizione testuale della procedura da effettuare, ovvero la generazione di un riassunto: tale prompt ha naturalmente subito diverse evoluzioni e accorgimenti per poterlo rendere sempre più funzionale, fino ad arrivare ad una versione che producesse una risposta soddisfacente da parte del modello.

Una volta generata la sintesi discorsiva di ciascuna delle tre sottosezioni, siamo passati a quella complessiva, facendo uso all'interno del prompt delle work activities, delle skills e dei task relativi al lavoro stesso, oltre che all'arricchimento dello stesso tramite RAG, come esplicato in precedenza. Di seguito, il processo di raffinamento subito dall'input fornito al modello (lo riportiamo solo per quest'ultima sezione in quanto sostanzialmente analogo per le altre)

1. Primo tentativo: risultato sotto forma di bullet point list

```
("Can you summarize a description for the job described with the following lists of work activities, skills and tasks? Each number before them is the importance of the activity, task or skill for the job. Work activities: + work_activities + " Skills: " + skills + "Tasks:" + tasks +

"This is about an open position for a " + job_name +

"These are some job description examples:" + retrieveKMostSimilar(model.encode(job_name))
```

2. Secondo tentativo: risultato poco chiaro e conciso

```
("Can you summarize a description for the job described with the following lists of work activities, skills and tasks? Each number before them is the importance of the activity, task or skill for the job. Work activities: + work_activities + " Skills: " + skills + "Tasks:" + tasks +

"This is about an open position for a " + job_name +

"These are some job description examples:" + retrieveKMostSimilar(model.encode(job_name)) +

" Avoid answering with a bullet point list.")
```

3. Terzo tentativo: risultato troppo poco discorsivo

```
("Can you summarize a description for the job described with the following lists of work activities, skills and tasks? Each number before them is the importance of the activity, task or skill for the job. Work activities: + work_activities + " Skills: " + skills + "Tasks:" + tasks +

"This is about an open position for a " + job_name +

"These are some job description examples:" + retrieveKMostSimilar(model.encode(job_name)) +

"Be clear and precise.” +

" Avoid answering with a bullet point list.”
```

4. Prompt finale

```
("Can you summarize a description for the job described with the following lists of work activities, skills and tasks? Each number before them is the importance of the activity, task or skill for the job. Work activities: + work_activities + " Skills: " + skills + "Tasks:" + tasks +

"This is about an open position for a " + job_name +

"These are some job description examples:" + retrieveKMostSimilar(model.encode(job_name)) +

"Be clear and precise. This should be read by a worker looking for a job, so you have to be clear. “

" Avoid answering with a bullet point list and be discoursive. Use no more than 20 lines.")
```

3.5 Comparazione tra job description e curriculum

Come effettuato per le job description, anche per il curriculum vitae nasceva la necessità di normalizzarlo, in quanto i curriculum possono essere strutturati in diversi modi e in diverse sezioni, e noi necessitavamo di una descrizione discorsiva e testuale.

Per effettuare ciò, abbiamo prima estrapolato il testo dal file contenente il curriculum, che verrà caricato dall'utente tramite l'apposito bottone visibile sull'interfaccia, e poi ci siamo affidati all'utilizzo di una richiesta API utilizzando il modello conversazionale di Meta LLaMa già citato. Per la scelta del prompt che restituisse in maniera più efficiente una rappresentazione precisa, sintetizzata e discorsiva del curriculum, abbiamo effettuato molteplici prove, di seguito ne riportiamo alcune:

1. Primo tentativo: risultato sotto forma di bulled point list

```
"Can you try to take this content related to the text extracted from a curriculum vitae and describe it trying to concentrate on the working skills in a detailed manner and avoiding section that not regards working skills?{file_content}"
```

2. Secondo tentativo: risultato troppo prolisso e poco preciso

```
“Can you try to take this content related to the text extracted from a curriculum vitae and describe it trying to concentrate on the working skills in a detailed manner and avoiding section that not regards working skills? please avoid pointed list and try to summarize{file_content}"
```

3. Terzo tentativo: risultato troppo riassuntivo e quindi anche poco descrittivo.

```
"Can you make a verbal description of the text provided in the following content? Summarize it and concentrate on working skills part, avoiding all the things not concerning working skill:{file_content}"
```

4. Prompt finale

```
"Can you take this content related to the text extracted from a curriculum vitae and resume it trying to focus on the working skills, working experiences and background in a detailed manner and avoiding section that do not regard those sections? please avoid answering with
```

`a bullet point list, be precise, clear and discoursive. Put only the Curriculum resume in the answer. The text to resume is the following: {file_content}"`

Fondamentale per la nostra applicazione è stato **all-MiniLM-L6-v2** che è un modello di embedding facente parte di **sentence-transformers**, un modulo Python specializzato nella costruzione e nella manipolazione di embedding

Dopo aver ottenuto una descrizione testuale del curriculum, abbiamo utilizzato il già citato modello di embedding **all-MiniLM-L6-v2**, facente parte del modulo di Python **sentence-transformers**. Con cui abbiamo derivato gli embeddings, per poi poterli confrontare con gli embedding derivati dalla job description, utilizzando la similarità del coseno tra essi.

Per calcolarla abbiamo utilizzato la funzione *cosine_similarity* della libreria scikit-learn, libreria open-source per il machine learning in Python. Per utilizzare la importiamo da *sklearn.metrics.pairwise*, dove *sklearn* è un' abbreviazione della libreria appena descritta, *metrics* è un modulo che contiene alcune funzioni per valutare prestazioni dei modelli di machine learning e il sottogruppo *pairwise* è specificatamente progettato per calcolare le distanze tra coppie di campioni, nel nostro caso, gli embeddings delle due descrizioni.

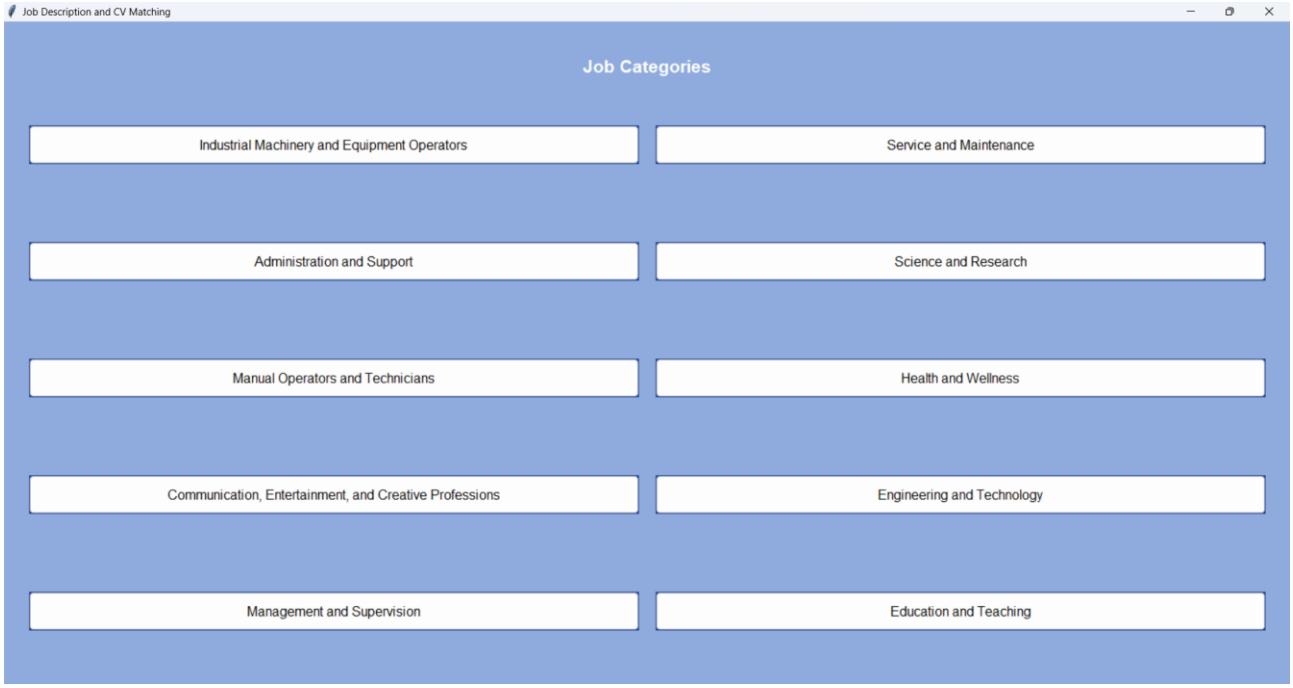
La cosine similarity prende in input due array di embeddings e applica la seguente formula ad essi:

$$K(X,Y) = \frac{< X,Y >}{(\|X\| * \|Y\|)}$$

4. Descrizione dei risultati

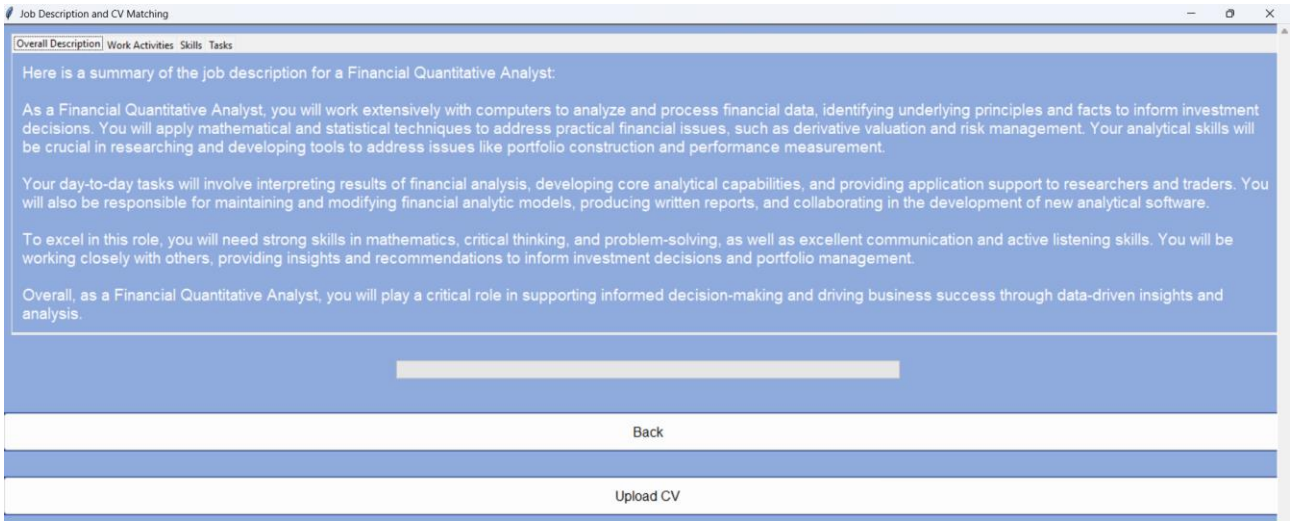
Durante la fase di prompt engineering, di cui abbiamo già parlato nei capitoli 3.4 e 3.5, abbiamo anche verificato se le descrizioni in uscita dal modello a seconda di un prompt o di un altro, rappresentassero più similarità in descrizioni di lavori simili al curriculum presentato. Per esempio abbiamo preso il curriculum di una persona con esperienze nell'informatica e nella programmazione, e abbiamo verificato la cosine similarity con un lavoro affine e che un lavoro non affine. In particolare, testando con la job description per un lavoro di "Computer Programmer", il risultato è positivo, con una similarità del coseno di 0.31, mostrato attraverso un popup che si genera a seguito della comparazione tra gli embedding, e che suggerisce all'utente come muoversi in relazione alla proposta di lavoro presa in esame. Se invece effettuo la medesima operazione, ma per un lavoro nell'ambito dei servizi e nella cura di sé stessi, per esempio "Manicure and Pedicure", mi dà un punteggio di 0.20, sconsigliandomi di applicare per tale lavoro.

Per rendere questa applicazione facilmente utilizzabile dall'utente abbiamo creato a GUI (Graphical User Interface) semplice e intuitiva, mostrata nelle seguenti immagini:

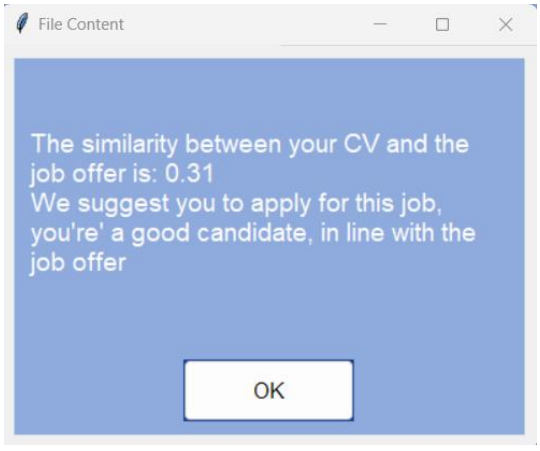


Nella prima pagina sono visibili tutte le categorie di lavori definite con l'operazione di clustering esplicata nella sezione 3.3. È possibile, attraverso un bottone, accedere alla lista di lavori facenti parte di tale categoria, per esempio i lavori per la categoria "Science and Research" si presenteranno come di seguito:

Come è possibile evincere dall'immagine soprastante, nella pagina mostrata è possibile accedere alla descrizione di ogni lavoro, e in alto è possibile utilizzare una barra di ricerca per cercare più rapidamente il lavoro desiderato, poi è visibile una barra che mostra il progresso del caricamento, in seguito alle operazioni necessarie per l'apertura delle varie sezioni della job description, visibili nella successiva immagine:



In questa pagina è possibile navigare tra le 4 sezioni che descrivono il lavoro, una descrizione generale, le attività che si andrebbero a svolgere, le skills e i tasks, sempre spiegati in maniera concisa e discorsiva con l'aiuto di chiamate API al modello precedentemente descritto.



Infine, è possibile caricare il curriculum con l'apposito bottone, ed è lì che avviene la comparazione, prima descritta, tra il curriculum appena caricato e il lavoro descritto nella seguente pagina. Per mostrare il risultato viene mostrata una finestra, come una sorta di popup, visibile nell' immagine sulla sinistra.

5. Conclusioni

Per quanto riguarda le tecniche utilizzate, notiamo dei punti di possibile miglioramento per quanto concerne il modello, come spiegato al capitolo 3.1, riguardano la scelta di esso, in quanto potremmo, anche se molto probabilmente a pagamento, accedere a modelli di livello superiore. L'altro possibile miglioramento riguarda senz'altro il prompt engineering effettuato, abbiamo cercato di migliorare il più possibile utilizzando un database per usufruire della tecnica RAG, esplicata nel capitolo 3.2, ma possibili miglioramenti potrebbero essere effettuati, per esempio suddividendo i task richiesti al modello in subtask di complessità minore, oppure dando una valutazione agli output del modello, utilizzando risposte gold-standard con cui valutare le risposte del modello.

In questo progetto abbiamo cercato di dimostrare che l'Intelligenza Artificiale può essere utilizzata per favorire il compito dell'essere umano, e non per sostituirlo.

In particolare, abbiamo effettuato il caso d'uso di una persona che ricerca lavoro, in cui l'AI può essere di aiuto per facilitare il compito dell'utente, andando ad offrire un feedback più oggettivo nel confronto tra l'esperienza dell'utente e la descrizione del lavoro preso in considerazione.