POLITECNICO DI TORINO

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Report di tirocinio curriculare

HR assistant basato su intelligenza artificiale

Tutori:
Roberto Maiocco
Guido Marchetto

Studente:
Simone Grassi

17 maggio 2020

Sommario

Nel contesto del tirocinio curriculare da 250 ore, inserito nel carico didattico, ho avuto la possibilità di lavorare presso l'azienda CGM Consulting; una realtà in rapida espansione che offre servizi di consulenza informatica e sviluppo software in ottica internazionale.

Dopo essere entrato in contatto con uno dei fondatori nonché CEO, Fabrizio Maiocco, durante l'evento Time For Jobs organizzato dal Politecnico, ho trovato fin da subito interessante l'azienda e la sua propensione verso nuove idee e progetti.

La proposta di tirocinio consisteva nello sviluppo di un software in Python, in grado di sfruttare una rete neurale che imparari dai numerosi curriculum vitae che ricevono ogni giorno, al fine di ricavare informazioni utili, catalogarli e mettere in evidenza le specifiche caratteristiche di ogni candidato, per poi proporlo in modo strutturato all'ufficio risorse umane il quale, in modo celere, potrebbe approfondire e contattarlo.

La proposta di tirocinio non solo era originale e mi permetteva di mettere alla prova le mie capacità tecniche ed impararne nuove nel ramo in cui ho deciso di continuare i miei studi, ma la natura semi-autonoma dello stesso mi permetteva di autogestirmi, organizzando un progetto dalle fasi iniziali alla consegna e relazionandomi con figure diverse quali stakeholder.

Dopo poche settimane, l'emergenza Covid ha costretto tutto l'ufficio a lavorare in modalità smart-working, questo non mi ha permesso di vivere a pieno la realtà aziendale, tuttavia grazie agli strumenti di gestione segnalazioni e repository che mi sono stati prontamente forniti, ho potuto terminare il progetto senza nessun inconveniente.

Il principale problema che ho dovuto affrontare, riguardava la tipologia e il numero di dati che devono essere analizzati, infatti i documenti in formato PDF sono portabili, ma contengono poche informazioni riguardo l'impaginazione e questo ostacola l'estrazione del testo, inoltre ogni rete neurale necessita di un dataset già catalogato, tuttavia il progetto era solo un'idea quindi esso non era presente.

Di conseguenza, dopo aver dedicato le prime settimane ad uno studio di fattibilità e delle tecnologie migliori, ho deciso di procedere realizzando due software: il primo è una versione semplificata con lo scopo di fornire un supporto iniziale all'ufficio HR permettendo di valutare le potenzialità dello

strumento e allo stesso tempo di etichettare i documenti in modo indiretto mentre si lavora, senza doversi occupare per ore a catalogare decine di file in modo fine a sè stesso; la seconda versione è un evoluzione che permette un'analisi molto più approfondita e potrà essere integrata quando il numero di esempi di addestramento sarà nell'ordine delle migliaia.

I risultati di questo approccio sono stati molto buoni sia in termini di feedback sia a livello tecnico, la rete semplificata riesce ad estrarre 200 chiavi informative con un'affidabilità di circa l'80%.

Nelle settimane di tirocinio ho potuto dedicare molto tempo a migliorare le mie capacità di programmazione in Python, prendendo familiarità con framework e librerie, ho potuto approfondire l'argomento Intelligenza Artificiale mediante ricerche, test e videocorsi ed infine ho imparato ad utilizzare molti software utili a livello aziendale come Anaconda, Atom, Jupiter, GitLab, Redmine ed approfittando della scrittura del presente elaborato ho avuto modo di impare Latex.

Glossario

Iperparametro sono i parametri di una rete neurale che devono essere scelti arbitrariamente sulla base di test, esperienza e linee guida, ad esempio il learning rate α o le epoche

Tensore struttura generica, su cui operano le reti neurali, ottenibile da un singolo spazio vettoriale a n dimensioni; a livello pratico si può definire come una matrice a n dimensioni

Overfitting Ridotto errore sistematico e alta varianza, sintomo di eccessivo adattamento al dataset di addestramento, la rete non impara dagli esempi ma solo a "barare" collegandoli al risultato corretto; le cause sono molteplici: set di addestramento troppo piccoli o grandi, iperparametri errati o rete troppo complessa.

Underfitting Alto errore sistematico e bassa varianza dovuto al dataset di addestramento troppo ridotto.

Dropout Eliminazione di una percentuale casuale dei nodi, da un iterazione alla seguente, per costringere gli altri a compensare ed evitare l'eccessivo adattamento al set di addestramento.

Dataset Insieme di dati sotto forma di tensore, tipicamente si parla di dataset di addestramento, ovvero un insieme di esempi (righe) che rappresentano le features (colonne) da cui la rete imparerà a predire il risultato (label).

Esempi Righe del dataset di addestramento, ogni esempio è caratterizzato da una serie di features rappresentative.

Label o etichetta, risultato previsto per un dato esempio, può essere un numero, una variabile ordinale o sconnessa, in tutti i casi viene trasformata in numeri nell'intervallo [0-1] in fase di pre-processing.

Dati sequenziali Serie di dati in cui il significato non dipende dal singolo elemento ma dal contesto, ovvero i dati precedenti e successivi; esempio tipico sono le frasi.

HR Human Resources, in lessico manageriale colui che si occupa delle risorse umane all'interno dell'azienda ad esempio della ricerca di nuovo personale.

CV Curriculum Vitae.

MCTS Monte-Carlo Tree Search, un algoritmo euristico di visita degli alberi utilizzato nel reinforcement learning quando non si hanno sufficienti informazioni per prendere una decisione.

Micro-mondi Problemi limitati e descritti da asserzioni che richiedono ragionamento per essere risolti.

Agente/Ambiente In ambito di reinforcement learning Agente è l'algoritmo che interagendo con l'Ambiente provoca un determinato effetto mediante la propria azione.

Principi di Asilomar [4] Vademecum con 23 principi per affrontare problematiche etiche, sociali, culturali e militari dell'IA, sottoscritto da oltre 800 esperti nel 2017, a seguito di un convegno mondiale promosso dal Future of Live Institute.

Adam Algoritmo di ottimizzazione per evitare la scomparsa del gradiente basato sull'unione del Nesterov Momentum e del learning rate adattivo, vedi 4.1.2

GD Gradient Descent, algoritmo iterativo alla base dell'apprendimento delle reti neurali, che sfrutta il segno del gradiente di una funzione di costo per giungere al minimo della stessa, vedi sottosezione 4.1.2

Rete ricorrente rete neurale che lavora su dati sequenziali mantenendo una memoria tra iterazioni successive, il numero di parametri da ottimizzare cresce poiché non sono solo spaziali ma anche temporali; esempi citati sono LSTM, GRU e biGRU.

Rete convoluzionale rete neurale utilizzata tipicamente per l'analisi di immagini o tensori di grandi dimensioni, sfrutta delle matrici chiamate filtri per ridurre la dimensionalità mantenendo l'informazione.

Strato nascosco Strati (layer) interni di una rete neurale, se presenti in numero maggiore a 1 la rete viene definita profonda e si può parlare di deep learning.

Strato Denso Strato generico composto da un numero arbitrario di nodi e legato agli strati precedenti e successivi mediante archi pesati, in keras sono nominati Dense.

Embedding Rete neurale ricorrente che riceve in ingresso un dato sequenziale e impara ad assegnare una codifica di lunghezza fissa inferiore, la logica con cui questa viene scelta non è nota ma si basa sulla distanza del coseno, ad esempio in due frasi, parole simili come *re* e *sovrano* avranno una codifica simile e opposta a *plebeo*.

Scala Likert Scala che va da 'Per niente d'accordo' a 'Molto d'accordo'

Psicometria Indagine psicologica tendente alla valutazione quantitativa dei comportamenti.

Mining del testo Estrazione di informazioni o stringhe da documenti in vari formati testuali, esempio PDF.

Regressione lineare La regressione formalizza e risolve il problema di una relazione funzionale tra variabili misurate sulla base di dati campionari estratti da un'ipotetica popolazione infinita.

Regressione logistica Nota come Logit, in statistica e in econometria, è un modello in grado di stabilire la probabilità con cui un'osservazione può generare uno o l'altro valore della variabile dipendente; può inoltre essere utilizzato per classificare le osservazioni, in base alla caratteristiche di queste, in due categorie .

Pre-processing Nel contesto delle reti neurali si intende l'insieme di operazioni atte a rendere i dati compatibili con l'ingresso delle stesse, esempi tipici sono la normalizzazione, la codifica di parole, l'inserimento di dummies.

Dummies Features fantoccio necessarie per codificare delle label sconnesse (es. colori, specie animali) mediante valori booleani.

Token Corrispondente a parole nel senso sintattico del termine, ad esempio Los Angeles viene considerato un unico token, è una parte fondamentale del pre-processing del testo poichè da ogni token viene ricavato il lemma (forma priva di declinazione) e assegnato un codice.

GloVe Database di pesi pre-calcolati per strati di Embedding, contiene 400 000 token unici ricavati da Wikipedia con la relativa codifica su 50, 200 o 300 dimesnioni.

LSTM Long-Short Term Memory, tipico layer di una rete ricorrente adatta ad evitare la scomparsa del gradiente su sequenze lunghe.

GDPR General Data Protection Regulation, regolamento europeo per la protezione dei dati sensibili

Indice

Capitolo 1

Presentazione dell'azienda

CGM Consulting nasce dalla passione di quattro amici, che unendo le loro competenze informatiche, commerciali ed amministrative, hanno voluto trasformare la loro passione in un'azienda sul loro territorio di origine, Torino, e ben presto hanno iniziato una rapida espansione verso altre città italiane. L'azienda offre servizi di consulenza informatica e sviluppo software in ambito web sia front-end che back-end, desktop, server e mobile.

Al momento conta clienti del rango di *Reply*, *Engeneering*, *Almaviva*, *Magnolia* e la rapida espansione in settori come Finance, Media & Comunication, Health & Care, Managment Software in diverse sedi li porta a cercare una continua espansione del team mediante un ufficio risorse umane dedicato.

Tra i progetti più interessanti si possono citare lo sviluppo del software di voting per *XFactor italia* e *Italian's Got Talent*, le campagne ADV web di *Ikea* ed *Enel Green Light*, il database sulle malattie rare per *ASL Piemonte* e componenti per il sito di *Radio R101*.

Nella scelta dell'azienda in cui fare il tirocinio preferivo evitare realtà di grandi dimensioni, poiché spesso si viene affidati a specifiche parti di progetti estesi senza la possibilità di mettere in gioco le proprie capacità di problem solving e soprattutto, senza avere un contatto diretto con altri membri dell'azienda e tanto meno con la direzione.
Trovo difficile emergere in quel contesto e a mio parere il tirocinio rischiava di essere un'attività fine a sè stessa, per questo motivo ho trovato subito interesse per l'offerta di CGM Consulting, la quale oltre che apparirmi come un contesto lavorativo molto sereno, proponeva una cosa che difficilmente le aziende fanno: affidare direttamente un progetto ad un tirocinante e questa era un'ottima opportunità.

Capitolo 2

Problema ed obiettivi

La rapida espansione dell'azienda la porta ad una ricerca continua di nuove figure da includere nel team, tuttavia quelle degli sviluppatori software sono competenze estremamente richieste negli ultimi anni e di conseguenza tendono ad rimanere disponibili per poco tempo.
L'ufficio HR dispone di una rete di contatti molto ampia, distribuita nelle principali città italiane e composta da istituti tecnici, università e associazioni quali Almalaurea; questo gli permette di avere accesso ad una quantità molto elevata di curriculum vitae misti, che tuttavia devono essere analizzati uno alla volta nel minor tempo possibile per individuare e contattare i candidati migliori in tempo.

In questo scenario risulta evidente come la potenza e velocità di calcolo di un computer potrebbero costituire uno strumento utile; dopo un'analisi ed un confronto con l'HR manager, le funzionalità principali sono risultate:

1. Filtro e catalogazione dei cv, al fine di scartare fin da subito figure per nulla compatibili con le necessità e rendere più efficiente la ricerca, effettuandola solo tra i candidati adatti ad un determinato ruolo.
   Ad esempio se si ricerca una segretaria per l'ufficio vendite, risulta controproducente controllare anche tutti i cv di sviluppatori web.

2. Ordinamento secondo l'attinenza alla ricerca, questo permette di porre subito all'attenzione le figure più adatte al ruolo richiesto, in modo da poterli contattare nel minor tempo possibile.

A queste funzionalità si aggiunge anche che il programma deve poter lavorare in maniera autonoma sui documenti di ingresso, adattandosi non solo al contesto aziendale (in vista di una possibile messa sul mercato del prodotto stesso), ma anche alle attitudini personali del HR trovando un giusto compromesso fra le due linee guida.

In tale contesto, è evidente che l'intelligenza artificiale ed in particolare le reti neurali trovino la loro perfetta applicazione, tuttavia si presentano due principali ma fondamentali problematiche.

Le reti neurali lavorano come black box [1] e questo nel tempo ha portato a diversi scandali: secondo una ricerca della *Harvard Business School*[1] le persone tendono a seguire più volentieri i consigli dettati da un'algoritmo, ma come osserva Francesco Marconi [2]

> La faziosità che è propria dell'essere umano può essere moltiplicata a dismisura dalle macchine. È questo il vero rischio quando si parla di Intelligenza artificiale.

Non sono pochi i casi di discriminazioni portate avanti in modo completamente invisibile da algoritmi sessisti o razzisti (per quanto sembri fantascientifico attribuire questi termini a delle macchine), ad esempio gli algoritmi di computer vision di IBM e Microsoft tendevano a non riconoscere come tali (-35%) i visi delle donne di pelle scura, oppure l'algoritmo alla base del motore di ricerca di Google mostrava proposte di lavoro più remunerative agli uomini, o ancora in America alcune corti si affidano a software per determinare le sentenze e le persone provenienti da aree malfamate finiscono in prigione con maggiore probabilità, indipendentemente dal reato.
Questo problema è limitato finché il software rimane all'interno di GCM Consulting ed è estremamente improbabile per l'algoritmo semplificato sezione 5.2, poiché le informazioni vengono filtrate in fase di pre-processamento, tuttavia può diventare rischioso nell'algoritmo evoluto, il quale analizza senza limiti tutto il contenuto del curriculum e potrà essere distribuito ad altre aziende con accesso a dati potenzialmente illimitato. Per quanto questa osservazione possa sembrare futuristica, costituirebbe un grosso danno di immagine per l'azienda e per questo motivo è fondamentale un algoritmo che rimanga sempre semi-autonomo con la possibilità continua di apprendere ricevendo in modo intuitivo dalla figura umana delle osservazioni.

Il secondo problema riguarda i dati di addestramento, infatti la richiesta era di utilizzare i PDF senza nessuna manipolazione da parte dell'HR, tuttavia questo formato consente un'ottima portabilità a discapito del contenuto informativo, in particolare riguardante l'impaginazione, inoltre l'addestramento di reti neurali con un tale livello di intelligenza richiede un dataset nell'ordine di $10^3 \div 10^5$ esempi classificati manualmente ma, essendo in una fase iniziale, questo non era disponibile.

[1] E' possibile comprendere solo l'input e l'output della rete neurale, il ragionamento fatto e le assunzioni rimangono completamente ignote

[2] Scrittore e giornalista italiano esperto di intelligenza artificiale

Capitolo 3

Descrizione del tirocinio

Il tirocinio si è svolto nel periodo dal 02-03-2020 al 06-05-2020 per un monte ore complessivo di 250 ore, l'impegno era organizzato in tre giorni a settimana, da otto ore l'uno.
Le prime due settimane sono state svolte nella sede centrale di Torino, ho potuto conoscere il personale dell'ufficio e mi sono state date le indicazioni sul progetto che avrei dovuto sviluppare in collaborazione con il mio tutor aziendale Roberto Maiocco e uno sviluppatore Ismail Nasry.
Essendo il tutto soltanto un idea, la prima attività svolta è stata una ricerca delle soluzioni presenti sul mercato per effettuare uno studio di fattibilità.
Questo mi ha permesso non solo di fissare degli obiettivi concreti a breve e lungo termine, ma anche di discutere con gli utilizzatori finali delle funzionalità che servivano maggiormente.

Riassumendo i risultati, esistono delle soluzioni che dimostrano sia la fattibilità dell'approccio sia il trend in crescita dell'interesse in tecnologie di questo tipo, tuttavia i software effettivamente completi non sono molti poiché la maggior parte sono più assimilabili a motori di ricerca.
Studiando le possibili applicazioni delle reti neurali in questo contesto sono emerse idee interessanti, ad esempio sul curriculum i candidati tendono a presentarsi al meglio, nessuno riporterebbe i propri difetti mentre si propone per un lavoro, tuttavia l'accesso ad una grande mole di esempi da parte di una rete neurale gli permette di fare ciò che un HR acquisisce con l'esperienza, ovvero "leggere tra le righe", questo tipo di analisi prende il nome di LIWC[1] e permette di ricavare tratti caratteriali dal modo in cui una persona scrive.

Dalla terza settimana l'emergenza Covid ha costretto il personale al lavoro da casa, così mi sono stati forniti una cartella di repository su *Gitlab* per i miei lavori e *Redmine*, uno strumento per gestire segnalazioni, richieste e segnare le mie ore di lavoro per una data attività, oltre che assistenza Skype per qualsiasi richiesta.
In questo modo ho iniziato a fare una ricerca sui framework e librerie uti-

[1] Da leggersi liùc, Linguistic Inquiry and Word Count

lizzabili, effettuando diversi test su dei campioni di curriculum; inoltre per riuscire a sviluppare un software valido e sfruttare al meglio il tirocinio, dovevo approfondire le mie conoscenze nel campo dell'intelligenza artificiale, così ho iniziato dei corsi online su Cursera e Udemy, che ho portato a termine nelle settimane successive: *Deep Learning e reti Neurali*, *Natural Language Processing* e *Computer Vision* forniti da Profession.IA e *Machine Learning* della Standford University.

In contemporanea, ho iniziato a sviluppare la versione semplificata del software poiché si basa su dati in ingresso pre-processati su 200 metriche chiave, la conseguente perdita di informazione mi ha permesso di addestrare parecchie volte le reti sul mio portatile in tempi accessibili, oppure facendo ricorso al cloud computing offerto da *Google Collaboratory*, su un dataset di ridotte dimensioni.
In questo modo mi sono potuto concentrare solo sugli iperparametri della rete e sulla struttura dell'algoritmo wrapper, queste strutture sono adattabili anche all'algoritmo finale che differisce solo per la complessità della rete e la sua conseguente gestione.

Infine, una volta terminata la versione semplificata, ho realizzato una serie di classi e metodi per l'integrazione futura della seconda versione e mi sono dedicato alla scrittura della documentazione per lo sviluppatore che avrebbe ereditato il progetto.
Approfittando di questa occasione, ho deciso di sfruttare le ultime giornate di lavoro per imparare a scrivere documenti il LATEX, competenza che da tempo volevo approfondire ma che ero costretto a rimandare per mancanza di tempo.

Capitolo 4

Analisi preliminare e studio di fattibilità

4.1 Keras e richiami teorici sulle reti neurali

4.1.1 Cenni storici sull'intelligenza artificiale

Da quando il celebre software di DeepMind *Alpha-Go*, nel 2016, sconfisse il campione del mondo Sud-Coreano Master Lee Se-dol nel gioco da tavolo Go, il termine Intelligenza Artificale (IA) ha acquisito popolarità esponenziale, come dimostra l'andamento della parola su Google Trends che ha quadruplicato la sua frequenza.
Alpha-Go è basato sulla combinazione di un algoritmo tradizionale di visita e ricerca su un albero MCTS[1] con profonde reti neurali convolutive; questa rete venne addestrata mediante un sistema di rinforzo Attore-Critico, ovvero fatto giocare contro versioni di sè stesso per molto tempo.

Intelligenza artificiale è un termine spesso abusato e non è facile darne una definizione rigorosa, secondo *Marco Somalvico*[2]:

Definizione 1. *L'intelligenza artificiale è una disciplina dell'informatica che studia i fenomeni teorici, le metodologie e le tecniche che consentono la progettazione di sistemi hardware e software capaci di fornire all'elaboratore elettronico prestazioni che, a un osservatore comune, sembrerebbero essere di pertinenza esclusiva dell'intelligenza umana.*

Tuttavia anche il concetto di intelligenza umana è labile, infatti anche tra esseri viventi è difficile stabilire quali sono intelligenti e se l'imitazione del

[1] Monte-Carlo Tree Search, un algoritmo di visita degli alberi utilizzato nel reinforcement learning quando non si hanno sufficienti informazioni per prendere una decisione

[2] Ingegnere italiano specializzato nell'intelligenza artificiale

nostro comportamento nasconde un processo logico analogo[3], inoltre possiamo noi definirci intelligenti?[4]
Si può definire intelligente usando una misura di somiglianza al comportamento umano ideale (razionale) e di efficacia:

1. Agire umanamente: il risultato dell'operazione compiuta dal sistema intelligente non è distinguibile da quello svolto da un umano.

2. Pensare umanamente: il processo che porta il sistema intelligente a risolvere un problema ricalca quello umano.

3. Pensare razionalmente: il processo che porta il sistema a risolvere un problema è un procedimento logico.

4. Agire razionalmente: il processo porta al miglior risultato atteso con le informazioni a disposizione.

La disciplina si ritiene abbia avuto origine nel 1956 al Dartmouth College durante un convegno con le più illustri figure del campo della computazione, un team di 10 ricercatori avrebbe dovuto creare in due mesi una macchina in grado di simulare in ogni aspetto apprendimento ed intelligenza umana, in questa occasione venne anche coniato il termine intelligenza artificiale.
Le iniziali previsioni erano estremamente ottimistiche, si utilizzavano questi algoritmi per risolvere problemi di micro-mondi[5] e si stimò che entro 10 anni una macchina avrebbe battuto il campione di scacchi (cosa effettivamente successa, ma dopo 40 anni), l'approccio era tuttavia fondamentalmente sbagliato, infatti si credeva che per scalare la complessità dei problemi bastasse scalare hardware e memoria.

Il primo sistema di intelligenza artificiale utilizzato in ambito commerciale fu R1 della Digital Equipment nel 1982, lo scopo era fornire supporto nella gestione degli ordini di computer e fu in grado di far risparmiare alla compagnia oltre 40 milioni l'anno, nel 1988 il mercato dell'IA valeva nell'ordine dei miliardi di dollari.

Al giorno d'oggi, dopo lo sviluppo di sistemi più articolati quali la back propagation, ci sono programmi in grado di battere i giocatori di scacchi (Deep Blue), svolgere missioni spaziali (Remote Agent), guidare auto, tecnofinanza o più semplicemente termostati che predicono i cambi di temperatura e agiscono in anticipo.
Nel 2017 a seguito di un convegno mondiale promosso dal *Future of Live Institute* è stato redatto un ampissimo vademecum con 23 principi per affrontare problematiche etiche, sociali, culturali e militari dell'IA, sottoscritto

[3] Il problema è parso evidente a seguito dello sviluppo del test di Turing nel 1950 sulla rivista Mind

[4] Per approfondimenti di carattere filosofico e proprietà emergenti vedi [3]

[5] Problemi limitati e descritti da asserzioni, che richiedono ragionamento per essere risolti

da oltre 800 esperti prende il nome di Principi di Asilomar[4] ed è adottato anche dall'UE.

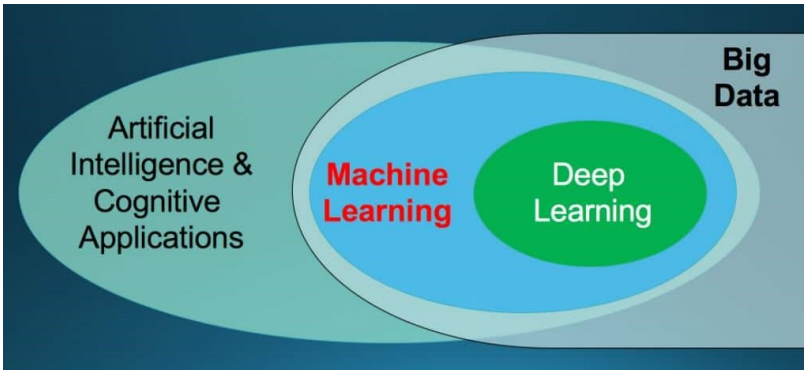

Figura 4.1: Rappresentazione insiemistica delle categorie di intelligenza artificiale

Come precedentemente accennato, da quando l'intelligenza artificiale è entrata a far parte della cultura di massa i termini sono spesso utilizzati in contesti non adatti o con accezioni fuorvianti, per fare chiarezza:

Il *Machine Learning*, o apprendimento automatico, è una delle possibili strade per l'attuazione dell'intelligenza artificiale, per questo può esserne considerato un sottogruppo che si concentra sulla capacità della macchina di ricevere una serie di dati e apprendere da essi, modificando gli algoritmi.
L'educazione, in gergo addestramento, richiede un enorme quantità di dati ed un algoritmo efficiente in grado di convergere ad una situazione stabile ed affidabile, il machine learning quindi automatizza la costruzione di modelli analitici usando reti neurali, modelli statici e ricerche operative per trovare delle relazioni nascoste fra i dati; un tipico esempio è la visione artificiale.

Il *Deep Learning*, o apprendimento approfondito, è uno degli approcci al machine learning che prende spunto dalla struttura del cervello, ovvero l'interconnessione di neuroni, oppure da modelli DAG (grafici aciclici diretti) come nel clustering o le reti bayesiane.
L'apprendimento profondo usa enormi modelli di reti neurali con diverse unità di elaborazione, questo unito a grandi quantità di dati e tecniche di addestramento permette di ottenere modelli molto complessi e raffinati, usati per esempio nello speech e image recognition.

Il *Reinforcement Learning*, o apprendimento per rinforzo, è una tecnica che punta a realizzare agenti autonomi in grado di scegliere le azioni da compiere per il conseguimento di determinati obiettivi e tramite l'interazione con l'ambiente in cui sono immersi.
L'apprendimento per rinforzo è uno dei tre paradigmi base, insieme ad apprendimento supervisionato e non supervisionato, e permette all'agente di imparare autonomamente continuando a provare e commettendo errori o

raggiungendo un premio (in gergo ricompensa), la tecnica più comune di addestramento consiste nel creare l'ambiente e lasciare che due agenti si sfidino.

4.1.2 Richiami teorici alle reti neurali

Il termine rete neurale deriva dalla palese analogia con la struttura del cervello umano, in particolare un nodo della rete ha una struttura molto simile a quella di un neurone biologico, benché molto semplificata.

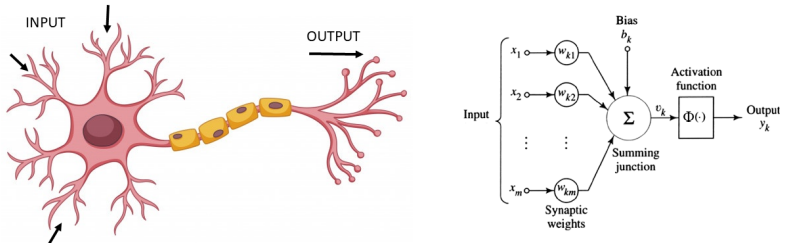

Figura 4.2: Analogia tra neurone umano e nodo di una rete neurale

Qualunque sia la natura dei segnali di ingresso, essi vengono codificati in numeri decimali da zero a uno[6], in seguito il neurone applica la seguente relazione

$$y_k = \Phi(\sum_i (w_{ki} * x_i) + b_k)$$

La funzione Φ è detta *funzione di attivazione* e riveste una fondamentale importanza per il funzionamento e la natura stessa del neurone, ipotizzando che Φ sia la funzione identità e di avere un solo ingresso è possibile vedere che l'uscita sia una retta che tende ad interpolare i campioni forniti nel dataset di addestramento.

[6]Se i dati sono ordinali (es. taglie dei vestiti) viene fatta una mappatura su valori ordinati, se sono sconnessi (es. colore) viene creata una features per ogni valore e assegnato un valore booleano (variabili dummies), mentre se si tratta di numeri vengono normalizzati

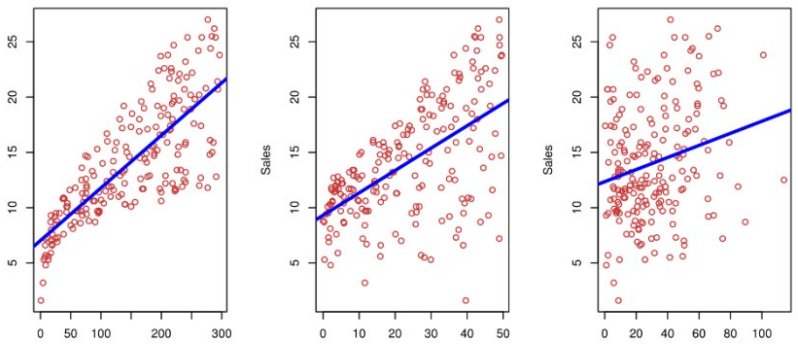

Figura 4.3: $y = w * x + b$ con in rosso i valori forniti dal dataset di addestramento

In realtà la funzione di attivazione può avere forme ben più complesse, ad esempio la *funzione identità* viene utilizzata in caso di regressione lineare, la *Leaky Relu* $\Phi(z) = max(\alpha * z, z)$ o la *tangente iperbolica* $\Phi(z) = (1 - e^{-2z})/(1 + e^{-2z})$ vengono usate nei layer interni, mentre la *sigmoide* $\Phi(z) = 1/(1 + e^{-2z})$ per la regressione logistica.

Quando si parla di *regressione lineare* si intende che l'uscita è un valore continuo nella scala [0,1], mentre con il termine *regressione logistica* si indica che y_k rappresenta la probabilità di appartenenza ad una data classe (interpretata come valore booleano 0/1).

Appare evidente come il problema principale di questa logica sia assegnare i valori corretti ai pesi, questa è un operazione tutt'altro che semplice per la loro mutua influenza e per il numero molto elevato che raggiungono all'interno di reti anche semplici.
La soluzione sancì il vero e proprio inizio dell'era dell'intelligenza artificiale, si tratta della *back propagation* ed in particolare dell'algoritmo di *gradient descent*.
Rappresentando tutte le variabili come vettori e matrici e ponendo $h(x) = \vec{w} * \vec{x} + b$ possiamo definire la *funzione di costo* come l'errore quadratico medio[7]:

$$J(\theta_0, \theta_1) := \frac{1}{2m} \sum_{i=1}^{m} (h(x)^{(i)} - y^{(i)})$$

Lo scopo dell'algoritmo è trovare i valori di θ tali da minimizzare la funzione di costo e per farlo utilizza due iperparametri:

[7] Osservazione sulla notazione usata nella trattazione: il simbolo di vettore $\vec{z}$ viene omesso poiché tutte le variabili in gioco sono in forma matriciale (più precisamente tensori) di dimensioni coerenti, inoltre la forma $z^{(i)}$ indica l'i-esima riga della matrice z mentre z_j è la j-esima colonna

1. α detto *learning rate* ovvero la dimensione del "salto" compiuto dall'algoritmo per avvicinarsi al minimo, un valore troppo piccolo non porterà alla convergenza, un valore troppo alto lo renderà divergente.

2. *epochs* o epoche, numero di iterazioni scelto arbitrariamente dopo le quali si può considerare raggiunto il minimo.

$$\theta : \min(\bigtriangledown J(\cdot))$$

$$\forall epochs\, \theta := \theta - \alpha \cdot \bigtriangledown J(\cdot)$$

$$\theta_j = \theta_j - \frac{\alpha}{m} \cdot \sum_{i=1}^{m} (h(x)^{(i)} - y^{(i)}) \cdot x^{(i)}$$

da notare che := significa uguaglianza, mente = viene usato con il significato di assegnazione, inoltre l'assegnazione dei valori di θ_j deve essere contemporanea a tutti.

Per completezza si riporta che esiste anche una formula analitica che benché non richieda iterazione sulle epoche ha una complessità $O(n^3)$ contrapposta a quella $O(n^2)$ del gradient descent.

$$\theta := (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

L'approccio scelto per il presente progetto è quello dell'*apprendimento supervisionato*, ovvero si dispone di dati di ingresso (esempi) la cui uscita è nota a priori (etichettati), questi compongono il *dataset* (set) di addestramento; la rete neurale verrà fatta lavorare per un numero di epoche arbitrario sui dati del dataset in modo che essa provi a prevedere un uscita, la confronti con quella attesa e modifichi i propri pesi, di fatto imparando.

Il gradiente $\bigtriangledown J(\cdot)$ assume valori sempre più piccoli man mano che ci si avvicina al minimo, tuttavia a causa della precisione finita del calcolatore esso potrebbe diventare nullo, bloccando l'apprendimento per tutte le epoche successive; questo problema è detto *scomparsa del gradiente* e deve essere evitato agendo sugli iperparametri mediante test.

E' da notare che parlando di minimi non si è accennato al fatto che essi siano locali o assoluti, la scelta della funzione di costo (errore quadratico medio non è l'unica) influisce pesantemente sulla loro presenza, ma è comunque necessario implementare algoritmi di ottimizzazione: la scelta ricadde quasi sempre su *Adam*[8] poichè esso racchiude il *learning rate adattivo*, ovvero α che diminuisce avvicinandosi al minimo, e il *Nesterov Momentum*, concettualmente immaginabile come lo sfruttare la quantità di moto accumulata durante una discesa per superare un'avvallamento locale.

Il modo in cui ci si approccia ai dati contenuti nel dataset di addestramento determina prestazioni e accuratezza dei risultati:

[8] Per reti convoluzionali e ricorrenti su sequenze lunghe si dimostra sperimentalmente che il migliore sia RMSprop derivante da Adagram

Full Batch GD Ad ogni epoca eseguo il gradient descent (GD) su ogni esempio presente nel dataset

Stochastic GD Ad ogni epoca eseguo il gradient descent (GD) su un solo esempio casuale

Mini Batch GD è la via di mezzo tra i due precedenti approcci, ad ogni epoca eseguo il GD su un sottoinsieme di dimensione $\sim 32\%$ del dataset

La scelta dell'approccio è fondamentale non solo perchè influisce pesantemente sui risultati in termini di accuratezza, ma allo stesso tempo espone a problemi di overfitting[9] e underfitting[10].

La complessità delle reti realizzabili con nodi concettualmente così semplici è notevole, infatti è possibile sfruttare la velocità di calcolo di un computer (in particolare GPU e TPU) per addestrare modelli con decine di migliaia di nodi e realizzare strati sempre più complessi.
Per citare alcuni esempi: le *reti convoluzionali*, utilizzate in particolare per lavorare su immagini considerare come tensori, ottengono i dati di ingresso applicando secondo criteri logici delle matrici dette filtri con lo scopo di ridurre la dimensionalità dei dati senza perdere il contenuto informativo; oppure le *reti riccorrenti*, che verranno trattate nello specifico nell'applicazione pratica della sezione 5.3 parlando di strati di Embedding 5.3.

4.1.3 Analisi delle tecnologie disponibili

Un'importante analisi preliminare riguarda le tecnologie ed in particolare i framework e linguaggi che conviene utilizzare infatti, benché non vi fossero particolari specifiche riguardanti il prototipo, dare fin da subito una determinata struttura al software avrebbe ridotto tempi e incompatibilità future.
Il software deve girare su server per permettere ad un'unica rete neurale di servire tutti gli utenti ed imparare da essi, inoltre lo stesso deve fornire l'hardware necessario e la compatibilità per eseguire l'addestramento periodico.
Il linguaggio di programmazione classico, nonché il più adatto ad applicazioni di questo tipo, è *Python*, mentre per quanto riguarda il framework da utilizzare il panorama è molto vasto:

Tensorflow prodotto da Google, contiene modelli statici adatti a vari utilizzi, dalla prototipazione alla produzione, viene utilizzato da AirB&B e DropBox.

[9] Ridotto errore sistematico ma alta varianza, sintomo di eccessivo adattamento al dataset di addestramento, la rete non impara dagli esempi ma solo a "barare" collegandoli al risultato corretto; le cause sono molteplici: set di addestramento troppo piccoli o grandi, iperparametri errati o rete troppo complessa

[10] Alto errore sistematico e bassa varianza dovuto al dataset di addestramento troppo ridotto

Pytorch prodotto da Facebook, include un altro famoso framework, ovvero Coffe2, e dispone di molti modelli computazionali dinamici.

CNTK prodotto da Microsoft, non è totalmente opensource e viene utilizzato in servizi come Skype, Xbox e Cortana.

MXnet prodotto da Amazon e utilizzato nei servizi AWS.

CoreML prodotto da Apple, non è propriamente paragonabile agli altri poichè permette solo di far girare reti già addestrate su dispositivi Apple.

Theano nato in ambito accademico con caratteristiche simili a Tensorflow, è stato abbandonato da anni perchè non era possibile reggere il confronto con gli investimenti dei big del settore tecnologico.

Keras si tratta di un API di alto livello che funge da wrapper per altri servizi come Theano, CNTK, Tensorflow, proprio in quest'ultimo è stato integrato nel 2018.

La scelta è ricaduta su *Keras*[9] (e di conseguenza anche Tensorflow), le reti neurali vengono considerate come strutture `Model` o `Sequential` a cui si possono aggiungere layer con specifiche funzionalità; la semplicità d'uso, la curva di apprendimento molto rapida e la varietà di funzioni disponibili lo rendono la scelta migliore per questa applicazione.
Inoltre, è possibile addestrare la rete sfruttando CPU, GPU e TPU[11] e se in futuro si decidesse di abbandonare Python e il suo server framework Flask è possibile tradurre il codice in Javascript ed utilizzare ad esempio Node.js.

4.2 Modello dei Big 5

Dai testi scritti dal candidato non è possibile comprendere solo le sue hard skills, formazione o dati anagrafici, infatti il modo in cui si scrive fa trapelare moltissimi tratti della nostra personalità; ovviamente quello del curriculum è un contesto molto condizionante a causa di formalismi, linguaggio e contenuti, ma un'analisi attenta riesce comunque a ricavare delle informazioni.
A differenza dell' approccio basato sui questionari, la LIWC sfrutta un dizionario di parole classificate secondo semantica e psicolinguistica per estrarre la frequenza con cui queste vengono utilizzate e da essa ricavare delle conclusioni.

La teoria dei *Big Five di Robert R. McCrae e Paul T. Costa* è una tassonomia dei tratti della personalità, che si è distinta dalla moltitudine di modelli incentrati su un approccio nomotetico [12] e al momento risulta tra le

[11] Rispettivamente Control Processing Unit, Graphics Processing Unit e Tensor Processing Unit, quest'ultima è otimizzata per lavorare con i tensori

[12] Basato su principi fissi, analoghi a leggi universali

più testate sia a livello teorico che empirico.
Le basi su cui si fonda questa teoria sono:

- L'approccio fattoriale che identifica le dimensioni caratterizzanti proposto da *Hans Eysenck*, in psicometria[13] permette di evidenziare tratti latenti non direttamente misurabili attraverso indicatori osservabili.

- La teoria della sedimentazione linguistica elaborata da *Raymond Cattell*, che individua nel vocabolario della lingua quotidiana un serbatoio di descrittori delle differenze individuali.

Da questi principi vengono definite 5 grandi dimensioni di personalità: *estroversione-introversione*, *gradevolezza-sgradevolezza*, *coscienziosità-negligenza*, *nevroticismo-stabilità emotiva*, *apertura-chiusura mentale*.
Esse derivano da studi psicolessicali secondo cui l'uomo ha codificato in forma verbale le esperienze significative per la comunità, tra le quali le differenze individuali; è possibile verificare quanto sopra affermato soffermandosi sulle parole utilizzate quotidianamente per descrivere le persone, esse corrisponderanno a queste macro-aree.

Le macro-aree rappresentano anche il punto di convergenza delle strutture di altri modelli a tratti come: test della personalità, 16PF, GZTS, EPQ, CPS, CPI, infatti tutte queste considerano i poli opposti (come introversione ed estroversione) come estremi di una linea su cui la popolazione si distribuisce in forma gaussiana; mentre teorie come MBTI, basate su tipi, li considera come poli di attrazione distinti e di conseguenza si sviluppa una distribuzione di Bernoulli.
Il test dei Big Five[7], grazie alla sua attendibilità, viene utilizzato in contesti organizzativi mediante la compilazione di questionari strutturati attraverso una Scala Likert[14] o attraverso la valutazione della condotta in contesti di simulazione (Assessment center).

E' interessante considerare come le 5 dimensioni emergano solo da alcuni contesti linguistici e culturali, come Italia, USA, Germania, Giappone ec. mentre in altri, come la Cina, essi non emergano e la teoria viene applicata con alcune modifiche.

La principale critica mossa verso questo modello è che i risultati si basano su costrutti psicologici dall'eccessiva eterogeneità, per questo motivo non è ancora da considerarsi una teoria unificante né una risposta a quesiti nozionistici, ma uno strumento di supporto in determinati contesti ed ambienti, motivo per cui è più opportuno definirla 'Modello' piuttosto che 'Teoria'.

[13] Indagine psicologica tendente alla valutazione quantitativa dei comportamenti
[14] Scala che va da 'Per niente d'accordo' a 'Molto d'accordo'

4.2.1 Interpretazione delle 5 dimensioni della personalità

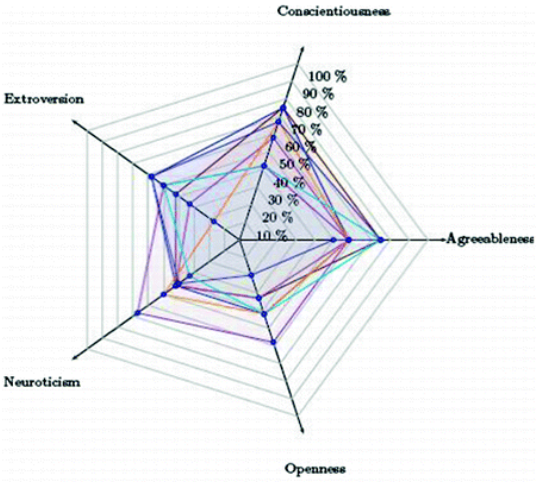

Figura 4.4: Esempio di rappresentazioni della personalità mediante il modello big five

ESTROVERSIONE (estroversione-introversione): detta anche energia, rappresenta la socialità e la tendenza a cercare stimoli in compagnia di altri, oltre che di loquacità; alta estroversione caratterizza caratteri alla ricerca di attenzioni e prepotenti, mentre bassa è associata a egocentrismo, riservatezza, comportamenti non adatti a ruoli dominanti in contesti sociali.

AMICALITA' (gradevolezza-sgradevolezza): qualità e quantità di relazioni interpersonali positive che la persona è portata a intraprendere, sue sottodimensioni sono la cooperatività ed empatia, inoltre risulta essere indice di accuratezza metodologica, affidabilità, scrupolosità, perseveranza e volontà di successo; valori bassi di questo indice invece caratterizzano persone competitive ma impegnative e polemiche.

COSCIENZIOSITA' (coscienziosita'-negligenza): tendenza ad essere organizzato ed affidabile, provvisto di autodisciplina, pianificazione piuttosto che comportamento spontaneo; livelli bassi caratterizzano comportamenti spontanei ma spesso trascurati o inaffidabili.

NEVROTICISMO (nevroticismo-stabilita' emotiva): grado di resistenza allo stess emotivo (resilienza), soprattutto per quanto riguarda ansia, depressione e vulnerabilità; elevata stabilità si manifesta in persone calme, disinteressate e indifferenti, mentre alti valori indicano individui dinamici, instabili e insicuri.

APERTURA ALL'ESPERIENZA (apertura-chiusura mentale): apprezzamento per l'arte, avventura, idee creative, curiosità, è indice di fantasia e indipendenza, ma anche di mancanza di messa a fuoco e attitudine a impegnarsi in comportamenti a rischio; i soggetti con bassa apertura mentale cercano di ottenere l'adempimento con perseveranza e pragmatismo.

E' importante interpretare in modo corretto le informazioni ottenute dal test, infatti non esistono caratteristiche positive o negative, vengono disposte come due poli della stessa retta solo per questioni visive, inoltre i soggetti che dimostrano una distribuzione gaussiana su tutti gli aspetti, senza eccedere in un singolo fattore, sono considerati adattabili, moderati e ragionevoli, ma allo stesso tempo potrebbe rappresentare soggetti senza principi e calcolatori. Le informazioni ottenute sono tratti caratteriali e come tali vanno trattati, nonostante l'attendibilità del test non possono essere considerati fattori come lo sforzo a migliorare, la volontà di sottoporsi a contesti non in linea con la propria personalità e comfort zone, per tale motivo questa analisi si affianca come ulteriore strumento, ma in nessun modo preclude un'analisi personale successiva.

Capitolo 5

Presentazione della soluzione

Di conseguenza agli obiettivi descritti precedentemente, ho optato per realizzare una soluzione semplificata mediante delle assunzioni in modo da poterla testare nonostante il ridotto numero di campioni nel dataset di addestramento, ed in seguito creare le classi e le funzioni per adattare il software alla versione finale.

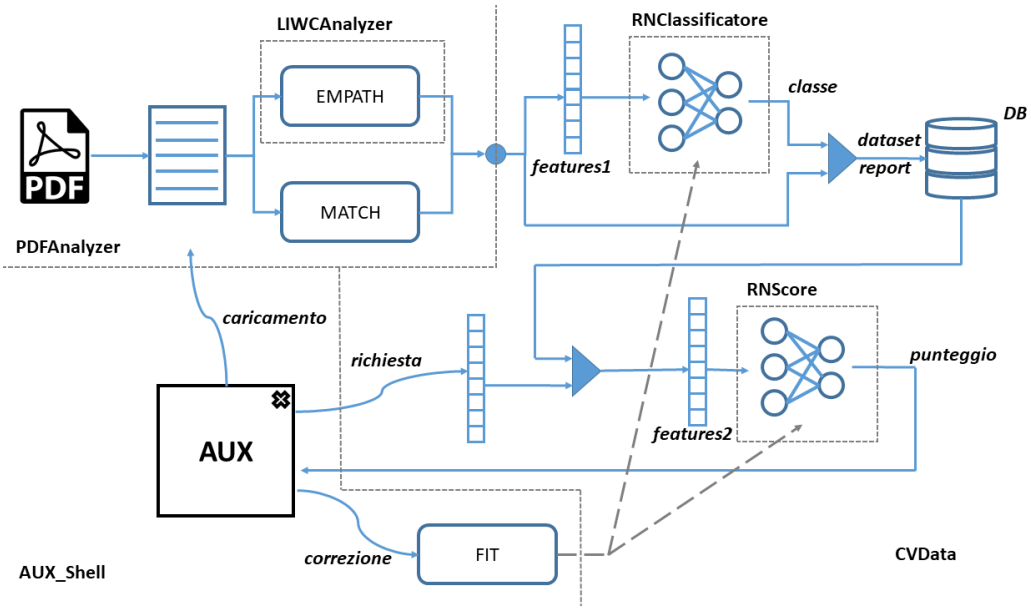

Figura 5.1: Struttura generica dell'algoritmo

La struttura generica dell'algoritmo si basa su quattro componenti fondamentali:

1. `PDFAnalyzer` si occupa dell'estrazione del testo dal PDF, della sua formattazione e infine esegue l'analisi vera e propria dei dati, fornendo in uscita una struttura rappresentativa del candidato; mediante la classe `LIWCAnalyzer` esegue l'analisi empatica basata sul metodo dei big five.

2. Sono presenti due reti neurali che operano in modo autonomo su richiesta di `CVData`, la prima `RNClassificatore` sfrutta unicamente i dati ricavati dal PDF per assegnare il candidato ad una particolare categoria tra Amministrativi, sviluppatori Front-end, Back-end, Full-stack, Grafici e Scartati; la seconda rete `RNScore` effettua una fusione dei dati del candidato e della richiesta da parte dell'HR per assegnare un punteggio di attinenza usato per l'ordinamento dei risultati.

3. L'interfaccia fornisce accesso al database, raccoglie i dati delle richieste e delle correzioni e coordina le operazioni di `PDFAnalyzer`.

4. La funzione di `FIT` permette di simulare con una rete di apprendimento supervisionato il funzionamento di un algoritmo logicamente più simile al reinforcement learning dove però la realizzazione dell'ambiente sarebbe risultata eccessivamente complessa.

L'implementazione di queste componenti varia da una versione all'altra con lo scopo di servire al meglio reti neurali dalla struttura e complessità profondamente diverse, tuttavia le funzionalità sono le stesse al fine di mantenere una compatibilità.

5.1 Mining del testo

La fonte di informazione per il programma sono i curriculum vitae, inviati solitamente in formato PDF, sebbene questo in prima analisi possa sembrare un vantaggio data la portabilità, il formato unico e il testo visivamente ben strutturato, in realtà non lo è poiché non porta con sé sufficienti informazioni riguardo la sua origine e il suo assemblaggio.
Un file PDF non contiene nessun'informazione riguardo il riassemblamento di paragrafi, frasi e parole, se proviene da un testo ci si preoccupa solo delle lettere e del loro posizionamento, tuttavia i layout sono molto vari e possono contenere titoli, icone, immagini, spazzi, tabelle, soprattutto nei cv, dove si tende a ricercare un aspetto inusuale per apparire più facilmente nel numero.

Tutto ciò rende estremamente difficile estrarre dati "puliti" da dei file PDF; senza scendere troppo nel dettaglio, è interessante soffermarsi sui test di confronto degli algoritmi e moduli effettuato da me su un discreto numero di file campione, per trovare quello che forniva le migliori prestazioni nel caso di interesse, tra i principali: `Slate`, `PyPDF2`, `Textract`, `Pdfminer`, `Pdfminer.six`, `Pdf2text`, `PDF-Layout-Scanner`, `Plumber`, `Parsel`, `Tabula-py`,

`Pdftotext` (command line utility), questi con opportune modifiche, impostazioni e varianti differenti e con la presenza o meno di algoritmi di tokenization quali `Beautifulsoup`, `Spacy` e `Nltk`.

La conclusione è che nessun algoritmo presente al momento è in grado di estrapolare la stringa di testo in modo impeccabile, si deve scendere ad un compromesso tra accuratezza delle parole e ricostruzione del contesto.
Non solo frasi, è persino difficile ottenere parole singole opportunamente separate, posizionate e con i caratteri corretti, ad esempio `pdf2text` non lavora correttamente con pagine multi-colonna, `Slate` o `PDF-layout-Scanner` compongono meglio le frasi ma commettono un numero molto alto di errori nel riconoscimento dei caratteri.
Gli errori presenti nel testo ricavato compromettono la divisione in token che in molti casi raggruppa parole sintatticamente separate, mentre in altri separa link o termini che sarebbero stati facilmente identificabili con un'analisi ad espressioni regolari, la tokenization non portava nessun miglioramento effettivo e aumentava la complessità, così è stata abbandonata nella versione semplificata.

Le due versioni presentano una fondamentale differenza logica e di conseguenza esigenze molto differenti, infatti la versione semplificata lavora con chiavi estratte tramite espressioni regolari, ne deriva che l'importanza maggiore doveva essere data alla correttezza dei caratteri e la scelta è ricaduta su `PdfMiner`; la seconda versione invece utilizza un'analisi contestuale, per questo anche se una parola fosse sbagliata il significato verrebbe mantenuto, come avviene quando a leggere è una persona, quindi la scelta è stata `pdftotext`.

Questi algoritmi usano un processo di ricostruzione delle strutture basato sulla posizione dei caratteri in tre stadi: raggruppare le lettere in parole e linee, raggruppare le linee in box e infine disporre i box in un albero gerarchico.
Il raggruppamento si basa sulle coordinate cartesiane, se la distanza tra due lettere è minore dell'ingombro di un carattere e rispetta l'overlap verticale allora faranno parte della stessa parola, altrimenti viene inserito uno spazio, questo passaggio è problematico poiché il formato PDF non ha il concetto di spazio e questo causa un maggior numero di errori.
Lo stesso concetto viene applicato per gli altri raggruppamenti ed è evidente come grafiche particolari e continui cambi di font siano il punto debole di questo approccio.

5.2 V.1 Versione semplificata

L'obiettivo della versione semplificata (in seguito chiamata V1) era quello realizzare un MVP[1] per testare la struttura generica senza scontrarsi con il

[1] Minimum Valuable Product

problema del dataset troppo ridotto, consapevole di non poter raggiungere affidabilità elevate (comunque impossibili a causa dell'overfitting) mi sono concentrato sul fornire uno strumento che permettesse all'HR di catalogare i dati in modo passivo durante il suo lavoro e valutare l'utilità della soluzione.

Riguardo al database, essendo il prototipo spostato su diversi dispositivi non era conveniente integrarne uno vero e proprio, quindi per semplicità e per permettere alle reti neurali di portare con sé "esperienza" anche quando il package viene spostato, tutti i dati sono salvati in formato TXT e CSV in locale, in un albero di directory interne al package stesso.

Il flusso di lavoro si articola come segue:

1. I nuovi cv vengono inseriti nella cartella ./cv/new/ e da comando tramite l'interfaccia utente essi vengono analizzati, classificati, spostati nelle cartelle dedicate e per ognuno viene salvato in modo permanente un oggetto `CVData`.

2. Quando un HR vuole eseguire una ricerca dall'interfaccia utente indica il ruolo e le caratteristiche richieste, sulla base di queste informazioni vengono estratti i candidati più adatti e tramite un merge dei dati `CVData` e quelli della richiesta, per ognuno viene calcolato un punteggio che influenza l'ordine di presentazione sull'interfaccia.

3. L'HR può visualizzare in modo intuitivo le informazioni chiave, aprire il curriculum e visualizzare i contatti dell'utente.

4. Se le predizioni effettuate dalle reti neurali non fossero corrette o adatte alle preferenze soggettive è possibile effettuare delle modifiche manuali, queste vanno ad aggiornare direttamente i dataset e ad addestrare le reti, in questo modo si ottengono dati classificati e algoritmi accurati in modo contemporaneo al lavoro ordinario dell'impiegato.

5.2.1 Analisi LIWC

Come spiegato nella sezione 4.2 dai testi scritti dal candidato è possibile ricavare informazioni legate alla sua personalità, nel programma non vi è un'analisi contestuale, ma si opta per uno studio sulle singole parole effettuato da istanze della classe `LIWCAnalyzer`, realizzata come wrapper per lo scopo.
Tra le varie librerie presenti, la scelta è ricaduta su `Empath` opportunamente modificata e utilizzata in modalità normalize (analisi delle parole prese singolarmente), il database originale, in lingua inglese da oltre 180 000 parole diviso in 190 categorie, è stato integrato con la traduzione in italiano e adattato al contesto.
Al fine di rendere più comprensibili i dati si è deciso di fare un ulteriore raggruppamento delle categorie nelle dimensioni del modello dei big five, poiché più intuitivo e accessibile per un'analisi che non richiede estrema profondità.

5.2.2 Analisi Match

Questo tipo di analisi è concettualmente molto semplice, ma si è dimostrato estremamente affidabile e preciso nell'estrazione di informazioni, questo è dovuto al contesto in cui i dati sono inseriti, infatti trattandosi di curriculum è ragionevole ipotizzare che:
- Viene riportata una competenza solo se si vuole mostrare di possederla
- Tutte le frasi sono affermative, infatti nessuno si candiderebbe per un lavoro con frasi del tipo "non so programmare in java, javascript e sql", semplicemente non verrebbero riportati

Di conseguenza lo stesso contesto che rappresenta fonte di incertezza per l'algoritmo LIWC è ottimale per una ricerca basata su match con espressioni regolari di parole chiave, le chiavi di ricerca opportunamente scritte non risentono degli errori presenti nel testo estratto dal PDF e i test hanno riportato affidabilità superiore al 90% nell'estrazione di 200 caratteristiche chiave.

5.2.3 CVData

Si tratta della classe che rappresenta un curriculum e di conseguenza un candidato, oltre a potersi salvare e ricaricare in modo permanente da una sessione all'altra, elabora i vettori di features e gestisce due reti neurali per fornire servizi all'interfaccia utente.

5.2.4 RNClassificatore

E' una classe wrapper per una rete neurale basata su regressione logistica a 6 classi esclusive, ovvero che riceve in ingresso un'array di features generato dalla funzione `exportArrayParametersClass()` di `CVData` e fornisce in uscita il ruolo unico tra Amministrativi, Front-end, Back-end, Full-Stack, Grafici e Scartati alla quale a livello probabilistico il candidato è più adatto. Essendo il dataset di cv poco profondo, il rischio di non convergenza e overfitting al set di addestramento era molto alto, per questo la strategia scelta è stata ridurre la dimensionalità delle features di ingresso, raggruppandole senza perdere troppa informazione, in futuro rimane possibile espandere la rete modificandone solo la struttura.
Le features scelte sono in Tabella 5.1.

La rete neurale non profonda ha una struttura di tipo `Sequential` composta da un layer denso di ingresso con funzione di attivazione `Relu` e uno di uscita a sei classi che sfrutta `Softmax`, per un totale di 276 parametri da addestrare, numero più che accessibile con dataset ridotti.

Features	Tipo
ID	
Sviluppatore	Bool
Back-end	Bool
Front-end	Bool
Full-stack	Bool
SEO/Pubblicità	Bool
Cyber security	Bool
Stampa 3D	Bool
Artificial intelligence	Bool
Data science/Data mining	Bool
Designer/Grafica	Bool
Video/Audio	Bool
Mobile	Bool
Web/Landing page	Bool
Comunicazione digitale	Bool
Human resources	Bool
Management	Bool
Project management/Responsabilità	Bool
Score attività extrascolastiche	Int [0-1]
Score esperienze pregresse	Int [0-1]
Titolo di studio	Ordinale [nessuna, diploma, laurea, magistrale]

Tabella 5.1: Features nell'array di input della rete naurale RNClassificatore

5.2.5 RNScore

Come la precedente si tratta di una classe wrapper che fornisce funzionalità analoghe, ma basandosi su un neurone (non si tratta di una rete avendo solo un nodo) a regressione lineare; la scelta di utilizzare un output lineare e discretizzarlo in seguito per somigliare di più ad un punteggio è dovuta sempre alla dimensione del dataset, infatti addestrare una rete logistica a 20 classi (voti da 1 a 10 con step di 0.5) avrebbe richiesto un numero molto alto di pesi mentre la soluzione adottata è risultata più semplice ed affidabile.
Le features scelte sono in Tabella 5.2.

5.2.6 Gestione dei dati

Per evitare di dover riaddestrare le reti, entrambe dispongono di funzioni di salvataggio e caricamento in opportuni file H5, i dataset invece sono memorizzati in file CSV per essere gestiti al meglio tramite la libreria `Pandas`, in ottica di un prodotto finale i dati sono stati organizzati in forma strutturata,

Features	Tipo
ID	
Conoscenza dell'italiano	Bool
Score lingue	Int [0-1]
Titolo di studio	Ordinale nessuna, diploma, laurea, magistrale
Score match linguaggi programmazione	Bool
Score match framework	Bool
Score match software	Bool
Score attività extrascolastiche	Int [0-1]
Score esperienze pregresse	Int [0-1]

Tabella 5.2: Features nell'array di input della rete naurale RNScore

adatta a database relazionali nel caso del prototipo o non relazionali per la versione 2.

5.3 V.2 Versione futura

L'obiettivo di questa seconda fase era principalmente di ricerca, non essendo possibile realizzare un dataset delle dimensioni di $10^3 \div 10^5$ esempi in breve tempo, non era possibile addestrare una rete di tale complessità; tuttavia la classe è organizzata per essere logicamente compatibile con la struttura base.
Rispetto al prototipo si è scelto un'approccio totalmente differente poiché privo di analisi preliminare: il testo ottenuto dal PDF viene codificato e fornito alla rete neurale che in questo modo sfrutta tutte le informazioni per fornire i risultati.

Il preprocessing consiste nell'utilizzo della libreria `Spacy` per effettuare una divisione in *token*[2], in seguito ogni parola viene sostituita dal suo lemma[3] con lo scopo di ridurre il numero di termini distinti, infine utilizzando un vocabolario di 400 000 parole estratto dal database GloVe ogni token viene sostituito da un codice intero univoco e posto in un'array di lunghezza fissa[4].
La rete neurale ha una struttura complessa basata su più layer ricorrenti nascosti:

1. `Embedding` è il layer di ingresso, il quale riceve un'array di interi dove ognuno codifica una parola del testo ricavato dal PDF (la lunghezza dell'array è di conseguenza pari alla lunghezza massima di testo presente in un cv, ipotizzata di 10 000 parole) ed il numero di codici ricavato

[2] Corrispondenti a parole nel senso sintattico del termine, ad esempio Los Angeles viene considerato un unico token
[3] Termine privo di declinazione, es. Pensante - Pensare
[4] Viene aggiunto un padding di codici dal significato nullo

dal database `GloVe`[5].
Lo scopo del layer di Embedding è imparare ad assegnare ad ogni parola (codice intero) una sequenza di valori di lunghezza fissa (nel nostro caso 50) che la codifichi mediante distanza del coseno rappresentandola come punto in uno spazio a 50 dimensioni; come già detto non è possibile identificare il ragionamento a valle, ma le parole contestualmente simili dovranno avere codifiche simili, ad esempio re e sovrano.
Lo strato di Embedding è concepibile come una rete a sè stante piuttosto complessa da addestrare, per questo motivo risulta utile affidarsi a database con i parametri già addestrati come GloVe.

2. Un testo è per sua natura un dato *sequenziale*, il significato di una parola cambia molto a seconda del contesto, ovvero delle parole che la precedono e seguono; dato che uno strato denso perde la "memoria" ad ogni nuovo ingresso è necessario ricorrere a due strati `LSTM`[6], composti da nodi che hanno archi pesati non solo verso gli strati precedenti, ma anche verso sè stessi nelle iterazioni successive.

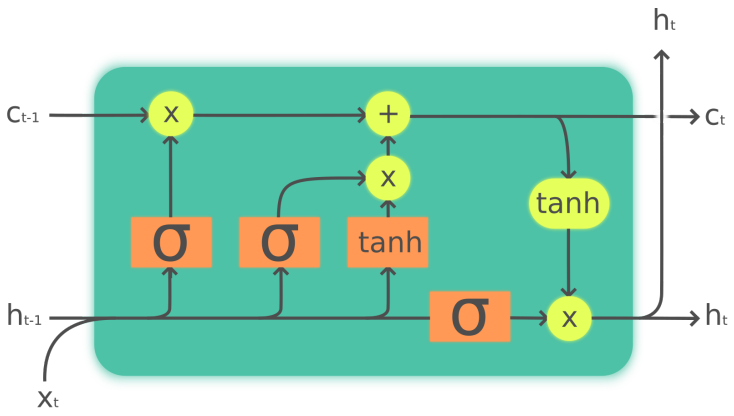

Figura 5.2: Nodo di un layer LSTM

In relazione alla figura 5.2 il canale di comunicazione che collega lo stato precedente C_{t-1} all'attuale C_t è detto *memory cell*, le sue funzionalità vengono controllate da layer interni con funzione di attivazione Sigmoid

[5]Database che contiene i parametri di Embedding addestrato sulle 400000 parole differenti presenti su Wikipedia
[6]Long Short-Term Memory

detti Gate[7] σ, in questo modo l'uscita della cella risulta essere

$$C_t = f * C_{t-1} + i * \tilde{C}_t$$

$$y_t = \Phi(o * \tanh(C_t))$$

dove f è calcolata dal Pass Gate, i dall'Input Gate, o dall'Output Gate e Φ è la funzione di attivazione dell'uscita.

3. Per concludere la rete sono stati inseriti due strati densi che riducono progressivamente il numero di nodi fino ad arrivare al numero di classi desiderato, in questo caso la complessità della rete permette di considerare anche i punteggi come un'uscita discreta di 20 classi.

Il risultato di questa struttura è una rete in grado di seguire logiche tutt'altro che banali, ma la complessità aumenta di conseguenza e il numero di parametri da addestrare è di 20 041 646, 5 ordini di grandezza superiore rispetto alla versione semplificata.
L'addestramento di tale rete, oltre a richiedere potenza di calcolo, potrebbe porre dei problemi legati all'underfitting e overfitting sul set di addestramento, oltre che alla scomparsa del gradiente, motivo per cui sono stati inseriti degli iperparametri di dropout[8].

[7] In figura 5.2 da sinistra: Forget Gate: decide se la memoria va modificata, Input Gate: calcola il nuovo stato tramite regressione lineare, Output Gate: sceglie la parte di memoria da utilizzare per l'output

[8] Eliminazione di una percentuale casuale dei nodi da un iterazione alla seguente per costringere gli altri a compensare ed evitare l'eccessivo adattamento al set di addestramento

Capitolo 6

Conclusioni

6.1 Evoluzione futura del progetto

Il progetto ha dimostrato di avere ottime potenzialità, l'analisi delle soluzioni già presenti sul mercato ha mostrato un generale interesse verso soluzioni di questo tipo, che benché complesse risultano un valido supporto agli uffici risorse umane.

La versione semplificata è riuscita ad ottenere affidabilità molto alte, nonostante la sua attendibilità sia compromessa dall'eccessivo adattamento al ridotto set di addestramento, inoltre l'approccio basato sulla funzione `FIT` si è dimostrato valido non solo per aggiornare le reti in contemporanea all'uso quotidiano senza disporre di tutti i dati all'inizio, ma anche per adattare le previsioni in base alle personali propensioni e soggettive valutazioni dell'utilizzatore.
La prima versione, per quanto ridotta, risulta essere comunque un valido supporto in grado di dimostrare le potenzialità del prodotto, è necessario tuttavia precisare che l'utilizzo di relativamente pochi dati preprocessati porterà ad avere precisioni molto alte tra ingresso ed uscita della rete, ma questo non combacia necessariamente con le conclusioni raggiunte dall'utilizzatore umano.
Infatti, quest'ultimo ha una visione d'insieme più ampia su informazioni a cui la rete non ha accesso, per questo motivo la versione finale dovrà basarsi sull'approccio della seconda versione, ovvero un'analisi libera di tutto il documento.

Per la distribuzione del servizio ci sono due approcci che differiscono non solo a livello tecnico:

1. Rete neurale centralizzata: l'algoritmo di intelligenza artificiale viene posto su un server comune ed utilizzato con un approccio REST, in questo modo si compensano le valutazioni soggettive del singolo utilizzatore e le previsioni tendono ad allinearsi con la politica aziendale, inoltre maggiori utilizzatori significa un numero maggiore di informa-

zioni per addestrare lo stesso algoritmo e solo il server dovrà disporre della potenza di calcolo necessaria.

2. Rete neurale distribuita: l'algoritmo necessariamente pre-addestrato viene duplicato su ogni macchina, in questo modo le previsioni si adattano alle preferenze personali venendo percepite come più precise, tuttavia all'aumentare della complessità della rete i singoli host dovranno essere occupati per ore ad ogni riaddestramento, motivo per cui un server rimarrebbe comunque necessario.

Una possibile strada da seguire potrebbe essere quello dell'uso interno della versione semplificata per raccogliere curriculum catalogati indicativamente in numero ≥ 5000[1]; la struttura del database mantiene mediante una mappa il collegamento codice $\leftrightarrow$ nome_file che tramite Pandas (o SQL se riportato su database relazionale) permetterà di effettuare un join tra gli input calcolati dalla versione due e le label assegnate nell'uso della uno.

Per quanto riguarda invece gli aspetti tecnici della rete più evoluta, data la dimensione degli input (array di codici interi di lunghezza dell'ordine di 10^5) e il loro numero, sicuramente sarà necessario agire sul iperparametro batch per evitare il caricamento in RAM di tutto il dataset, inoltre a seconda della quantità e qualità dei dati si presenteranno problemi di underfitting o overfitting.
Altro problema possibile sarà la scomparsa del gradiente, ovvero quando nella regressione lineare il gradiente dell'errore quadratico medio diventa prossimo a zero impedendo un aggiornamento per le epoche successive, per limitare questo problema la rete dispone già di strati di dropout e implementa un ottimizzatore *Adam* (unione del Nesterov Momentum e del learning rate adattivo), tuttavia soltanto eseguendo dei test sarà possibile regolare adeguatamente gli iperparametri e compensare il problema.

Speculando sulle possibili funzionalità future alcune proposte interessanti potrebbero essere: sfruttare i dati contenuti nel cv per analizzare anche i profili social o i siti personali mediante web scrapping e dare accesso diretto a database comuni o realizzarne in collaborazione alle varie organizzazioni, questo è consentito dal GDPR[2] poichè le reti trattano in modo completamente anonimo i dati da cui apprendono e non ricavano un risultato in relazione diretta all'identità della persona; oppure aumentare ulteriormente la complessità della rete per implementare strutture di *Attention* e *BidirectionalGRU* per creare un'analisi intrecciata con fonti diverse dai curriculum, ad esempio la vision aziendale.

[1] Solitamente immaginando il dataset di addestramento come una tabella con gli esempi disposti sulle righe e i relativi dati (features) sulle colonne, si tende a ricercare n°righe>n°colonne, di conseguenza il consiglio è avere più cv catalogati rispetto al numero massimo di parole negli stessi

[2] General Data Protection Regulation, regolamento europeo per la protezione dei dati sensibili

6.2 Considerazioni riguardo il tirocinio

La mia esperienza di tirocinio è stata al passo con le aspettative, nonostante gli imprevisti causati dall'emergenza Covid non mi abbiano permesso di trascorrere molto tempo in ufficio, l'azienda si è mossa prontamente per mettermi nella condizione di continuare il lavoro da casa.

Nonostante volesse dire che non vi fossero figure che potessero insegnarmi direttamente aspetti teorici e pratici, ho trovato entusiasmante avere la possibilità di curare in modo autonomo un progetto aziendale nelle sue prime fasi, inoltre potendomi gestire autonomamente ho potuto organizzare il tempo in modo da terminare il software, ma allo stesso tempo ricavare il massimo della formazione riguardo a tematiche che in università comprensibilmente non vengono trattate.

Durante il periodo di tirocinio ho seguito diversi corsi online: *Deep Learning e reti Neurali*, *Natural Language Processing* e *Computer Vision* forniti da Profession.IA e *Machine Learning* della Standford University, alcuni di essi approfondivano aspetti più teorici e matematici mentre altri avevano un'approccio pratico all'uso di Keras.

Il contesto mi ha permesso di acquisire anche diverse competenze riguardo l'uso di specifici software quali IDE come Atom con Hidrogen, Visual Studio Code, Jupiter, Anaconda, Google Collaboratory, software per la gestione di repository e segnalazioni come GitLab e RedMine oppure di approfondire l'uso framework specifici del linguaggio Python tra i quali Pandas, Numpy, Scikitlearn, Scipy, Spacy, Re, Keras ec.
L'ultima settimana di lavoro dopo la consegna, mi ha concesso il tempo per concludere la mia formazione includendo anche delle librerie di Computer Vision come OpenCV e di imparare un linguaggio di markup, il Latex, approfittando della scrittura del presente elaborato per fare pratica.

In conclusione, ritengo che l'attività di tirocinio sia stata gratificante ed utile per approcciarmi al mondo del lavoro in un contesto diverso dai miei luoghi di origine, ma soprattutto per definire meglio il percorso di studio che voglio intraprendere a seguito della laurea triennale; le competenze apprese mi saranno utili anche per lo sviluppo di miei progetti personali.

Bibliografia

[1] Logg Jennifer, Minson Julia, Moore Don *Algorithm Appreciation: People Prefer Algorithmic To Human Judgment* Harvard Business School w.p.17-086 2018

[2] https://it.mashable.com/intelligenza-artificiale/697/intelligenza-artificiale-e-discriminazione-quei-bug-ereditati-dalluomo

[3] Hofstadler Douglas *Gödel, Escher, Bach: an Ethernal Golden Braid* 1979

[4] https://www.engitel.com/news/2014/la-gestione-dell-ia-i-principi-di-asilomar.html

[5] https://www.innovationpost.it/2018/02/14/intelligenza-artificiale-deep-learning-e-machine-learning-quali-sono-le-differen

[6] https://it.wikipedia.org/wiki/Intelligenza_artificiale

[7] https://it.wikipedia.org/wiki/Big_Five_(psicologia)

[8] https://it.qwe.wiki/wiki/Big_Five_personality_traits

[9] https://keras.io/

[10] https://pdfminersix.readthedocs.io/en/latest/topics/converting_pdf_to_text.html

[11] Adobe System Inc. (2007). Pdf reference: Adobe portable document format, version 1.7

[12] https://en.wikipedia.org/wiki/Long_short-term_memory

[13] Hochreiter Sepp, Schmidhuber Jurgen *Long short-term memory* 1997

[14] Olah Christopher *Unterstanding LSTM Networks 2015*

[15] Goodfellow Ian, Bengio Yoshua, Courville Aaron *The Deep Learnign Book* 2015

[16] https://www.garanteprivacy.it/il-testo-del-regolamento