Alma Mater Studiorum · Università di Bologna

Campus di Cesena
Scuola di Ingegneria e Architettura
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA E SCIENZE INFORMATICHE

Mining LinkedIn Social Media: Tecniche di Data e Text mining applicate a Recommender System per la Ricerca ed Offerta di Lavoro

Tesi in: Data Mining

Relatore:
Prof. GIANLUCA MORO
Correlatori:
Dott. Ing. ROBERTO PASOLINI
Dott. Ing. GIACOMO DOMENICONI

Presentata da:
KARIN PASINI

ANNO ACCADEMICO 2013–2014
III SESSIONE

Indice

Introduzione

La nascita dei social media e più in generale la digitalizzazione della società hanno dato origine ad enormi masse di dati in formato elettronico. Da qui l'idea di analizzare ed elaborare questi dati con lo scopo di creare informazione. Data e Text Mining sono due discipline che hanno proprio questo obiettivo e la ricerca sta mostrando sempre più interesse per esse perché hanno il pregio di avere tantissimi campi di applicazione e come detto prima le "materie prime" abbondano. Il periodo in cui viviamo è caratterizzato dalla crisi economica dove il tema del lavoro ricopre un ruolo fondamentale. L'idea generale di questo elaborato è quella di sfruttare le capacità del Data e Text Mining per realizzare degli strumenti in grado di rendere più facile la ricerca e l'offerta di lavoro.

La ricerca si è già occupata di questo problema affrontandone diversi aspetti. L'applicazione del Data Mining per il recruitment ha prodotto numerosi paper, alcuni esempi trattano la selezione di personale nel settore dell'high-technology [11], la predizione delle possibilità di assunzione per studenti [16] oppure lo sviluppo di una strategia per l'assunzione di conducenti di camion in aziende logistiche [37]. Questi elaborati propongono soluzioni per settori circoscritti, ma non offrono strumenti generici per il supporto alla ricerca ed offerta di lavoro.

Il filo conduttore di tutta la tesi è quello di impiegare il Data e Text Mining per sviluppare un recommender system in grado di agevolare chi è alla ricerca di lavoro e chi lo offre. Il sistema dovrà essere generale, applicabile in tutti i settori lavorativi e utilizzabile da entrambe le parti, ovvero sia dalle persone che cercano lavoro sia dalle aziende che ricercano una particolare figura. La tesi è strutturata in 7 capitoli:

- **Capitolo 1 - Stato dell'Arte**. Verrà fatta una panoramica sul tema dell'estrazione di conoscenza da grandi masse di dati, il Data Mining; successivamente si scenderà nel dettaglio del mining di dati social e del mining applicato

al recruitment; infine verranno presentati i recommender system con particolare attenzione ai recommender system che sfruttano dati provenienti dai social media.

- **Capitolo 2 - Raccolta dei dati**. In questo capitolo si realizzerà la prima fase di un processo di Data Mining: il reperimento di dati, ottenuto tramite un crawler Java che ottiene i profili pubblici degli utenti italiani del social media LinkedIn.

- **Capitolo 3 - Preprocessing e organizzazione dei dati**. Dopo aver ottenuto i dati sotto forma di pagine HTML è necessaria una fase di preprocessing in cui vengono estratte le informazioni salienti. In prima battuta verranno memorizzati i dati secondo un modello ad oggetti, per poi passare alla persistenza in un database.

- **Capitolo 4 - Analisi preliminare dei dati**. Dopo aver trasformato i dati in un formato fruibile verranno realizzate delle analisi per evincere la distribuzione dei dati e le loro caratteristiche salienti.

- **Capitolo 5 - Skill recommendation**. In questo capitolo verrà realizzato il primo recommender system che ha l'obiettivo di suggerire agli utenti LinkedIn in fase di registrazione gli skill da aggiungere al profilo, basandosi su informazioni dell'utente.

- **Capitolo 6 - Job recommendation**. In questo capitolo verranno proposti due recommender system che a partire da un insieme di skill suggeriscono le posizioni lavorative (position) più adatte a quel particolare set di skill. Il secondo recommender system sfrutta un clustering gerarchico di position realizzato nella prima parte del capitolo.

- **Capitolo 7 - Architettura e implementazione**. Nell'ultimo capitolo verranno illustrati i dettagli implementativi di tutte le elaborazioni svolte nell'arco della tesi.

Capitolo 1

Stato dell'Arte

Al giorno d'oggi, grazie alla digitalizzazione della società in tutti i suoi aspetti, per esempio nel mondo del lavoro, nell'intrattenimento, nei rapporti interpersonali ecc., si è venuta a creare una enorme massa di dati in formato elettronico, provenienti dalle fonti più disparate. Questa mole di dati è una fonte preziosa di informazioni e negli ultimi anni l'importanza di questo aspetto è aumentata notevolmente, perciò sono state studiate e realizzate tecniche per estrarre informazioni dai dati. Questa disciplina prende il nome di Data Mining.
La ricerca sul Data Mining ha prodotto numerosi metodi, tool e algoritmi per manipolare grandi quantitativi di dati con l'obiettivo di risolvere problemi concreti. I social media sono largamente utilizzati in vari ambiti, e offrono vaste quantità di dati user-generated che possono essere utilizzate nel Data Mining. Il Data Mining sui social media può ampliare la capacità dei ricercatori di comprendere nuovi fenomeni causati dall'utilizzo dei social media e migliorare la business intelligence per fornire servizi migliori e sviluppare opportunità innovative [22].
Le applicazioni del mining sui social media sono innumerevoli e possono riguardare i temi più diversi: aumentare i profitti provenienti dalle vendite massimizzando il passaparola positivo tra consumatori [13], monitorare la diffusione delle epidemie di influenza [12], predire i ricavi al botteghino di film analizzando i tweet [4] ecc.
Come detto in precedenza, le tecniche di Data Mining sono utili per affrontare diverse tipologie di problemi. Un tema di grande interesse oggi è quello del recruitment, ovvero trovare lavoro per chi lo cerca e selezionare la giusta figura per le aziende. Il Data Mining può essere utilizzato per supportare il recruitment in diverse maniere, per esempio: determinare il profilo dei conducenti di camion per un'assunzione di successo [37], predire se la carriera intrapresa avrà successo oppure no [16], svilup-

pare un framework basato su decision tree e association rule in grado di generare regole utili per la selezione del personale [11].

1.1 Mining di grandi quantità di dati

La presenza di grandi quantità di dati, disponibili in diverse forme (documenti, formato grafico, video, registrazioni audio), ha fatto nascere l'interesse dell'Information Technology sulla scoperta di informazioni utili da grandi collezioni di dati. La crescente attenzione per questo campo deriva dall'intuizione che *siamo ricchi di dati ma poveri di informazioni*. La creazione di informazione necessita di masse enormi di dati ed è un processo non elementare composto non soltanto dal data retrieval. Per trarre il massimo beneficio sono necessari anche tool per la data summarization, tecniche per l'estrazione dell'essenza delle informazioni memorizzate e per la scoperta di pattern nei dati non elaborati. A causa dell'enorme ammontare di dati è sempre più importante sviluppare tool per l'analisi e l'interpretazione di essi e per l'estrazione di conoscenza utile nel processo decisionale. Il riassunto di tutto questo è chiamato "Data Mining".
Data Mining è l'estrazione di informazioni predittive nascoste da grandi database; è una potente tecnologia con grande potenziale per aiutare le organizzazioni a evidenziare le informazioni più importanti nei loro data warehouse. I tool per il Data Mining predicono gli andamenti futuri e i comportamenti, aiutano le organizzazioni a prendere decisioni proattive guidate dalla conoscenza. Le analisi automatizzate offerte dal Data Mining si muovono aldilà dell'analisi degli eventi passati fornita dai tool tipici dei sistemi di supporto alle decisioni. I tool per il Data Mining possono rispondere a domande che tradizionalmente sarebbero troppo time consuming da risolvere. Essi scovano pattern nascosti e potenzialmente utili dai database , estraggono informazioni predittive che gli esperti potrebbero perdere perché non rientrano nelle loro aspettative [40].
Le fasi di un processo di Data Mining comprendono: raccolta preliminare dei dati e comprensione; preparazione dei dati in cui si crea il dataset finale; analisi dei dati attraverso task di machine learning; valutazione dei risultati. I task di machine learning sono suddivisi in:

- *Classificazione.* Consiste nel formulare una funzione che mappa ogni elemento dei dati in una classe che fa parte di un insieme predefinito di classi [51]. Le tecniche di classificazione sono molteplici, le principali racchiudono gli alberi decisionali (ad es. l'algoritmo C4.5), le regole di decisione, i classificatori baye-

siani. La figura 1.1 mostra un semplice partizionamento di dati di prestiti in due classi; è da notare che non è possibile separare perfettamente le classi utilizzando un boundary lineare. Una banca utilizzerà le regioni di classificazione per decidere automaticamente se i richiedenti prestito riceveranno il prestito oppure no [18].

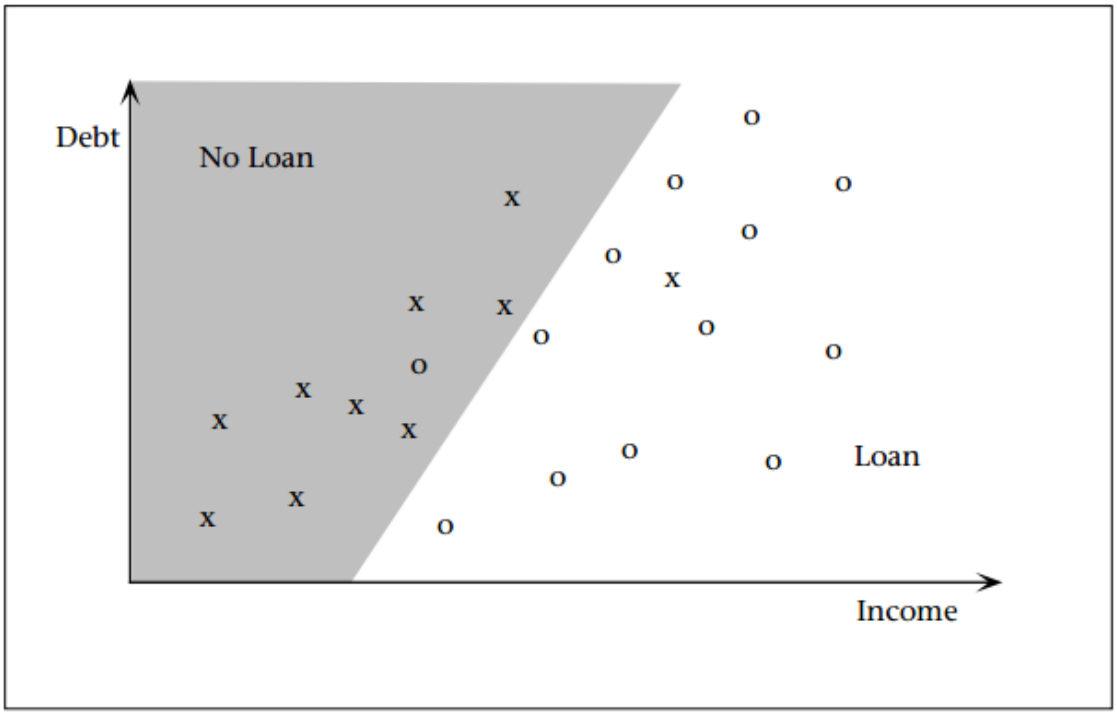

Figura 1.1: Classificazione lineare del dataset di prestiti. La regione in grigio denota chi non riceverà il prestito

- *Regressione.* Consiste nel formulare una funzione che mappa un elemento dei dati in una variabile di previsione a valori reali. Le applicazioni della regressione sono molteplici, per esempio: predire l'ammontare della biomassa presente in una foresta date misure a microonde telerilevate, stimare la probabilità che un paziente sopravviva dati i risultati di un insieme di test diagnostici, predire la richiesta dei consumatori di un nuovo prodotto come funzione delle spese pubblicitarie, predire serie storiche dove le variabili di input possono essere versioni ritardate delle variabili di predizione. La figura 1.2 mostra il risultato

di una semplice regressione lineare dove il debito totale è una funzione lineare del guadagno: il fit non è ottimale perché vi è una correlazione debole tra le due variabili [18].

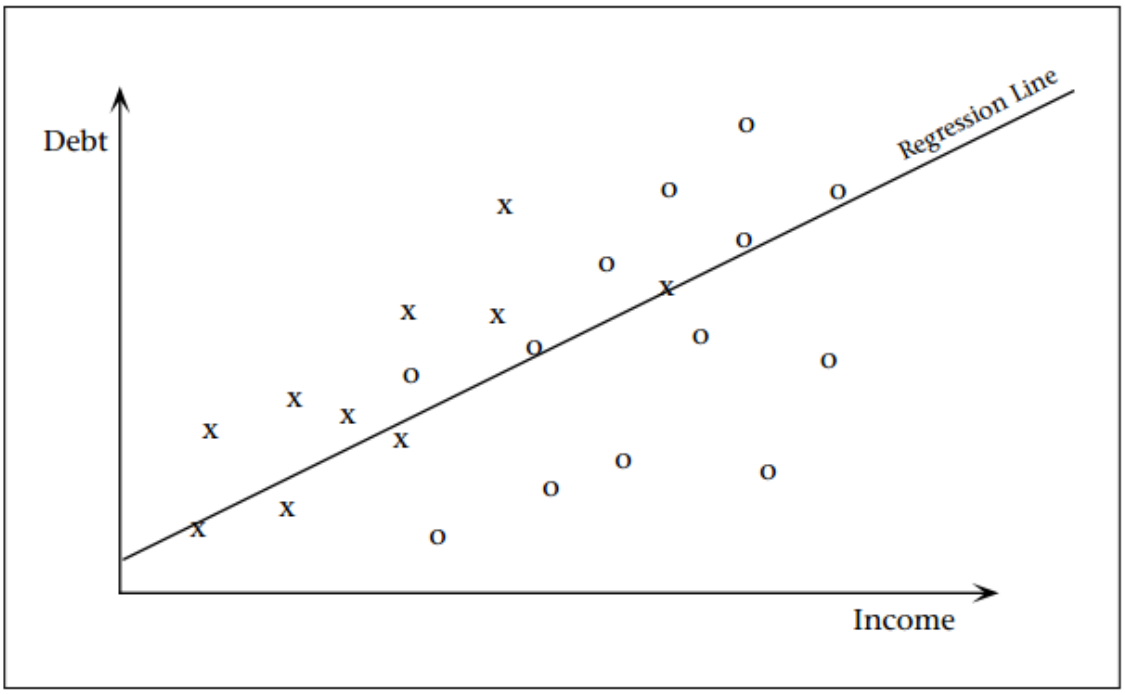

Figura 1.2: Regressione lineare del dataset di prestiti

- *Clustering*. Ha l'obiettivo di indentificare un set finito di categorie o cluster per descrivere i dati [50]. Le categorie possono essere mutualmente esclusive ed esaustive o consistere in una rappresentazione più ricca, come le categorie gerarchiche o sovrapposte. Esempi di applicazioni di clustering includono: scoperta di sottopopolazioni omogenee per consumatori nei database di vendite e scoperta di sottocategorie di spettri dalle misure ad infrarossi del cielo. La figura 1.3 mostra un possibile clustering del dataset di prestiti in tre cluster; è da notare che i cluster si sovrappongono, permettendo ai punti di appartenere a più di un cluster. Le etichette di classe originali (denotate da x e o nelle precedenti figure) sono state sostituite da + per indicare che l'appartenenza alla classe non è più assunta come conosciuta [18].

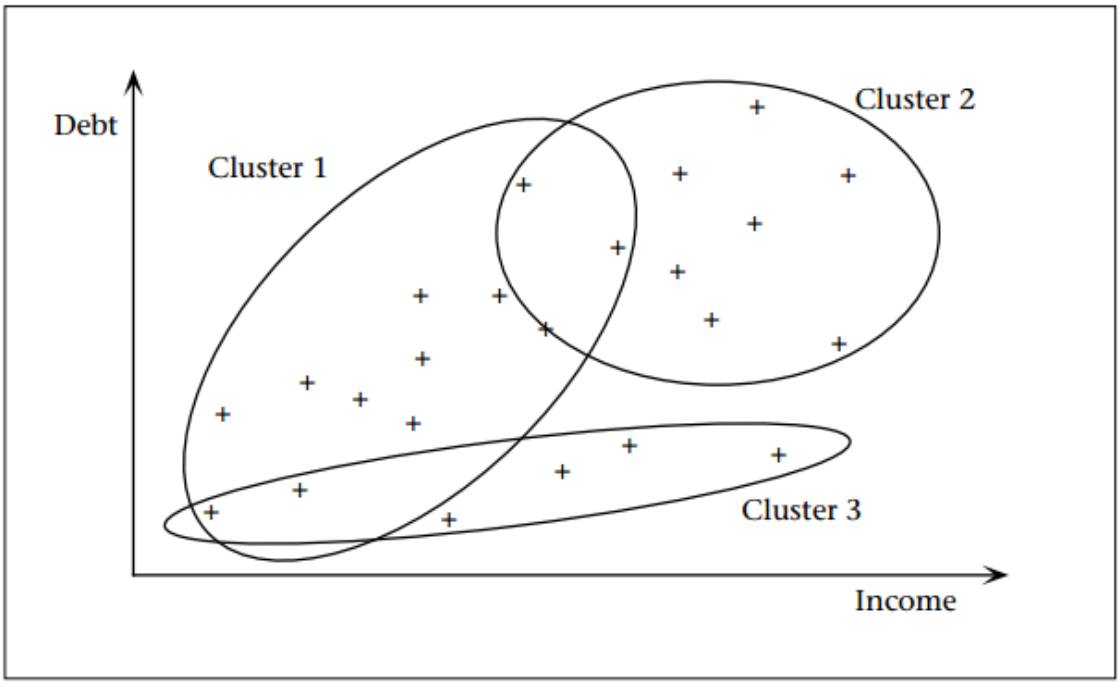

Figura 1.3: Clustering del dataset di prestiti in tre cluster

- *Regole associative.* Una regola associativa è un'espressione X ⇒ Y, dove X e Y sono insiemi di item all'interno di un database. Il significato intuitivo di una regola è che le transazioni che contengono X tendono a contenere anche Y. Un esempio di una regola potrebbe essere: "il 98% dei consumatori che acquistano pneumatici e accessori per l'auto hanno effettuato anche la manutenzione". I domini di applicazione per le regole associative spaziano dal supporto alle decisioni alla diagnosi degli allarmi di telecomunicazioni alla predizione. L'applicazione prototipale è l'analisi dei dati di vendita. Il/I codice/i a barre ha/hanno reso possibile per i venditori collezionare e immagazzinare grandi moli di dati di vendita, detti *basket data*. Un record di questi dati tipicamente consiste nella data della transazione e negli articoli acquistati nella transazione. Trovare regole associative dai basket data è utile per definire strategie di marketing [2].

1.2 Mining social media

Il Data Mining ha la caratteristica di poter essere applicato in svariati ambiti in cui siano presenti grandi quantità di dati. Con la diffusione dei social media si è venuta a creare una gigantesca massa di dati user-generated e l'interesse crescente sull'estrazione di informazione da dati grezzi ha fatto sì che questi due aspetti confluissero nel mining social media. Il mining social media è un'area multidisciplinare in rapida crescita dove ricercatori con diversi background posso apportare contributi importanti per la ricerca e lo sviluppo dei social media [22].
Social media è definito come un gruppo di applicazioni internet-based che sono costruite sui fondamenti ideologici e tecnologici del Web 2.0 e che permettono la creazione e lo scambio di contenuti user-generated [28]. Social media è un conglomerato di tipi differenti di siti di social media che includono media tradizionali come giornali, radio e televisione e media non tradizionali come Facebook, Twitter, ecc. La tabella 1.1 mostra le caratteristiche dei diversi tipi di social media.
I social media offrono all'utente un mezzo semplice per comunicare e connettersi con altri utenti su una scala senza precedenti e una frequenza mai vista nei media tradizionali. La popolarità dei social media continua a crescere esponenzialmente, che risulta in una evoluzione dei social network, blog, microblog, social network location-based, wiki (siti web che permettono di redigere in modo collaborativo i contenuti e strutture dai propri utenti), applicazioni di social bookmarking, notiziari social, condivisione media (testo, foto, audio e video), siti di recensioni ecc. Facebook, il sito di social networking, ha registrato più di 845 milioni di utenti attivi nel Dicembre 2011. Questo numero suggerisce che Cina (circa 1,3 miliardi) e India (circa 1,1 miliardi) sono i soli due paesi che hanno una popolazione più vasta di quella di Facebook. Facebook e Twitter hanno maturato più di 1,2 miliardi di utenti, più del triplo della popolazione degli Stati Uniti e più della popolazione di ogni continente eccetto l'Asia [22].
Vi sono molteplici temi di ricerca che riguardano il mining di social media, due dei più rappresentativi sono:

- **Sentiment Analysis and Opinion Mining**. Hanno lo scopo di estrarre automaticamente le opinioni espresse nei contenuti user-generated. I tool di Sentiment Analysis e Opinion Mining permettono alle aziende di capire l'opinione su un prodotto, la percezione del brand, la percezione su nuovi prodotti, e il reputation management. Questi tool aiutano gli utenti a percepire le opinioni o sentimenti sui prodotti su scala globale.
  La Sentiment Analysis è difficile da realizzare perché i linguaggi utilizzati per

creare i contenuti sono ambigui. Le fasi principali della Sentiment Analysis sono: (1) trovare i documenti rilevanti, (2) trovare le sezioni rilevanti, (3) capire il sentimento generale, (4) quantificare il sentimento, (5) aggregare i sentimenti a formare una overview [22].

- **Social Recommendation**. I sistemi di recommendation tradizionali tentano di suggerire oggetti basandosi sulle valutazioni aggregate degli utenti o sulla storia degli acquisti precedenti di un utente. Un sistema di recommendation social è basato sulle ipotesi che è più probabile che le persone che sono socialmente connesse abbiano gli stessi o simili interessi (omofilia), e che gli utenti possono essere facilmente influenzati dagli amici di cui si fidano e preferiscono le raccomandazioni dei loro amici alle raccomandazioni casuali. Gli obiettivi dei sistemi di recommendation social sono di migliorare la qualità della recommendation e alleviare il problema del sovraccarico di informazioni. Esempi di sistemi di recommendation social sono: recommendation di libri basata sulla lista di libri letti dagli amici su Amazon e recommendation degli amici su Facebook o Twitter [22].

Type	Characteristics
Online social networking	Online social networks are Web-based services that allow individuals and communities to connect with real-world friends and acquaintances online. Users interact with each other through status updates, comments, media sharing, messages, etc. (e.g., Facebook, LinkedIn)
Blogging	A blog is a journal-like website for users, aka bloggers, to contribute textual and multimedia content, arranged in reverse chronological order. Blogs are generally maintained by an individual or by a community (e.g., Huffington Post, Business Insider, Engadget)
Microblogging	Microblogs can be considered same a blogs but with limited content (e.g., Twitter, Tumblr)
Wikis	A wiki is a collaborative editing environment that allow multiple users to develop Web pages (e.g., Wikipedia, Wikitravel, Wikihow)
Social news	Social news refers to the sharing and selection of news stories and articles by community of users (e.g., Digg, Slashdot, Reddit)
Social bookmarking	Social bookmarking sites allow users to bookmark Web content for storage, organization, and sharing (e.g., Delicious, StumbleUpon)
Media sharing	Media sharing is an umbrella term that refers to the sharing of variety of media on the Web including video, audio, and photo (e.g., YouTube, Flickr, UstreamTV)
Opinion, reviews, and ratings	The primary function of such sites is to collect and publish user- submitted content in the form of subjective commentary on existing products, services, entertainment, businesses, places, etc. Some of these sites also provide products reviews (e.g., Epinions, Yelp, Cnet)
Answers	These sites provide a platform for users seeking advice, guidance, or knowledge to ask questions. Other users from the community can answer these questions based on previous experiences, personal opinions, or relevent research. Answers are generally judged using ratings and comments (e.g., Yahoo! answers, WikiAnswers).

Tabella 1.1: Caratteristiche dei tipi differenti di social media, riportata da [22]

1.3 Mining per il recruitment

Nel periodo storico in cui ci troviamo la ricerca del lavoro è diventata un problema molto sentito. Per questo sono nati svariati strumenti che aiutano le persone a trovare il lavoro più adatto ad esse ed allo stesso modo aiutano le aziende a selezionare il candidato migliore per ricoprire la figura ricercata.
La ricerca sul Data Mining si è occupata anche di questo ambito, grazie alla capacità di applicazione su moltissimi campi. In letteratura sono presenti numerosi esempi di Data Mining applicato al tema del lavoro in tutte le sue sfaccettature.

Analisi del profilo di conducenti di camion per un'assunzione di successo

In questo articolo vengono illustrate le fasi di costruzione di un decision tree che sono:

- *Raccolta dei dati.* È stato fatto un sondaggio in cui è stato inviato un questionario a diverse aziende di trasporti per cercare di esaminare le cause dell'elevato ricambio di autisti. I dati grezzi hanno poi subìto una fase di preprocessing dove sono stati eliminati i dati non corretti o inconsistenti e successivamente convertite le variabili categoriche in variabili numeriche.

- *Formattazione dei dati.* I dati sono stati formattati in modo da eseguire facilmente delle query e in modo da specificare i diversi pattern in categorie.

- *Induzione di regole.* Gli alberi decisionali generano un insieme di regole che possono essere facilmente comprese da un manager della logistica. Queste regole possono dare importanti indizi su come il conducente risponderà alle offerte dell'azienda e conseguentemente aiutare l'azienda a formulare una strategia di assunzione efficace. Tra i tanti algoritmi disponibili è stato scelto il C5.0 per la sua velocità, bassa richiesta di memoria, capacità di pruning, boosting e cross-validation.

L'albero decisionale risultante ha permesso di ricavare diverse regole che caratterizzano i conducenti con un alto turnover [37].

Analisi del profilo di studenti per valutare le possibilità di collocamento

Il lavoro presenta l'applicazione degli alberi decisionali per valutare se la carriera intrapresa da uno studente sia quella giusta.

I dati sono stati forniti da un ente che raccoglie feedback dagli studenti che hanno frequentato istituti tecnici in India.
È stato creato un tool web che accetta diversi parametri della persona, tra cui sesso, settore, ranking ecc. In base ai parametri di input, viene navigato l'albero decisionale fino ad arrivare al nodo foglia che rappresenta la classe, ovvero la possibilità di carriera (eccellente, buona, scarsa) [16].

Sviluppo di alberi decisionali e association rule per migliorare le strategie di selezione

Questo articolo illustra il processo di costruzione di un albero decisionale che ha l'obiettivo di estrarre le regole riguardanti le performance e i motivi di abbandono del lavoro da parte dei dipendenti.
I dati di partenza riguardano diverse caratteristiche della persona: età, sesso, stato civile, esperienza, grado di educazione, area di competenza, scuola in cui si è studiato e canale di assunzione. Una volta ottenute le regole sono state presentate ad un gruppo di persone esperte in risorse umane che hanno interpretato e organizzato le regole, dividendole in gruppi riguardanti: il canale di assunzione, educazione e esperienza lavorativa.
I risultati finali sono infine stati utilizzati per sviluppare una strategia di selezione ottimale [11].

1.4 Recommender System

1.4.1 Overview

Le persone spesso si trovano di fronte al problema di dover fare una scelta tra innumerevoli possibilità e senza aver una precedente esperienza. Da qui la necessità di avere degli strumenti che possano supportare questa scelta, i recommender system.
I primi studi in questo ambito apparvero negli anni '90 in cui nacque il concetto di collaborative filtering. Nel 1992 venne sviluppato Tapestry, un sistema sperimentale creato per ricevere, presentare e gestire documenti elettronici. Tapestry supporta il collaborative filtering, nel senso che gli utenti sono incoraggiati ad annotare i documenti che leggono, e queste annotazioni sono utilizzate per il filtering. Quando l'utente ricerca un documento con certe caratteristiche, Tapestry propone i documenti che fanno match con quel particolare filtro [20]. Nel 1994 venne sviluppato GroupLens, un sistema per il collaborative filtering di news online, che aiuta le per-

sone a trovare gli articoli di interesse nell'enorme massa di articoli disponibili [43]. Nel 1995 venne presentato un algoritmo social basato sulla storia di utilizzo che sfrutta le scelte già effettuate dalle persone come filtri e guide, mostrando la sua efficacia in un caso di studio specifico: la selezione di video da un grande repository [24].
La formulazione più comune dei recommender system, sviluppata negli anni '90, li definisce come problemi di recommendation che fanno affidamento esplicitamente ad una struttura di rating. Il problema è ridotto alla stima di rating (punteggi, voti) di item che non sono stati visti dall'utente. Intuitivamente, la stima è basata sui rating dati dagli utenti su altri item e una volta stimati gli item non ancora valutati, è possibile indicare all'utente gli item con il maggiore rating. Il problema di recommendation può essere formulato in maniera formale: sia C l'insieme di tutti gli utenti e S l'insieme di tutti i possibili item che possono essere suggeriti, come libri, film, ristoranti. Lo spazio S degli item possibili può essere molto grande, in un range di centinaia di migliaia o anche milioni di item in alcune applicazioni, come per i libri o CD. Similarmente, anche lo spazio degli utenti può essere molto vasto. Sia u una funzione di utilità che misura l'utility di un item s per uno user c, ovvero $C \times S \longrightarrow R$, dove R è un insieme ordinato. Per ogni utente $c \in C$, vogliamo scegliere gli item $s' \in S$ che massimizzano l'utility. Più formalmente:

$$\forall c \in C,\ s'_c = \arg\max_{s \in S} u(c, s)$$

Nei sistemi di recommendation, l'utility di un item è solitamente rappresentata da un rating, che indica quanto a un particolare utente piace un particolare item. L'utility tuttavia può essere una funzione arbitraria, inclusa una funzione di profitto.
Il problema centrale dei recommender system è che l'utility u non è definita nell'intero spazio $C \times S$, ma solo in un suo sottoinsieme. Questo significa che u deve essere estrapolato dall'intero spazio $C \times S$. I rating degli item non ancora valutati possono essere stimati in diverse maniere utilizzando metodi che si rifanno al machine learning, approximation theory e varie euristiche. I recommender system sono classificati in accordo al loro approccio di stima dei rating [1].

1. *Content-based recommendation.* I sistemi di recommendation content-based hano l'obiettivo di suggerire item simili a quelli che l'utente ha valutato positivamente nel passato. Infatti, il processo base eseguito da un sistema di recommendation content-based consiste nel far corrispondere gli attributi del profilo dell'utente che rappresentano le preferenze e gli interessi con gli attributi di un item, con il fine di indicare all'utente nuovi item interessanti [35].

2. *Collaborative recommendation.* Il collaborative filtering, a differenza della content-based recommendation, si basa sull'idea che simili utenti hanno simili preferenze. Gli item suggeriti all'utente saranno quindi gli stessi item che hanno una buona valutazione da parte di utenti simili. Gli algoritmi di collaborative filtering possono essere di due tipi: *Memory-based*, che operano sull'intero database per formulare suggerimenti; *Model-based*, che utilizzano il database per creare un modello che servirà per effettuare le predizioni [9].

3. *Approcci ibridi.* Questi sistemi combinano le caratteristiche dei recommender content-based e del collaborative filtering, con lo scopo di minimizzare i difetti presenti in ognuno dei due approcci. L'unione dei due metodi può essere realizzata in diverse maniere: realizzare separatamente i due recommender per poi combinare i risultati [42]; aggiungere caratteristiche del metodo content-based al modello collaborativo [5]; aggiungere caratteristiche collaborative al metodo content-based [48]; sviluppare un modello che coniuga caratteristiche del metodo content-based e del collaborative filtering [7].

1.4.2 Social media Recommender System

I recommender system, fin dalla loro prima apparizione, hanno ricoperto e ricoprono tutt'ora un'area di ricerca molto importante ed attiva. Il grande interesse per questi sistemi è dovuto alla loro capacità di essere applicati negli ambiti più disparati: dall'e-commerce [32], alla recommendation di film [36], alla recommendation di articoli [43]. Negli ultimi anni, con la nascita dei social media, l'attenzione si è posta sull'impiego dei dati provenienti da queste fonti per migliorare e perfezionare i recommender system. Numerosi lavori sono stati prodotti su questo argomento, di seguito alcuni esempi.

Social Networks and Social Information Filtering on Digg

L'articolo analizza il social media Digg, dove gli utenti possono sottomettere link, votare e discutere notizie. Gli utenti possono collegarsi ad altri attraverso l'"amicizia" e quindi seguire le loro attività. Viene mostrato che l'interfaccia degli amici realizza proprio un social filtering system, in cui all'utente vengono proposte le notizie che i suoi amici hanno trovato interessanti [31].

Social Media Recommendation based on People and Tags

In questo paper viene sviluppato un recommender system che colleziona e aggrega informazioni circa le relazioni tra utenti, tag e item. Le sorgenti di informazioni sono diverse e includono blog, segnalibri, community, wiki e file condivisi che fanno parte di una suite di applicazioni aziendali. Il sistema, basandosi sulle relazioni aggregate, suggerisce item legati alle persone e tag legati agli utenti. Ogni item suggerito è accompagnato dalla spiegazione che include le persone e i tag che hanno portato a questo suggerimento, così come le relazioni tra l'utente e l'item [23].

On Social Networks and Collaborative Recommendation

Il lavoro riguarda lo sviluppo di un recommender system di tracce musicali. La fonte di dati è il social network musicale last.fm, che contiene una grande quantità di dati multimediali arricchiti da espliciti commenti degli utenti e feedback impliciti che descrivono le preferenze di ogni utente. In questo tipo di sistemi è tendenza comune le creazione di reti virtuali tra gli utenti permettendogli di stabilire relazioni di amicizia. Considerando sia i commenti degli utenti che le relazioni intrinseche stabilite tra utenti, item e tag, viene creato un recommender system collaborativo che si adatta alle caratteristiche personali di ogni utente [29].

Capitolo 2

Raccolta dei dati

La prima fase di un processo di Data Mining è la raccolta dei dati. Essa può avvenire in diverse maniere: tramite sondaggi, raccolta personale, richiesta ai proprietari ecc. In questo caso l'obiettivo è ottenere dati dai social network per poi poterli analizzare ed applicare tecniche di Data Mining per estrarre conoscenza.

2.1 Social Network

La tipologia dei dati di interesse è tutto ciò che riguarda una persona con particolare attenzione all'ambito lavorativo. Inizialmente è stata effettuata una panoramica sui principali social network per individuare quale rendesse disponibili le informazioni di interesse, in particolare Google+, Facebook e LinkedIn. Data la tematica da trattare, LinkedIn è apparso come il social media più adatto su cui concentrarsi, perché specifica per ogni utente un insieme di caratteristiche ad-hoc per l'ambito lavorativo.
LinkedIn è un servizio web di social network, gratuito (con servizi opzionali a pagamento), impiegato principalmente per lo sviluppo di contatti professionali. Nel 2014 ha raggiunto i 300 milioni di utenti [38]. Gli obiettivi di LinkedIn sono: ottenere di essere presentati a qualcuno che si desidera conoscere attraverso un contatto mutuo e affidabile; trovare offerte di lavoro, persone, opportunità di business con il supporto di qualcuno presente all'interno della propria lista di contatti o del proprio network; i datori di lavoro possono pubblicare offerte e ricercare potenziali candidati; le persone in cerca di lavoro possono leggere i profili dei reclutatori e scoprire se tra i propri contatti si trovi qualcuno in grado di metterli direttamente in contatto

con loro [53].
I contenuti di LinkedIn, come di tutti i social media, sono oggetto di ricerca, infatti sono state prodotte diverse pubblicazioni e libri, come [45], [6], [41], [47].

2.2 Crawler

Tutti i social network più famosi offrono un insieme di API, *Application Programming Interface*, ovvero un insieme di funzionalità per poter comunicare con essi. Tra le varie funzionalità offerte, vi sono anche quelle per ottenere dati degli utenti, ma sfortunatamente le API di LinkedIn non permettono di memorizzare dati di utenti che non siano l'utente che ha ottenuto l'autenticazione all'utilizzo delle API.
Non potendo utilizzare le API per ottenere i dati degli utenti, un'alternativa poteva essere quella di sfruttare i profili pubblici degli utenti di LinkedIn: quando un utente effettua la registrazione, può decidere se rendere pubblico il proprio profilo a chi non è un suo contatto e addirittura a chi non ha effettuato l'accesso, specificando quali informazioni del suo profilo mostrare.
A questo punto occorre uno strumento che scarichi automaticamente più profili possibili, per esempio un *crawler*. Un crawler è un termine generico che indica qualsiasi programma utilizzato per l'individuazione e la scansione automatiche dei siti web seguendo i link da una pagina web all'altra. Partendo da una pagina web, il crawler analizza tutti i link presenti nella pagina fino ad una certa profondità impostata. LinkedIn ha una sezione in cui sono organizzati per lettera tutti gli utenti di un determinato paese e, navigando le directory, è possibile accedere ai profili pubblici degli utenti come mostrato in figura 2.1.

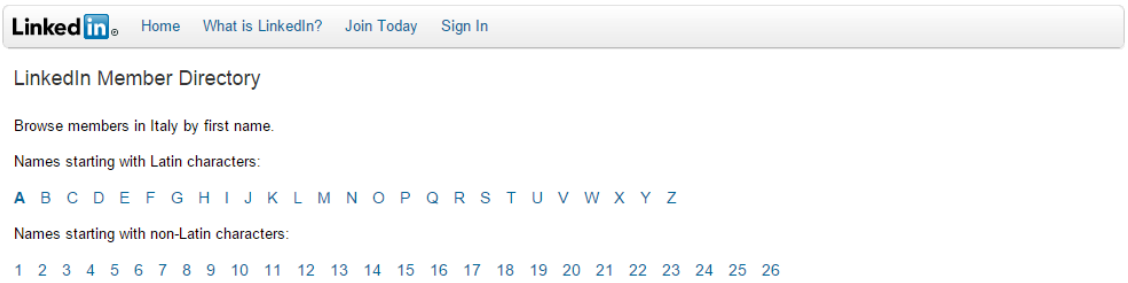

Figura 2.1: Directory di LinkedIn per navigare i profili pubblici degli utenti italiani

Utilizzando un crawler partendo dalla directory principale sarebbe in teoria possibile scaricare tutti i profili pubblici, ma essendoci 300 milioni di utenti, è necessario restringere ai soli profili italiani, che comunque hanno superato i 7 milioni nel 2014. Il crawler utilizzato è HTTrack, semplice da impostare e open-source. Facendo qualche prova con questo software, si è notato che i tempi di scaricamento dei profili erano molto lunghi in quanto il crawler navigava tutti i link della pagina, che nella maggior parte dei casi non erano di interesse e quindi diventava infattibile l'ottenimento dei profili tramite un software già esistente.
Data l'impossibilità di utilizzo di un crawler, si è pensato di realizzare un semplice programma Java ad-hoc che memorizzasse solamente le pagine d'interesse, ovvero i profili pubblici degli utenti. Per determinare se il link era una pagina di profilo oppure no è stata analizzata la struttura dell'URL ed è emerso che in base all'ultima parte era possibile capire se la pagina corrispondente era una pagina di un profilo o una pagina della directory. Se la pagina era un profilo, veniva memorizzata, se la pagina era una directory veniva a sua volta navigata fino ad arrivare ai profili. Con questo semplice programma sono stati memorizzati circa 1 milione e 200 mila profili di utenti italiani. Nel capitolo 7 verrà mostrato il codice del programma.

Capitolo 3

Preprocessing e organizzazione dei dati

Una volta ottenute le pagine HTML dei profili pubblici degli utenti italiani di LinkedIn, occorre estrarre e manipolare i dati grezzi contenuti per renderli in un formato fruibile e adatto al processo di Data Mining.
Lo strumento più largamente utilizzato per la memorizzazione e la persistenza di dati è il database relazionale (RDBMS). Un database relazionale è un sistema di gestione di banche di dati basato sul modello relazionale. Per poter popolare il database con i profili degli utenti di LinkedIn, è necessario prima strutturare le informazioni da estrarre. È stato deciso di sviluppare un modello dei dati basato sugli oggetti per poi passare alla rappresentazione a tabelle.

3.1 Modello dei dati

I dati grezzi si presentano sotto forma di pagine HTML in cui ogni pagina contiene le informazioni su un utente. Ogni utente può decidere quali campi del proprio profilo rendere pubblici, perciò ogni profilo pubblico conterrà diversi tipi di informazioni. Per avere una panoramica generale sui tipi di informazioni presenti nei profili, sono stati analizzati svariati profili e così si è determinato l'insieme massimo possibile dei campi presenti. Sulla base di ciò, è stato sviluppato un modello dei dati rappresentato dal diagramma UML nelle figure 3.1, 3.2.

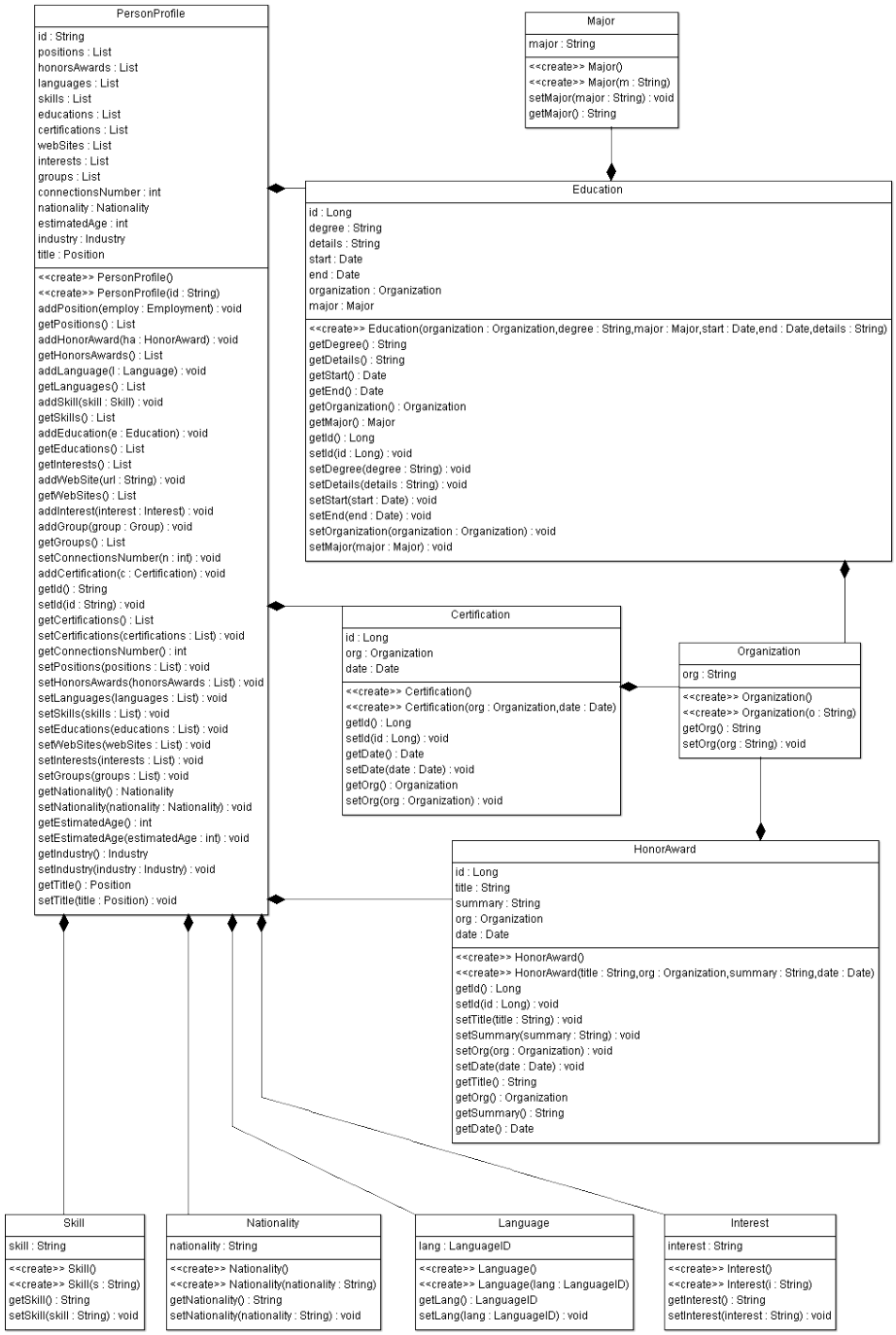

Figura 3.1: Diagramma UML dei dati (prima parte)

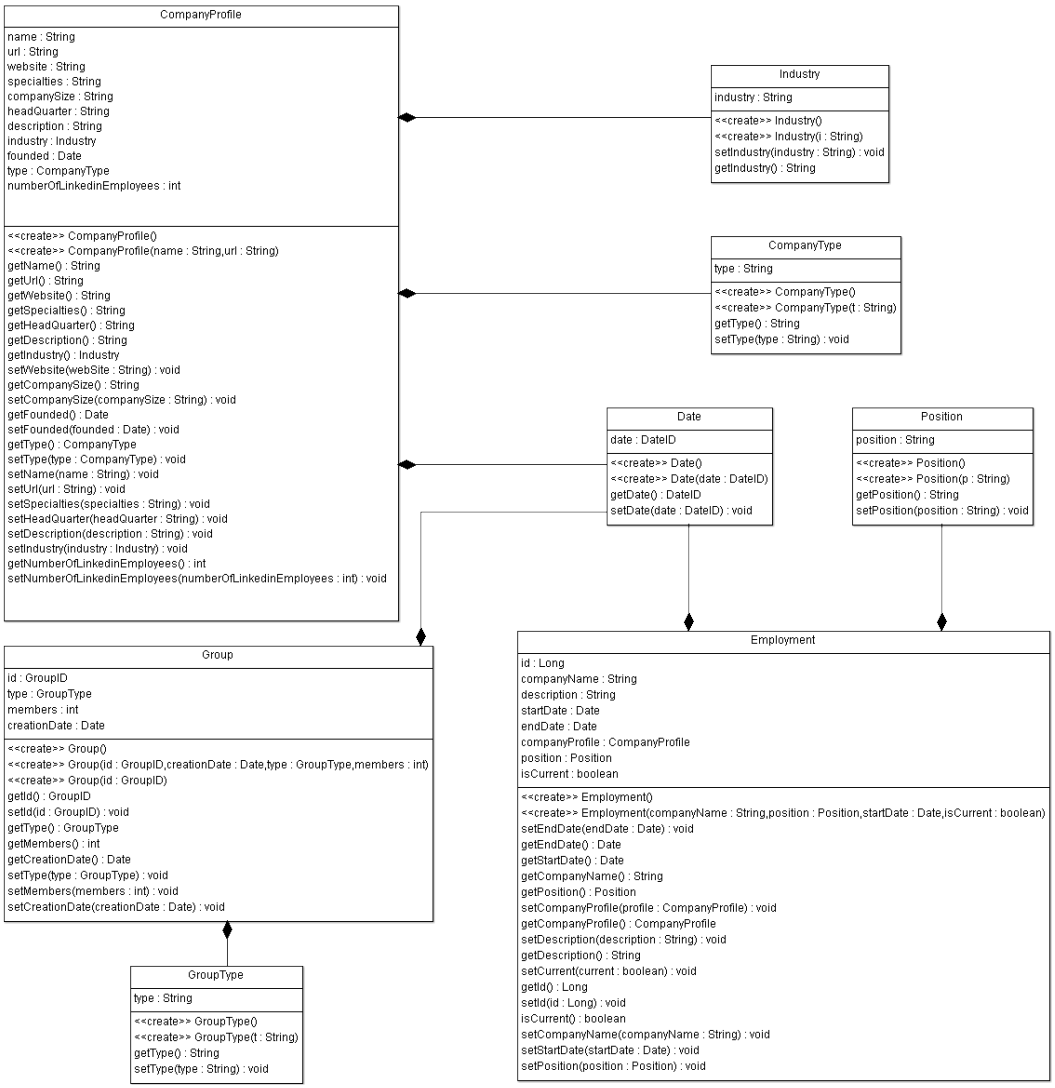

Figura 3.2: Diagramma UML dei dati (seconda parte)

3.2 Estrazione dei dati

Una volta progettato il modello dei dati, si è creato un progetto Java che estrae le informazioni dalle pagine HTML e le incapsula in oggetti del modello. Per fare questo ci si è avvalsi di un parser HTML, più precisamente della libreria jsoup. Jsoup offre diverse API per l'estrazione e la manipolazione dei dati contenuti in file HTML. Grazie a questa libreria, è stato semplice estrarre le informazioni contenute nei file sfruttando i tag specifici di ogni campo del profilo. Nel capitolo 7 verrà proposto un esempio di parsing HTML per estrapolare le informazioni sulle lingue.

3.3 Memorizzazione dei dati sul database

I DBMS sono i sistemi software più utilizzati per la gestione e la memorizzazione di database. Ci sono numerose alternative sul mercato, sia open-source che non. Le implementazioni open-source più diffuse sono MySQL, PostgreSQL e SQLite. È stato scelto PostgreSQL per la sua semplicità e completezza.
I dati da memorizzare hanno in questo momento una rappresentazione ad oggetti, perciò è necessario attuare una procedura di conversione dal modello ad oggetti al modello relazionale.

3.3.1 Object-Relational Mapping

La programmazione ad oggetti è attualmente la tecnica più utilizzata nello sviluppo di applicazioni. Un problema di questo approccio è che non offre soluzioni native al problema della persistenza, ovvero al problema della gestione di dati persistenti da parte di applicazioni OO. Il modello dei dati realizzato è basato su oggetti perciò nasce la necessità di rendere persistenti i dati attraverso DBMS relazionali, e quindi riuscire a gestire dati memorizzati in memoria secondaria. Il problema è che il paradigma OO nasce da principi diversi da quelli del modello relazionale, e le differenze tra i due sono notevoli.

Impedance Mismatch

Impedance mismatch è il termine usato per indicare la mancata corrispondenza tra il modello OO e il modello relazionale: il modello OO descrive il dominio in termini di oggetti e loro proprietà mentre il modello relazionale descrive il dominio in termini di relazioni tra valori. Il problema consiste nel far colloquiare l'applicazione che

parla in termini di oggetti con un DBMS che parla in termini di valori. Le principali differenze tra i due modelli riguardano:

- **Granularità**. I tipi di dato composti sono rappresentati nei linguaggi OO mediante classi di oggetti, mentre l'SQL non prevede alcun meccanismo standard per la definizione di tipi di dato composti
- **Ereditarietà e Polimorfismo**. Nei linguaggi OO l'ereditarietà è implementata attraverso l'utilizzo di super/sotto-classi e in alcuni casi anche il polimorfismo è utilizzabile. Nel modello relazionale non esiste il concetto di ereditarietà e polimorfismo
- **Identità**. In un DBMS l'uguaglianza è determinata dalla chiave primaria, mentre nel paradigma OO ci sono due concetti di uguaglianza: uguaglianza fisica di oggetti, ovvero di locazioni di memoria in cui sono memorizzati gli oggetti; uguaglianza semantica, implementata ad esempio in Java attraverso il metodo "equals"
- **Navigabilità**. Nei linguaggi OO il dominio si "naviga" da un oggetto all'altro attraverso le associazioni. Nei DBMS l'accesso ai dati relazionali avviene tramite join tra tabelle
- **Associazioni**. Nei linguaggi OO le associazioni sono rappresentate da relazioni unidirezionali mentre nei DBMS si utilizzano le foreign keys. Inoltre, non è possibile determinare la molteplicità di una associazione guardando il modello del dominio degli oggetti

Object Relational Mapping

La soluzione al problema della persistenza richiede un meccanismo per specificare la corrispondenza tra il dominio dell'applicazione e la base di dati che risolva opportunamente il problema dell'impedance mismatch. Tale meccanismo è comunemente chiamato Object-Relational Mapping (ORM). L'idea è quella di identificare le classi di oggetti dell'applicazione di tipo persistente e di fare in modo che i loro attributi e proprietà siano mappati su dati memorizzati in una base di dati relazionale. L'Object-Relational Mapping è stato oggetto di ricerca, infatti sono stati pubblicati numerosi paper a riguardo, per esempio: [27], [3], [39].
Le strategie di ORM sono diverse, e si distinguono l'una dall'altra dai diversi livelli di preparazione che deve avere il programmatore e dalla semplicità di realizzazione. Le possibili strategie sono tre:

- **Forza bruta**. È la tecnica più semplice e consiste nell'equipaggiare le classi di dominio con metodi che interagiscono direttamente con la base di dati, in pratica implementano le operazioni di create, read, update e delete (CRUD). Incorpora la logica di accesso ai dati sul DB nelle classi di dominio e nelle classi di controllo. È ragionevole quando l'applicazione è sufficientemente semplice e si può fare a meno di uno strato di incapsulamento. Un oggetto di una classe di dominio si popola con i dati presenti nel database a cui si accede tramite un driver ODBC (Open DataBase Connectivity) in questo modo:

  – si costruisce lo statement SQL necessario
  – si passa al driver
  – si ricevono i risultati dal database
  – si elaborano opportunamente

  I vantaggi di questo approccio sono: semplicità e rapidità di sviluppo; possibilità di accedere a db mal progettati o pre-esistenti all'applicazione.
  Gli svantaggi sono: presenza di un canale diretto dalla logica dell'applicazione al db; richiede a chi progetta l'applicazione di conoscere dettagliatamente il database; difficoltà di modifica del db, difficoltà di riuso dell'applicazione.

- **DAO** (Data Access Object). Prevede la realizzazione di uno strato dell'applicazione che ha il compito di gestire la comunicazione tra l'applicazione e il db. Il mapping è realizzato manualmente attraverso l'uso di ODBC/SQL. L'accesso al db viene incapsulato nelle classi DAO che: nascondono alla logica di business codice ODBC e SQL, forniscono un interfacciamento esplicito del codice, forniscono operazioni CRUD. Un oggetto di una classe di dominio per accedere al db:

  – invoca metodi di una classe DAO demandata a gestire gli accessi al db
  – la classe DAO costruisce lo statement SQL e lo passa al driver, riceve i risultati dal database e li riporta alla classe di business che l'ha invocata
  – la classe di dominio riceve i dati ed effettua le relative elaborazioni

  I vantaggi di questa tecnica sono: incapsulamento; possibilità di accedere a DB mal progettati o pre-esistenti all'applicazione; facilità di riuso dell'applicazione.
  Gli svantaggi sono: accoppiamento ancora presente tra persistence classes e

db; chi sviluppa l'applicazione (persistence classes) deve conoscere dettagli sul db; può essere dipendente dalla tecnologia.

- **Persistence Framework**. Prevede l'utilizzo di un framework predefinito per la gestione della persistenza (per esempio Hibernate). Ha l'obiettivo di liberare il programmatore quanto più possibile dalla necessità di scrivere codice SQL nella sua applicazione, infatti il codice SQL viene generato automaticamente sulla base di informazioni di meta-livello fornite dal programmatore (ad es. all'interno di file di configurazione). C'è un incapsulamento completo: il programmatore vede il db solo quando configura il framework. Dei metadati rappresentano la corrispondenza tra domain classes e tabelle, nonché le associazioni tra le domain classes. Oltre alle funzionalità di base CRUD, ne implementano anche altre, come per esempio le transazioni, la gestione della concorrenza, il caching ecc.
  I vantaggi dell'utilizzo di un framework sono: incapsulamento completo della logica di accesso al db; lo sviluppatore dell'applicazione non è tenuto a conoscere i dettagli sul db; facilità di riuso dell'applicazione e del persistence framework.
  Gli svantaggi sono: difficoltà di accedere a db mal progettati; decremento delle prestazioni (soprattutto se il framework non è costruito correttamente); può essere dipendente dalla tecnologia.

La soluzione adottata è quella del persistence framework, in quanto delega completamente la gestione del mapping in termini di codice SQL all'infrastruttura semplificando notevolmente il lavoro ed inoltre offre un numero elevato di funzionalità.

3.3.2 Hibernate

Hibernate è un piattaforma middleware open source che realizza il mapping tra oggetti di una applicazione Java e un database relazionale. Lo scopo principale di Hibernate è quello di fornire un mapping delle classi Java in tabelle di un database relazionale; sulla base di questo mapping Hibernate gestisce il salvataggio degli oggetti di tali classi su database (tipicamente attributi di oggetti per ciascun campo dati della tabella). Si occupa inoltre al rovescio del reperimento degli oggetti dal database, producendo ed eseguendo automaticamente le query SQL necessarie al recupero delle informazioni e la successiva reistanziazione dell'oggetto precedentemente ibernato (mappato su database). L'obiettivo di Hibernate è quello di esonerare lo sviluppatore dall'intero lavoro relativo alla persistenza dei dati. Hibernate si adatta

al processo di sviluppo del programmatore, sia se si parte da zero sia se si parte da un database già esistente. Hibernate genera le chiamate SQL e solleva lo sviluppatore dal lavoro di recupero manuale dei dati e dalla loro conversione, mantenendo l'applicazione portabile in tutti i database SQL [52]. Hibernate è un software molto articolato e ricco, tanto che sono stati prodotti libri sul suo utilizzo, per esempio [17] e [8].
Hibernate ha una architettura a livelli in modo da tenere isolato ogni livello da quello sottostante. Utilizza diverse API Java esistenti, tra cui JDBC (ODBC per applicazioni Java), Java Transaction API (JTA) e Java Naming and Directory Interface (JNDI). JDBC fornisce un livello rudimentale di funzionalità comune ai database relazionali, permettendo a quasi tutti i database con un driver JDBC di essere supportati da Hibernate. In figura 3.3 l'architettura di Hibernate.

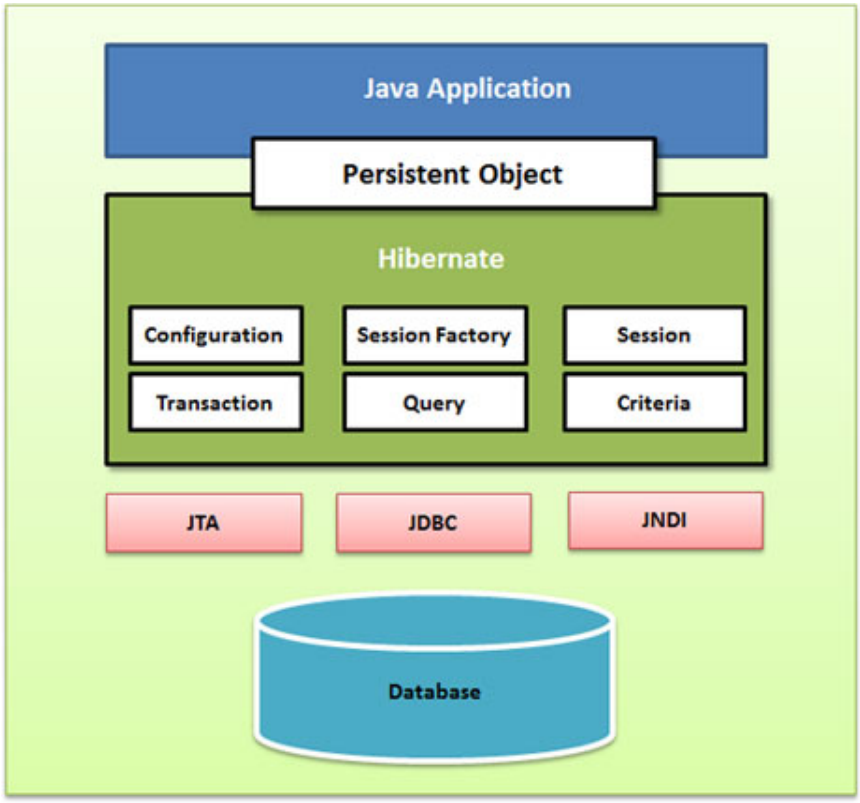

Figura 3.3: Architettura di Hibernate

Hibernate, come già detto, è un software molto articolato e completo, infatti

oltre ad implementare le operazioni di base CRUD offre una vasta gamma di funzionalità tra le quali supporto per le transazioni, gestione della concorrenza, gestione di eventi, caching/performance, grandi quantità di dati (batch processing), SQL nativo, HQL (Hibernate Query Language, simile a SQL ma completamente object-oriented).

3.3.3 Applicazione per la memorizzazione dei dati sul database

La memorizzazione dei dati dei profili utenti di LinkedIn sul database è stata realizzata attraverso un'applicazione Java utilizzando l'ambiente di sviluppo Eclipse e sfruttando il framework per l'ORM Hibernate.
Partendo dal modello dei dati definito precedentemente si è realizzato il mapping tra le classi Java e le tabelle relazionali. Le soluzioni per il mapping sono due: creare un file XML di mapping oppure utilizzare le annotazioni direttamente sulle classi Java. Il secondo metodo è quello adottato. Un'altra parte importante del mapping consiste nel file di configurazione *hibernate.cfg.xml* che definisce le informazioni di configurazione. Una volta realizzato il mapping è stata creata una applicazione che: analizza i file HTML dei profili ed estrae le informazioni di interesse; ottiene le pagine HTML delle aziende in cui gli utenti lavorano ed estrae le informazioni anche su di esse; incapsula le informazioni ottenute in oggetti del dominio; memorizza e rende persistente i dati attraverso Hibernate. I dettagli dell'applicazione saranno illustrati nel capitolo 7.
Il risultato finale è un database contenente circa 1 milione e 200 mila profili di utenti italiani e 130 mila profili di aziende. Lo schema risultante è nelle figure 3.4 e 3.5.

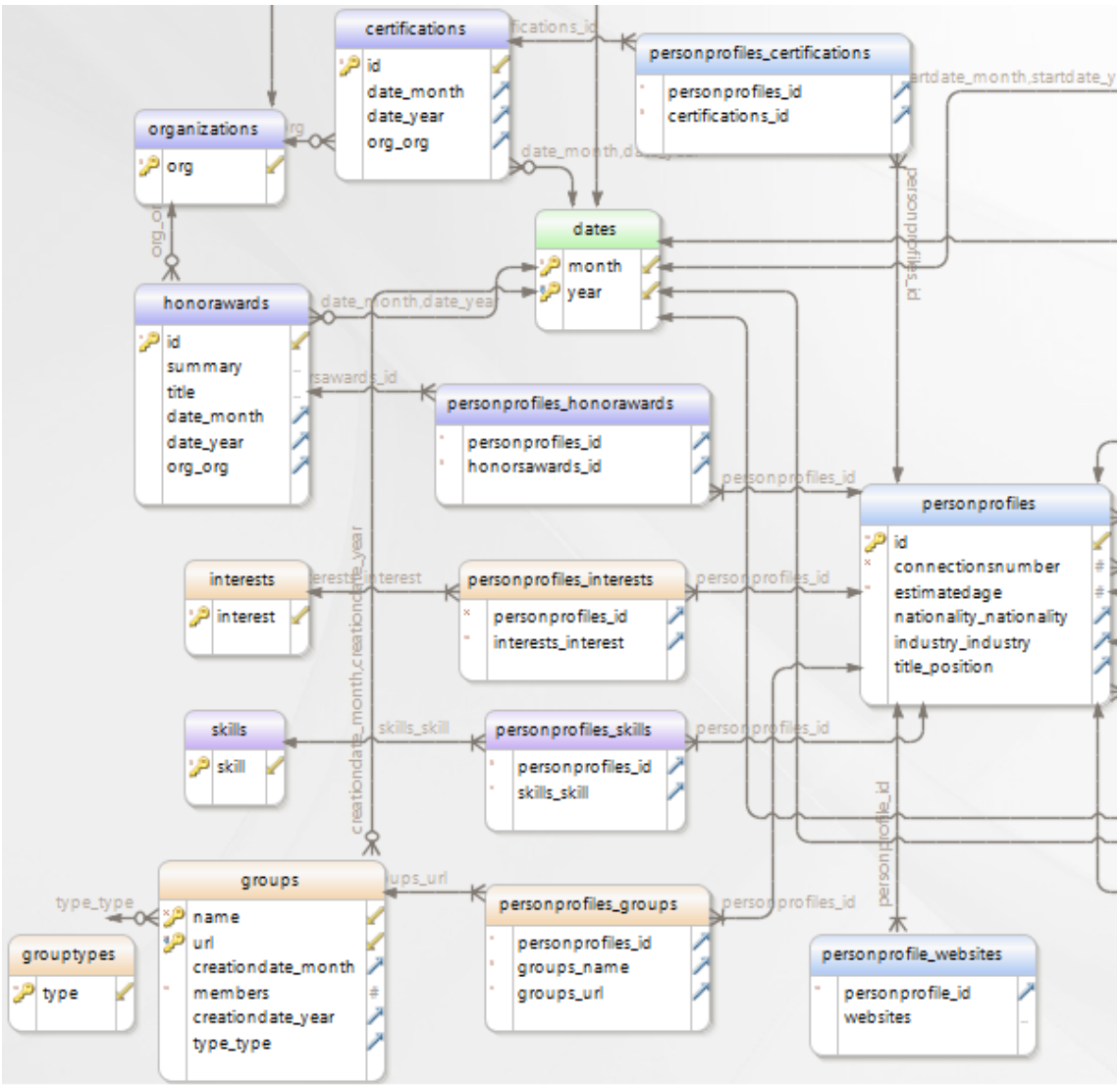

Figura 3.4: Schema database prima parte

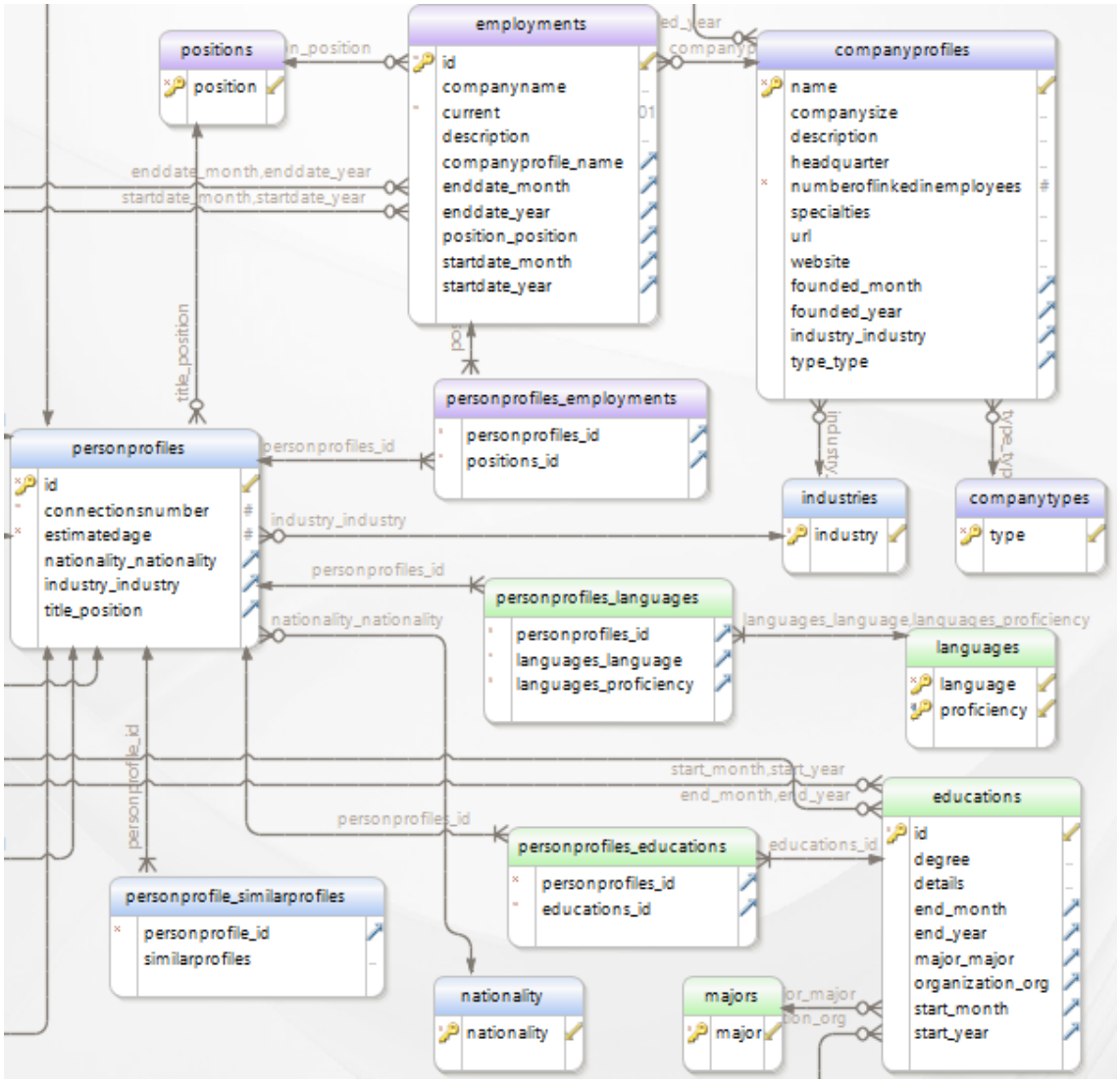

Figura 3.5: Schema database seconda parte

Capitolo 4

Analisi preliminare dei dati

Ogniqualvolta si ha a disposizione una grande quantità di dati su cui applicare tecniche di Data Mining e Text Mining, è fondamentale effettuare in primis un'analisi statistica sui dati per comprendere la distribuzione e l'andamento di ogni variabile. I dati raccolti sui profili degli utenti di LinkedIn presentano numerose caratteristiche interessanti su cui effettuare un'analisi statistica. L'attenzione è stata posta sugli aspetti più inerenti al mondo del lavoro e alle caratteristiche delle persone iscritte a LinkedIn, raggruppando i risultati in tre temi: aziende, posizioni lavorative e persone.

4.1 Aziende

Le aziende hanno un importante ruolo su LinkedIn, in quanto possono iscriversi al social network e pubblicare offerte di lavoro ed inoltre gli utenti possono aggiungere un collegamento con i profili delle aziende presso cui lavorano. Ad oggi più di 4 milioni di aziende sono iscritte attraverso una pagina aziendale.
Una caratteristica significativa riguarda sicuramente le dimensioni dell'azienda. Nell'immagine 4.1 si può vedere la distribuzione delle aziende in base alla dimensione in termini di dipendenti che dichiarano nella pagina aziendale.

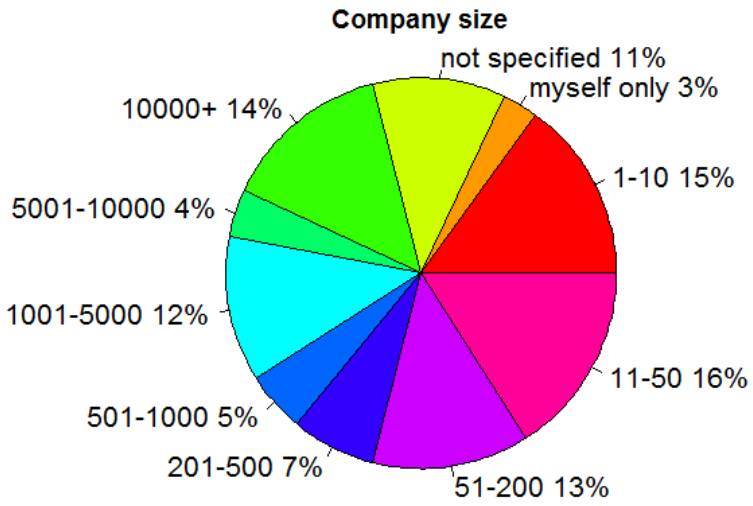

Figura 4.1: Distribuzione delle aziende in base alla dimensione

Dal grafico è possibile notare che non vi è una predominanza netta di una particolare dimensione, anche se le piccole-medie aziende (1-200 dipendenti) coprono quasi il 45% del totale.

Oltre alla dimensione è stata analizzata anche la tipologia di azienda sotto il punto di vista giuridico. Le possibili tipologie si distinguono in:

- *Privately held.* Società privata posseduta da organizzazioni non governative o da un numero relativamente stretto di soci o azionisti. Non commerciano le azioni al pubblico in borsa; le azioni sono offerte, possedute o scambiate privatamente.
- *Public company.* Aziende che consentono la vendita al pubblico dei loro titoli mobiliari (azioni, obbligazioni, ecc). Nonostante l'aggettivo pubblico, la public company è una società di diritto privato e di proprietà privata. Modello d'impresa tipico della realtà anglo-statunitense.
- *Educational institution.* Qualsiasi tipo di istituzione educativa, dalla scuola dell'infanzia all'università.

- *Non profit.* Organizzazione non a scopo di lucro che utilizza i propri guadagni per raggiungere i propri obiettivi invece di distribuirli come profitti o dividendi. Lo scopo di queste organizzazioni è affrontare temi socialmente utili.
- *Partnership.* Accordo in cui le parti, i partner, decidono di collaborare per promuovere i loro interessi reciproci. I soci possono essere di diversi tipi, come individui, imprese, scuole, governi o loro combinazioni.
- *Sole proprietorship.* Modello di business in cui l'azienda non è legalmente separata dal proprietario. Guadagni e perdite sono tassati direttamente sulla dichiarazione dei redditi del proprietario. Il proprietario è l'unico responsabile dell'azienda.
- *Government agency.* Agenzia governativa, ovvero una commissione permanente o semi-permanente del governo che è responsabile della supervisione e gestione di funzioni specifiche (ad esempio una agenzia di intelligence).
- *Self-employed.* Lavoratore autonomo che compie un lavoro intellettuale o manuale proprio a prezzo di un corrispettivo.

In figura 4.2 il grafico corrispondente.

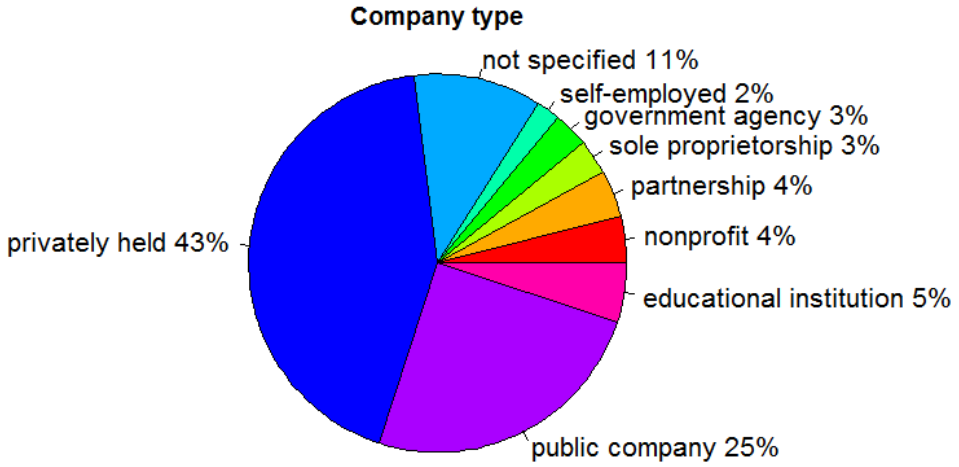

Figura 4.2: Tipologie di aziende dal punto di vista giuridico

Dal grafico si nota subito la grande predominanza delle società private, sia quelle che commerciano i loro titoli che quelle che non lo fanno. Raggiungono quasi il 70% della totalità.

Altro aspetto molto importante riguarda il settore delle aziende. Nell'istogramma 4.3 è possibile vedere i 10 settori più presenti nei dati raccolti.

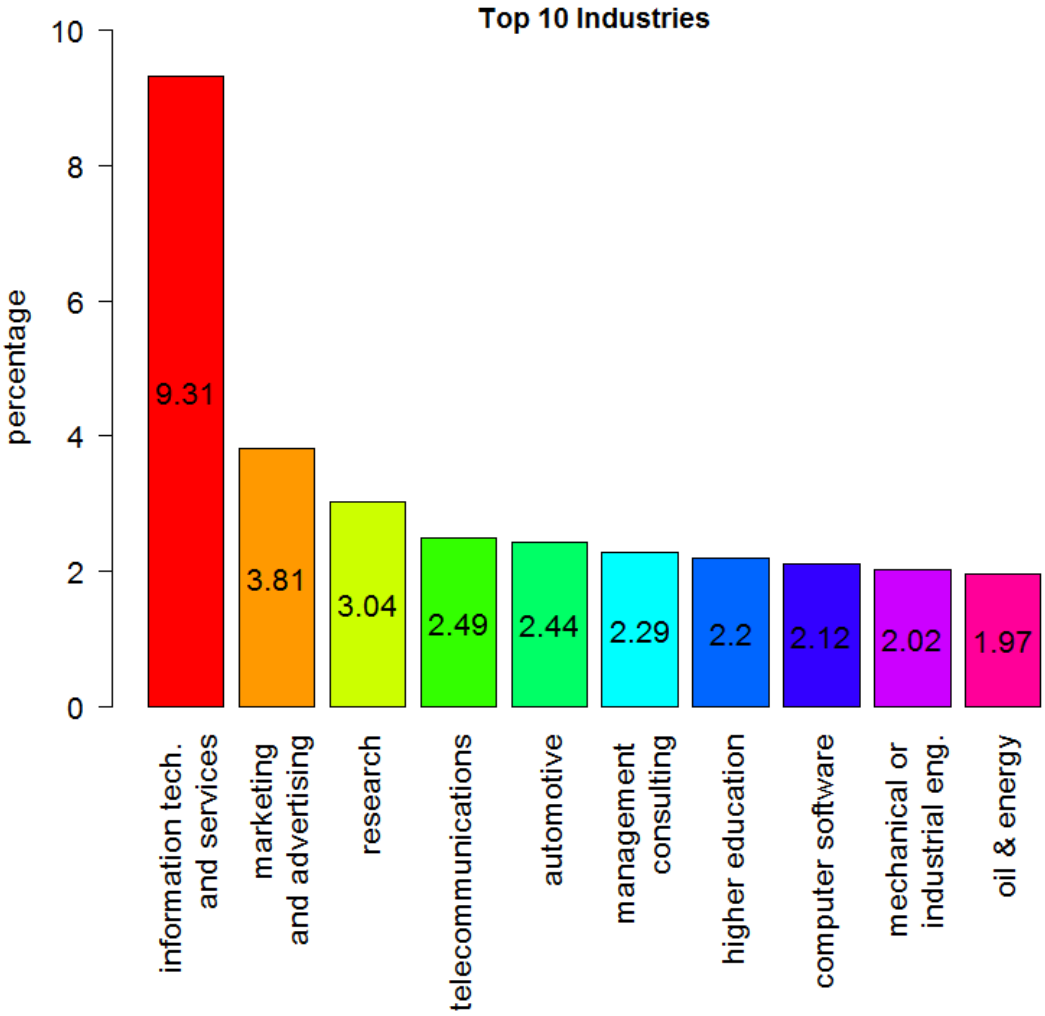

Figura 4.3: Settori aziendali più frequenti

Come era possibile prevedere, il settore dell'IT è quello più presente. Le aziende

di questo tipo infatti sono basate sull'informazione e perciò prestano maggiore attenzione a questo tipo di aspetto, ovvero la presenza dell'azienda in un social network orientato al business come LinkedIn.

L'ultima analisi sulle aziende riguarda il numero di utenti dipendenti di una azienda che possiedono un profilo LinkedIn e dichiarano di lavorare presso quella specifica azienda inserendo il collegamento con la pagina aziendale. In figura 4.4 il grafico corrispondente.

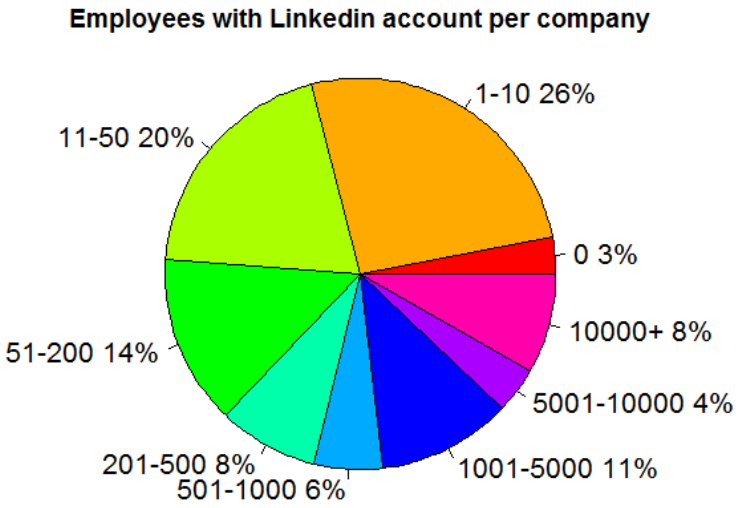

Figura 4.4: Numero di dipendenti con profilo LinkedIn per azienda

In questo caso non appare una predominanza assoluta. Da notare che questo grafico differisce leggermente da quello sulle dimensioni delle aziende, questo perché spesso non tutti i dipendenti di aziende presenti su LinkedIn hanno un proprio profilo, e quindi le percentuali per i numeri più bassi tendono a crescere e le percentuali per i numeri più grandi tendono a diminuire.

4.2 Posizioni lavorative

Su LinkedIn nel profilo utente è possibile inserire la propria storia lavorativa: dai lavori passati al/ai lavoro/i attuale/i, specificando periodo, mansione e azienda. È stata fatta distinzione tra le posizioni lavorative attuali e quelle passate e sono stati prodotti due grafici (4.5, 4.6) che riportano le 10 posizioni lavorative più frequenti.

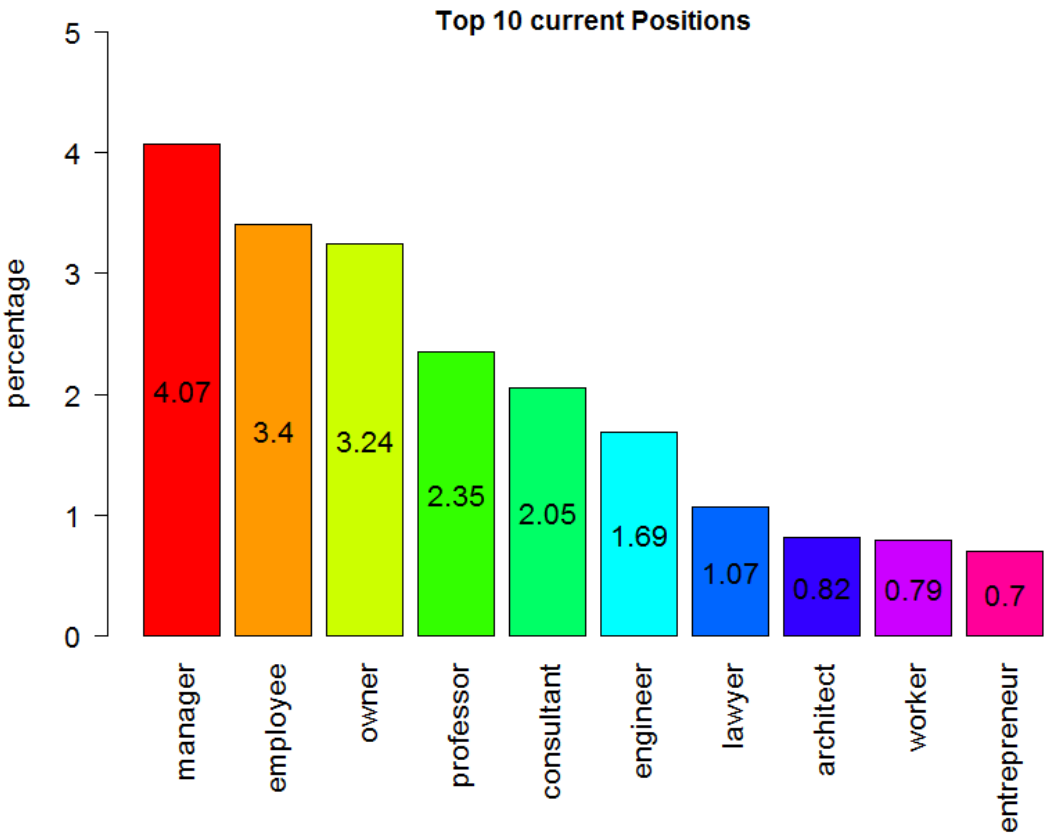

Figura 4.5: Posizioni lavorative attuali più frequenti

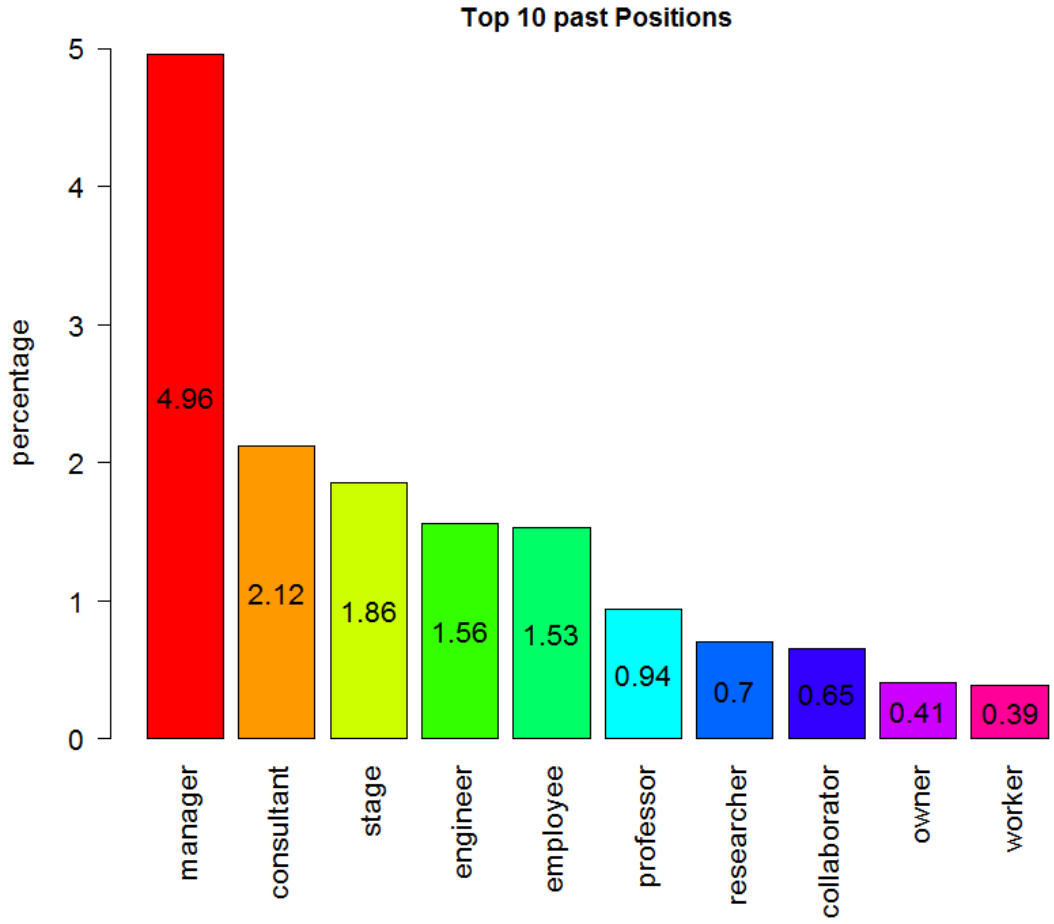

Figura 4.6: Posizioni lavorative passate più frequenti

Entrambi i grafici riportano il manager come figura più rappresentata. In generale è possibile notare che gli utenti di LinkedIn coprono ruoli abbastanza di alto livello in entrambi i casi, infatti sono presenti circa le stesse posizioni nei due grafici. L'unica differenza significativa riguarda la presenza dell'occupazione stage nelle posizioni passate, tra l'altro in posizione alta (terza); questo può essere indice del fatto che chi ricopre ruoli importanti (come la maggioranza degli utenti), all'inizio della sua carriera ha dovuto fare degli stage per potersi inserire e poi crescere di livello.

4.3 Persone

L'ultimo tema riguarda le caratteristiche delle persone iscritte su LinkedIn. In particolare sono stati analizzati: gli skill che gli utenti dichiarano di avere, le lingue parlate, gli interessi e l'educazione.

Primo aspetto, fondamentale nell'ambito del lavoro, sono le capacità (skill) che possiede una persona. Nell'analisi sono stati estrapolati i 10 skill più frequenti all'interno delle persone prese in esame. Nel grafico 4.7 i risultati.

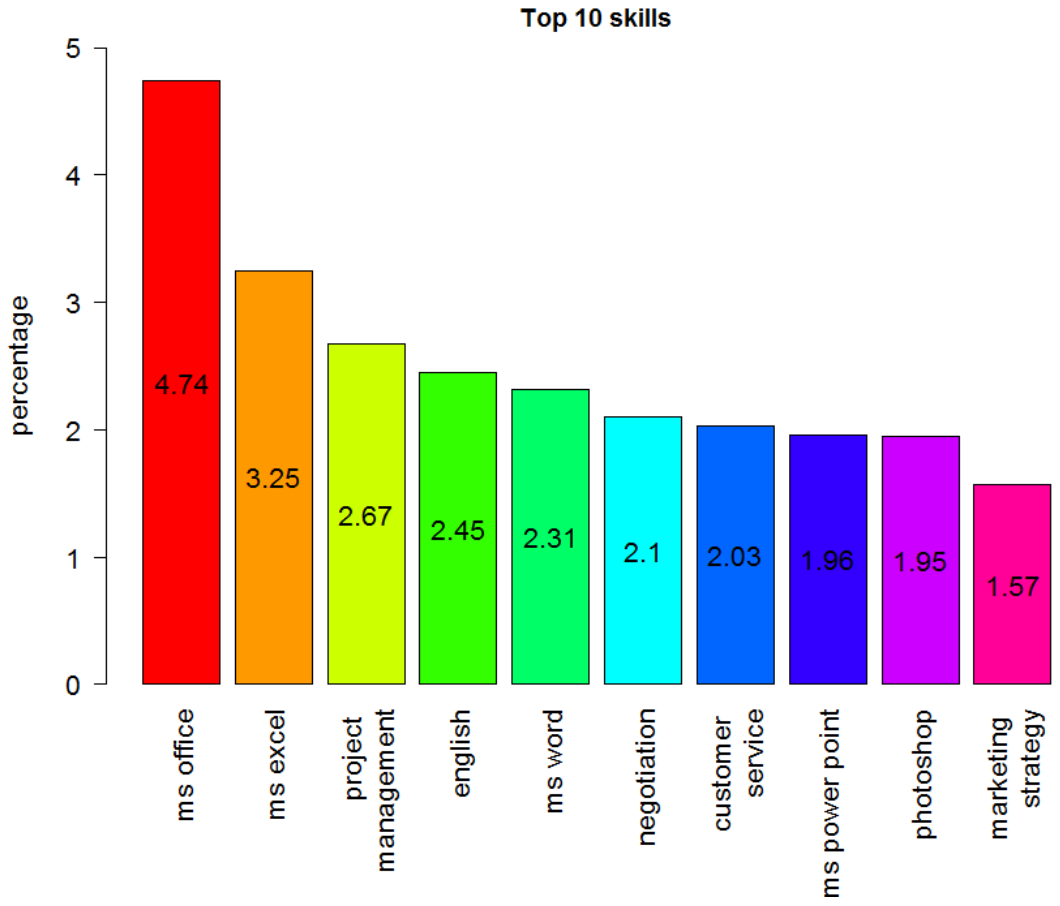

Figura 4.7: Skill più frequenti

Dal grafico si nota subito come le capacità informatiche, ovvero la capacità di

utilizzare un computer e i programmi più noti di office automation, occupino la maggioranza degli skill più frequenti. Questo è prova del fatto che al giorno d'oggi l'informatizzazione delle attività economiche è totale e quindi è necessario avere queste capacità per potersi inserire nel mondo del lavoro. Altra nota importante è la presenza della lingua inglese come skill: sopratutto nel settore IT la conoscenza dell'inglese è diventata essenziale in quanto le aziende sono sempre più internazionali e i termini tecnici sono tutti inglesi.

Altra caratteristica analizzata riguarda le lingue conosciute dagli utenti. Nella figura 4.8 sono riportate le 5 lingue più conosciute.

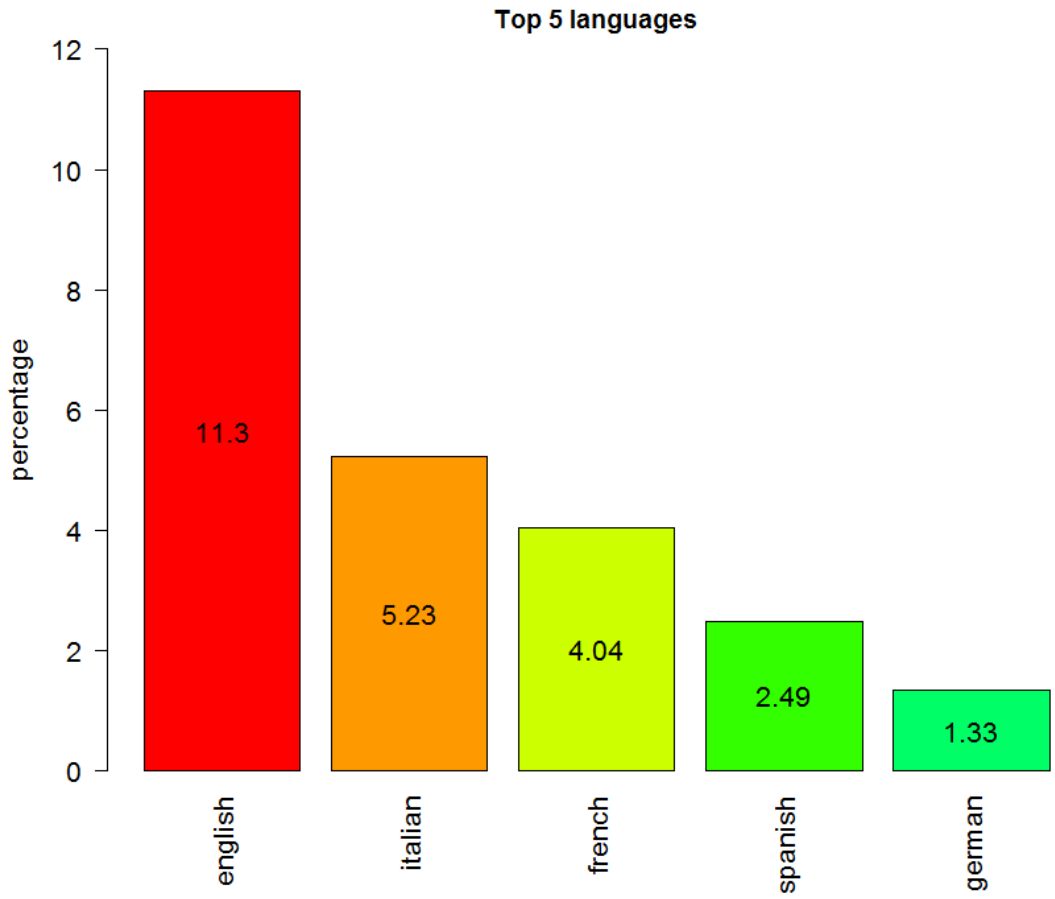

Figura 4.8: Lingue più conosciute

L'inglese ancora una volta predomina, a prova del fatto che come precedentemente detto esso è diventato fondamentale nel mondo del lavoro. Gli utenti LinkedIn su cui è stata fatta l'analisi sono italiani, perciò sarebbe stato logico che l'italiano fosse al primo posto; probabilmente molti non lo hanno inserito perché davano per scontata la conoscenza della loro lingua madre.

L'ultimo grafico presenta gli interessi degli utenti. In figura 4.9 i primi 10 interessi più frequenti.

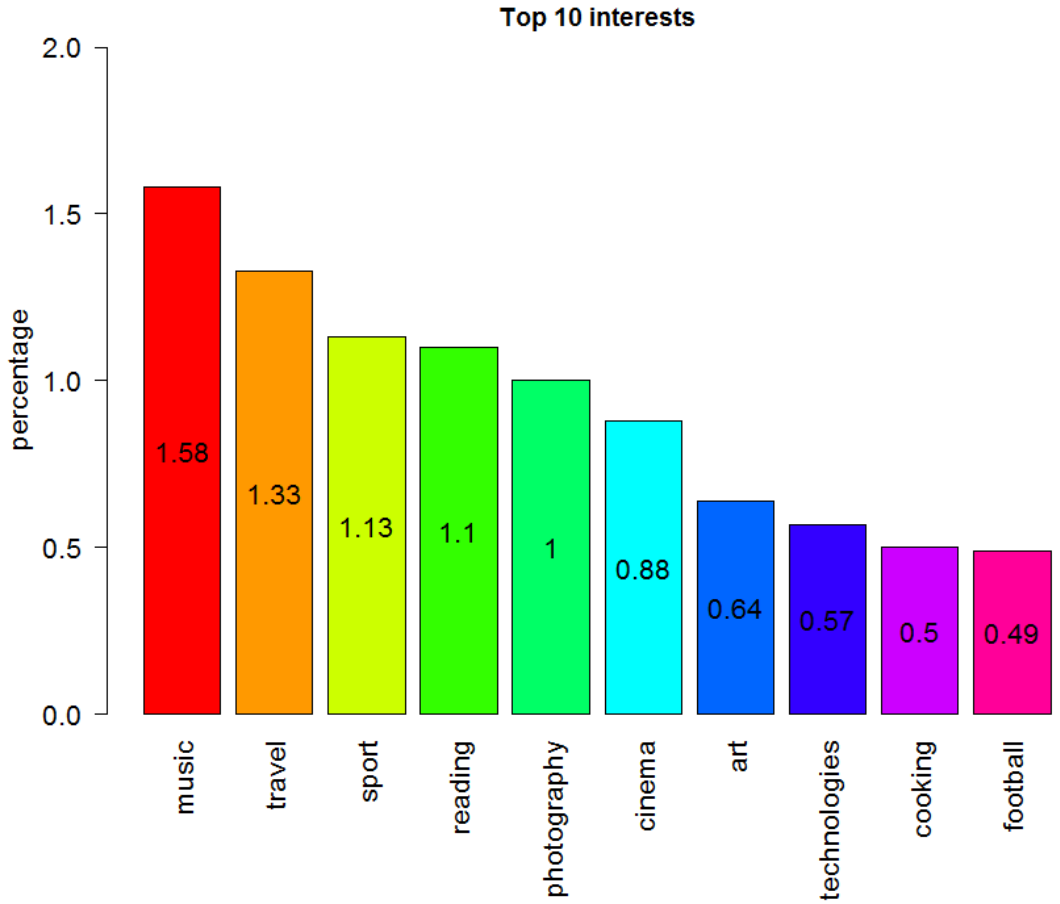

Figura 4.9: Interessi più frequenti

Dall'analisi non emerge niente di particolarmente notevole; gli interessi più gettonati sono anche quelli più classici come musica, viaggi e sport.

Per quanto riguarda l'educazione, dalle analisi è emerso che il 10,81% degli utenti dichiara di avere almeno una laurea e il 4,45% dichiara di avere al massimo un diploma di scuola superiore. Le basse percentuali non indicano una poca educazione degli utenti ma molto probabilmente c'è un gran numero di utenti che non hanno specificato la propria educazione. Il dato interessante è che ci sono più persone con una laurea rispetto a chi ha soltanto un diploma, a conferma del fatto che in generale LinkedIn è utilizzato da una fascia istruita di utenti.

Capitolo 5

Skill recommendation

I dati ottenuti dai profili degli utenti LinkedIn e poi organizzati in un database sono ora pronti per la fase di mining. Per prima cosa è stata fatta una ricerca su eventuali lavori già fatti riguardanti l'elaborazione di dati provenienti da LinkedIn. Il risultato della ricerca è un paper molto recente (ottobre 2014) pubblicato da sei ricercatori di LinkedIn, in cui presentano un lavoro di topic extraction a larga scala che include la realizzazione di una folksonomy (categorizzazione) di skill e di un sistema di recommendation di skill, [6]. Questo paper rappresenta la fonte di ispirazione su cui ci si è basati per la realizzazione del lavoro seguente. Il capitolo si articola in due parti: realizzazione di un clustering di skill e sviluppo di un metodo non supervisionato di skill recommendation basato sul sistema proposto nel paper. La skill recommendation consiste nel suggerire agli utenti gli skill che potrebbero avere in base alle loro caratteristiche.

5.1 Skill clustering

La prima parte del paper si sviluppa nella realizzazione di una categorizzazione di skill attraverso un clustering. Si parte con un insieme di skill opportunamente filtrato per poi arrivare ad un clustering realizzato tramite un algoritmo KMeans. Il metodo con cui viene realizzato il clustering è stato d'ispirazione per realizzare un clustering di skill sui dati dei profili ottenuti precedentemente.

5.1.1 Similarità di Jaccard tra gli skill

Nel database ottenuto dai profili degli utenti è emerso che sono presenti più di 170 mila skill distinti. Per rendere il problema più gestibile è stato fatto un filtering degli skill basato sulla frequenza. Questo è stato possibile eseguendo una query sul database in cui si ottenevano le coppie utente/skill in cui lo skill comparisse almeno n volte. N è stato fissato a 20 ottenendo circa 7500 skill distinti, considerandolo un buon compromesso tra significatività e trattabilità. Il risultato della query è stato poi memorizzato in un file .csv facilmente gestibile da R.
Nel paper viene illustrato come ottenere la similarità di Jaccard tra gli skill che servirà poi per realizzare il clustering. Per prima cosa si determinano le occorrenze di ogni skill: si definisce s_i come uno skill, d_k come un profilo utente, $S = \{s_1, s_2, ..., s_i\}$ l'insieme degli skill, $D = \{d_1, d_2, ..., d_k\}$ l'insieme dei profili utente, $f_{i,k}$ l'occorrenza dello skill s_i all'interno del profilo d_k. L'occorrenza totale F_i dello skill s_i in D è:

$$F_i = \sum_{k=1}^{d' \in D} f_{i,k}$$

Successivamente si definisce $f_{i,j,k}$ come la co-occorrenza degli skill s_i e s_j nello stesso profilo d_k. La co-occorrenza totale tra due skill i, j è:

$$F_{i,j} = \sum_{k=1}^{d' \in D} f_{i,j,k}$$

Infine la similarità di Jaccard $J_{i,j}$ tra due skill è calcolata come:

$$J_{i,j} = \frac{F_{i,j}}{F_i + F_j - F_{i,j}}$$

La parte successiva di realizzazione del clustering non è stata ripresa dal paper perché le informazioni riportate non sono sufficienti per la riproduzione esatta del metodo e non siamo riusciti a replicarne i risultati.

5.1.2 Clustering

Le tecniche di clustering sono molteplici, una prima grande distinzione è quella tra clustering partizionante e clustering gerarchico. Il primo è una divisione degli oggetti in sottoinsiemi (cluster) non sovrapposti, ogni oggetto appartiene esattamente a un cluster. Il secondo è un insieme di cluster annidati organizzati come un albero gerarchico. In tutti i casi le tecniche di clustering si basano su misure relative alla

somiglianza (o diversità) tra gli elementi.
Nel paper viene applicato l'algoritmo KMeans che è di tipo partizionante, mentre in questo caso si è optato per un clustering gerarchico. Il clustering gerarchico produce un insieme di cluster organizzati come un albero gerarchico. Può essere visualizzato come un dendrogramma, un diagramma ad albero che mostra la sequenza di fusioni tra cluster. A valori diversi sulle ordinate del dendrogramma possono corrispondere clustering composti da un numero diverso di elementi. Esistono due approcci per la creazione di un clustering gerarchico: divisivo in cui si parte con un unico cluster che include tutti gli elementi e ad ogni step si separa il cluster più "lontano" fino a che i cluster non contengono un solo elemento; agglomerativo in cui si parte con cluster formati da elementi singoli e ad ogni step si fondono i due cluster più "vicini" fino a che non rimane un solo cluster. L'approccio aggomerativo è il più comune ed è anche quello utilizzato. L'operazione fondamentale è il calcolo della similarità tra due cluster e può essere implementata in diverse maniere:

- *Single Linkage*, è la minima distanza tra due punti dei cluster
- *Complete Linkage*, è la massima distanza tra due punti dei cluster
- *Group Average*, è la media delle distanze tra tutti i punti dei cluster
- *Altri metodi*

Sono stati provati diversi metodi e quello più adatto alla struttura dei dati si è rivelato il Complete Linkage. L'algoritmo di clustering necessita di una misura di dissimilarità tra ogni elemento, perciò è stata sfruttata la similarità di Jaccard ottenuta precedentemente. Dato che la similarità di Jaccard va da 0 a 1, sono stati sottratti a 1 i valori di similarità per ottenere un valore di distanza. Il risultato dell'applicazione dell'algoritmo è un dendrogramma al cui vertice confluiscono tutti i cluster formati ai livelli inferiori. Per poter visualizzare i cluster a diversi livelli di dettaglio è necessario tagliare il dendrogramma. Sono stati provati diversi livelli di taglio e analizzato visivamente i cluster risultanti. Per non avere cluster nè troppo grandi (che quindi assocerebbero skill appartenenti a diversi settori) nè troppo piccoli (che quindi dividerebbero skill semanticamente correlati), si è deciso di tagliare il dendrogramma in modo da ottenere 76 cluster distinti. Ispezionando manualmente ogni cluster si è potuto accertare che i cluster associavano correttamente skill appartenenti allo stesso campo semantico. Di seguito alcuni esempi di cluster ottenuti.

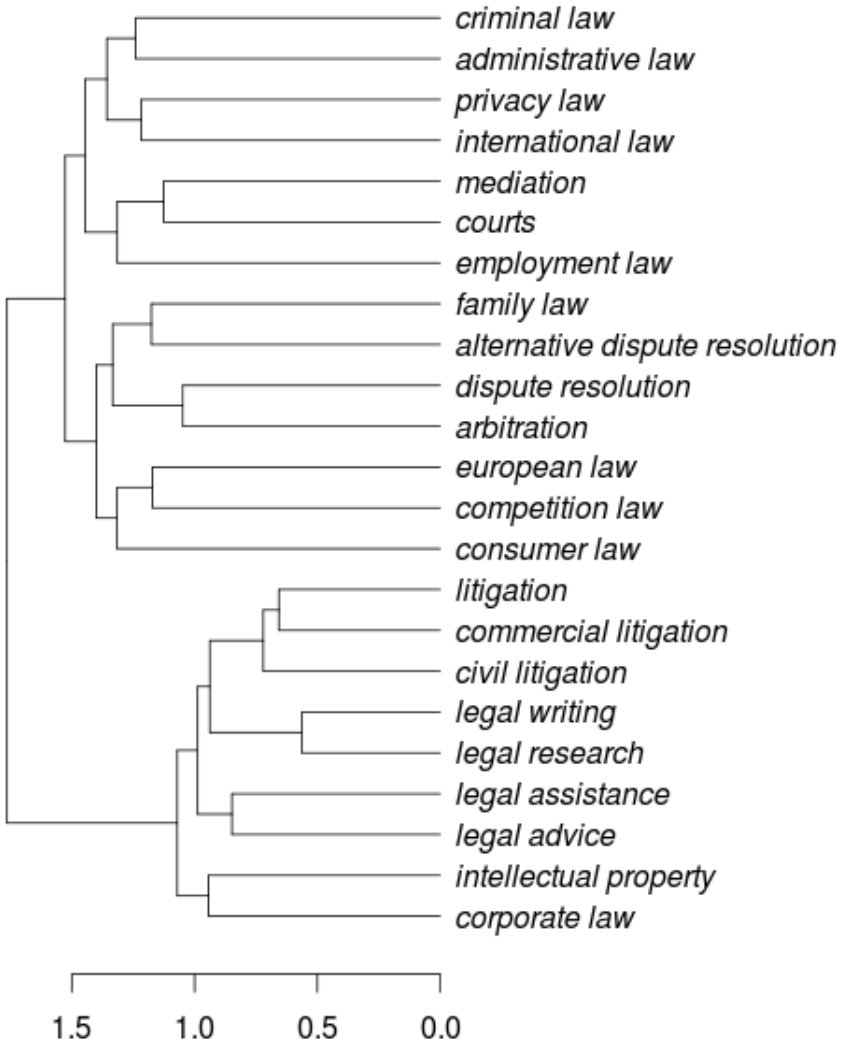

Figura 5.1: Cluster che raggruppa gli skill inerenti alla giurisprudenza

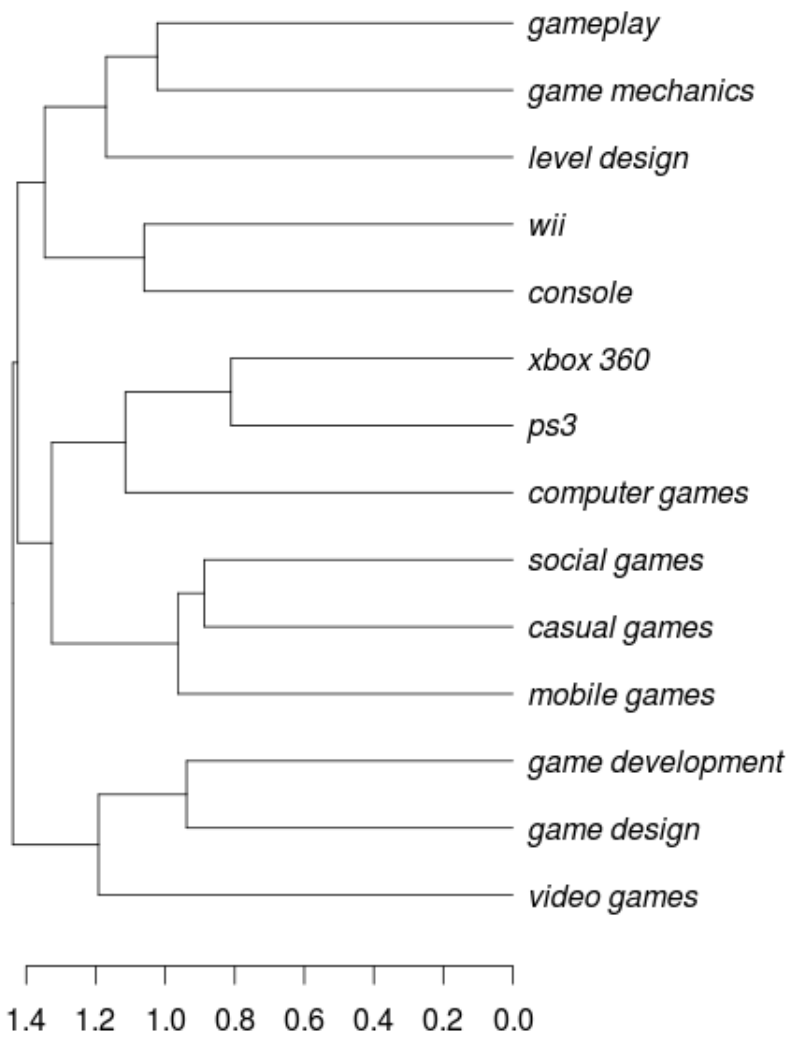

Figura 5.2: Cluster rappresentante il tema dei video games

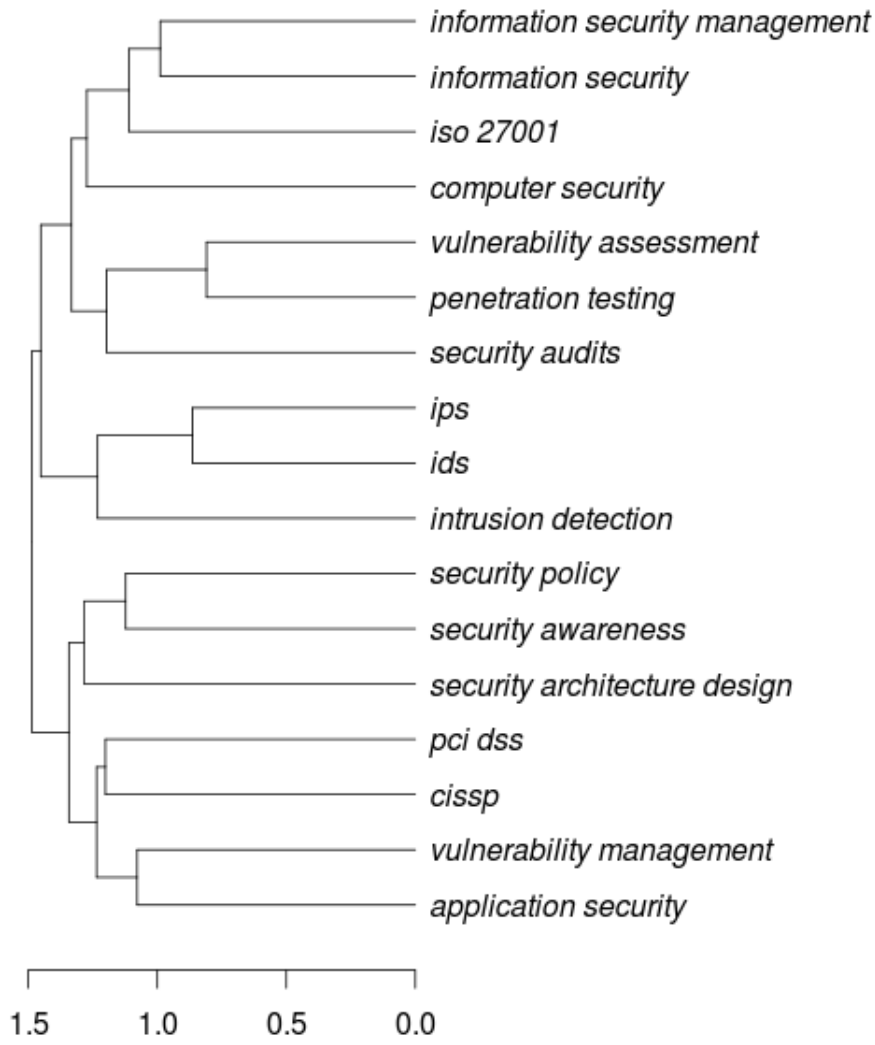

Figura 5.3: Cluster riguardante la sicurezza informatica

Tutti i dettagli implementativi saranno illustrati nel capitolo 7.

5.2 Metodo di skill recommendation

A seguito della realizzazione della folksonomy di skill, il paper prosegue descrivendo la realizzazione di un algoritmo di inferenza e recommendation di skill. L'obiettivo è quello di aiutare gli utenti nel momento della compilazione del profilo suggerendo gli skill che essi potrebbero avere, basandosi su informazioni dell'utente. I più comuni sistemi di recommendation suggeriscono un grande insieme di elementi, come nella recommendation di film o di prodotti, [30], [34]. Nel caso della recommendation di skill, invece, il numero di skill rilevanti è relativamente limitato, infatti il massimo numero di skill che un utente può aggiungere al suo profilo è 50. Questo problema è similare alla recommendation di tag per contenuti come fotografie o articoli, discusso in [46] e [33].
La recommendation di skill necessita di un meccanismo per stimare la probabilità che un utente abbia un determinato skill. Nel paper hanno considerato diverse possibilità, come per esempio un modello di regressione logistica oppure un modello di regressione basato sul collaborative filtering. Entrambi i metodi presentavano alcune difficoltà perciò hanno optato per un semplice modello di classificazione Naive Bayes.

5.2.1 Classificatore Naive Bayes

Il profilo di un utente contiene un insieme di feature standard come: azienda in cui lavora (company), posizione lavorativa (position), settore (industry),gruppi di appartenenza (group) ecc. Queste feature possono essere utili per inferire gli skill dell'utente. Dato un profilo che contiene un insieme di queste feature $a_1, ..., a_n$, si definisce la probabilità di avere uno skill come $P(s|a_1, ..., a_n)$ dove $s \in S$ e $S = s_1, s_2, ..., s_n$ è la folksonomy di skill. Sia $n(a_i, s)$ il numero di co-occorrenze della feature a_i e dello skill s nei profili del training set e $n(s)$ il numero totale di utenti con quello skill. Secondo il modello Naive Bayes, in cui si assume indipendenza tra le feature, si definisce:

$$P(s|a_1, ..., a_n) = \frac{1}{Z} P(s) \prod_{i=1}^{n} P(a_i|s)$$

dove $Z = P(a_1, ..., a_n)$, $P(s)$ è la probabilità a priori di uno skill $s \in S$ e $P(a_i|s)$ è la probabilità di a_i dato s definita come

$$P(a_i|s) = \frac{n(a_i, s)}{n(s)}$$

Per ottenere lo skill più probabile che un utente potrebbe avere si calcola:

$$\arg\max_{s \in S} P(s) \prod_{i=1}^{n} P(a_i|s)$$

Dato che $n(a_i, s)$ potrebbe essere nullo per qualche feature e skill, si applica un correttore Laplaciano per evitare che tutta la probabilità si azzeri. Quindi si ottiene:

$$P(a_i|s) = \frac{1 + n(a_i, s)}{|W_i| + n(s)}$$

dove $|W_i|$ è la grandezza del vocabolario per la feature i.

5.2.2 Feature selection

Il classificatore Naive Bayes prevede di calcolare le probabilità condizionate tra tutte le coppie possibili feature-skill. Come è possibile immaginare sia il numero di feature distinte che quello di skill è molto alto, perciò il problema potrebbe diventare di difficile o addirittura impossibile gestione. Nel paper viene quindi proposta una fase di feature selection precedente all'applicazione del Naive Bayes. Gli approcci possibili suggeriti dal paper sono due: applicare un filtro basato sulla frequenza oppure utilizzare la mutual information tra coppie feature-skill per valutare quali feature selezionare. Il primo metodo viene scartato perché secondo il paper penalizza le skill rare se si considera una frequenza minima troppo elevata. Dopo aver calcolato la mutual information tra ogni coppia feature-skill sono state provate diverse soglie minime per stabilire quale portasse ai risultati migliori.
La numerosità sia di skill che di feature dei dati a disposizione è sicuramente inferiore a quella dei dati del paper, ma molto probabilmente anche la disponibilità di risorse è molto differente perciò anche in questo caso si è realizzata una feature selection. Inizialmente si è provato a riprodurre il calcolo della mutual information tra coppie feature-skill proposto nel paper ma per insufficienza di risorse hardware non è stato possibile realizzarlo (il lavoro è stato eseguito su una macchina con 7 processori e 16 GB di Ram). Si è quindi optato per il filtro basato sulla frequenza. Le feature a disposizione sono industry, position, company e group; industry non viene filtrato perché la sua numerosità è già bassa (288 industry distinte), mentre le position sono state filtrate eliminando quelle con frequenza minore di 10 e per le altre feature la soglia minima sale a 20.

5.2.3 Risultati

Metodo paper

Il classificatore Naive Bayes descritto sopra è stato utilizzato per inferire gli skill degli utenti, utilizzando come feature alcune informazioni sugli utenti presenti nel profilo. Il numero massimo di skill inseribili nel profilo è 50, perciò sono stati selezionati i 50 skill più probabili determinati dalla classificazione. I dettagli implementativi del metodo saranno illustrati nel capitolo 7.
Prima di passare alla classificazione è utile fare una panoramica su quanti skill gli utenti inseriscono effettivamente nel loro profilo. Nel grafico 5.4 viene illustrata la distribuzione della frequenza del numero di skill per persona.

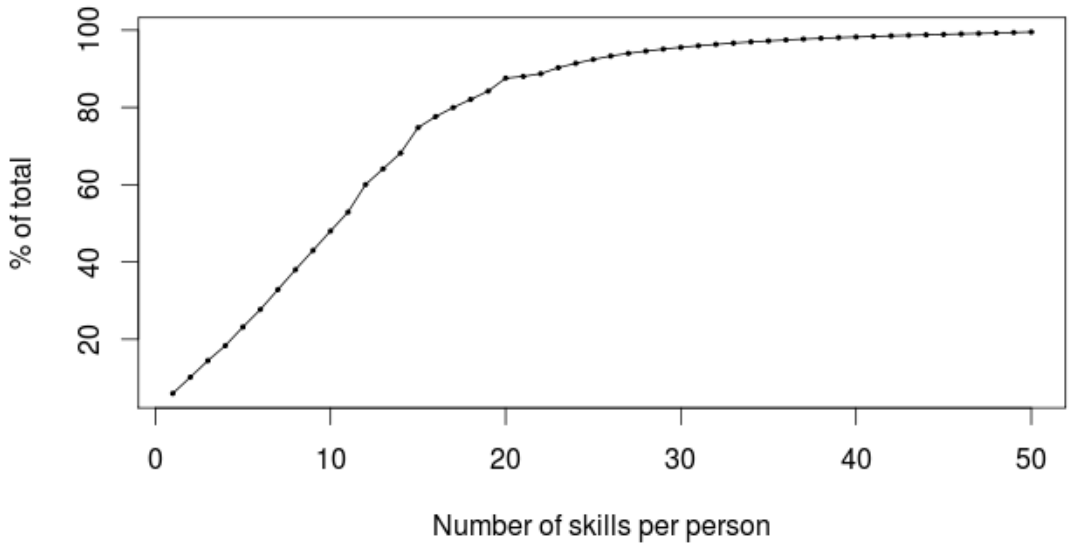

Figura 5.4: Numero di skill per persona

Per misurare la correttezza del metodo, è stata calcolata la recall dividendo il numero di skill suggeriti che l'utente effettivamente possedeva con il numero totale di skill posseduti dall'utente. Nel paper sono riportati i risultati della classificazione utilizzando diversi feature set.

Feature set	Recall@50
industry	0.45
industry + company + function	0.60
industry + company + function + title	0.66
industry + company + function + title + group	0.71

Tabella 5.1: Risultati recommendation di skill del paper

Lo stesso metodo utilizzato nel paper è stato applicato al dataset ottenuto dai profili pubblici degli utenti. Per avere un confronto col paper, sono stati utilizzati i feature set quanto più possibili simili a quelli utilizzati nel paper. Non è stato possibile avere feature set identici perché non tutte le feature utilizzate nel paper erano presenti nei profili pubblici degli utenti. In tabella 5.2 sono riportati i risultati.

Feature set	Feature distinte	Recall@50
industry	288	0.493
industry + position	15000	0.499
industry + company	17000	0.496
industry + position + group	51000	0.512
industry + position + company + group	68000	0.516

Tabella 5.2: Risultati recommendation di skill

Per realizzare la classificazione il dataset filtrato sulla base della frequenza di skill e feature è stato suddiviso in training set e test set. Il training set conta circa 145 mila profili mentre il test set circa 80 mila; le skill sono oltre 21 mila distinte. Il paper non cita informazioni quantitative su feature e skill, riporta soltanto che a seguito del calcolo della mutual information è stata scelta una soglia del 25esimo percentile per ridurre il training set ad una dimensione gestibile.
Mettendo a confronto i risultati del paper con quelli del nostro dataset è possibile notare che utilizzando come feature set soltanto l'industry la recall è leggermente inferiore nel paper; in entrambi i casi aumentando il feature set i risultati migliorano anche se nel paper il miglioramento è più netto. Le differenze possono essere dovute a diversi fattori: in primo luogo i dati su cui è stata fatta la classificazione, pur provenendo entrambi da profili LinkedIn, sono sicuramente diversi sia come numero che come tipo in quanto l'estrazione è stata fatta in tempi diversi e la disponibilità di risorse dei ricercatori LinkedIn è maggiore; la fase di feature selection precedente alla classificazione non è la stessa perché i ricercatori hanno filtrato i dati sulla

base della mutual information tra coppie feature-skill mentre nel nostro caso è stato fatto un semplice filtro basato sulla frequenza di skill e feature. Tutti questi motivi giustificano una differenza di risultati a parità di metodo che comunque rimane accettabile.

Classificatore basato sulla frequenza

Per dimostrare l'utilità del metodo di recommendation di skill è stato pensato di effettuare una classificazione "dummy", per poi confrontare i risultati. Il primo classificatore realizzato si basa sulla frequenza degli skill, ovvero suggerisce sempre i 50 skill più frequenti. Per questo tipo di classificazione non è stato necessario effettuare più ripetizioni perché gli skill risultanti rimangono sempre gli stessi. La classificazione è stata eseguita su tutti i feature set descritti in precedenza ma ovviamente il risultato rimane identico perché le feature non influiscono sulla classificazione; l'unico aspetto determinante è la frequenza con cui gli skill appaiono nei profili.
La recall@50, calcolata come nel caso precedente si ferma allo 0.288, molto lontana dai risultati ottenuti con il metodo di recommendation di skill.

Classificatore casuale basato sulla frequenza

Un'ulteriore prova dell'efficacia del metodo di recommendation di skill è stata apportata effettuando una classificazione casuale basata sulla frequenza degli skill. Il classificatore lavora in questo modo: è stato costruito un intervallo che va da 0 a 1 in cui ogni segmento rappresenta uno skill e la sua lunghezza è proporzionale alla frequenza dello skill, tanto più lo skill è frequente tanto più ampio sarà il suo segmento e quindi la probabilità di essere selezionato; per ogni profilo di test sono stati estratti 50 numeri random che vanno da 0 a 1, ogni numero cadrà in un certo segmento dell'intervallo che corrisponde ad uno skill, in questo modo si ottengono 50 skill distinti; gli skill ottenuti vengono confrontati con gli skill reali dell'utente e calcolata la recall@50 come nei casi precedenti.
La classificazione sull'intero test set è stata ripetuta 20 volte e fatta la media delle recall risultanti. La recall@50 media si attesta a 0.108, in questo caso ancora peggiore del classificatore basato sulla frequenza.

Capitolo 6

Job recommendation

Dopo aver implementato la recommendation di skill ed ottenuto buoni risultati il passo successivo è quello di cercare di sfruttare al massimo i dati a disposizione elaborando nuove strategie in linea con l'obiettivo generale della tesi: utilizzare i social network e i dati ottenibili da essi nella ricerca ed offerta di lavoro in tutti i suoi aspetti.
Il paper ampiamente citato nel capitolo precedente, [6], elenca alcune applicazioni e varianti interessanti del metodo di skill recommendation come possibili realizzazioni. Una di queste è quella di suggerire agli utenti l'offerta di lavoro più adatta basandosi sugli skill che possiedono. Partendo da questa idea è stato realizzato un sistema che dato un insieme di skill, riesce a suggerire le offerte di lavoro più affini a quel particolare set. L'insieme di skill può essere visto da due prospettive diverse: il set di skill rappresenta gli skill posseduti da una persona in cerca di lavoro e i risultati della recommendation sono delle offerte di lavoro adatte al particolare profilo della persona; il set di skill rappresenta una posizione aperta in una azienda e i risultati della recommendation sono delle persone con un set di skill adatto a quella particolare position. Di seguito verranno proposti diversi metodi di realizzazione del sistema di recommendation.
Per prima cosa è stata fatta una piccola analisi sulla distribuzione delle position nel campione di training utilizzato per la realizzazione del sistema di recommendation. Nel grafico 6.1 sono riportate le position ordinate per occorrenza e la % di persone che coprono. Le prime 110 position più svolte, il 4.5% di tutte le position, coprono già il 50% del totale delle persone.

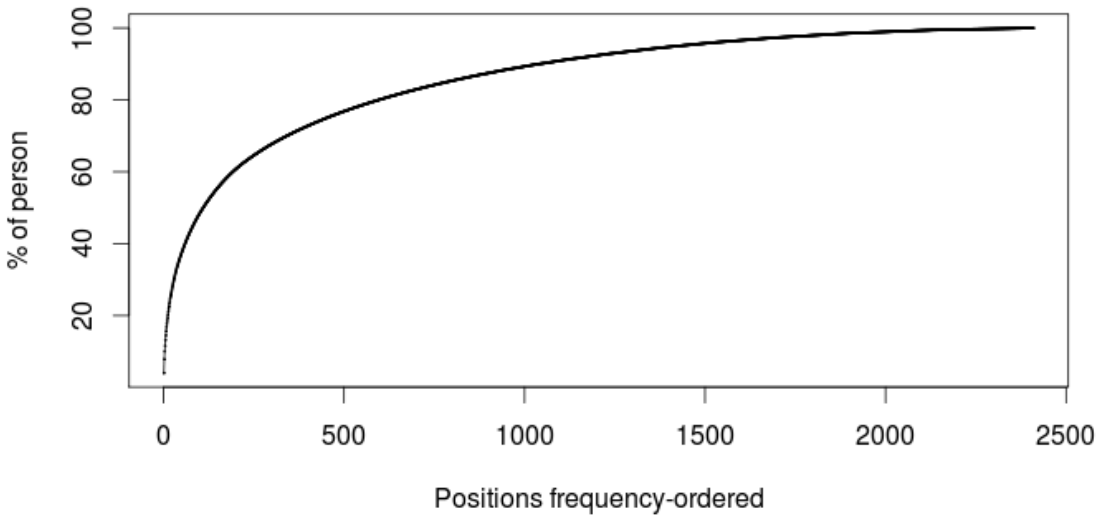

Figura 6.1: Occorrenza delle position in relazione al numero totale di persone

6.1 Position clustering

Prima di formulare un metodo di job recommendation è stato pensato di riproporre un clustering di position come quello realizzato per gli skill.
Inizialmente è stato seguito lo stesso procedimento degli skill ma applicato alle position, in particolare si è partiti dalle co-occorrenze skill-position per poi ottenere da queste le co-occorrenze tra position; una volta ottenute le co-occorrenze tra position si è calcolata la similarità di Jaccard tra position e la conseguente misura di distanza 1-similarità di Jaccard che è stata utilizzata per il clustering. Il risultato è stato un grande cluster contenente quasi tutte le position e pochi altri cluster piccoli contenenti una sola position. Facendo in questo modo probabilmente le co-occorrenze (e di conseguenza la similarità) non riuscivano ad emergere, perciò si è pensato di agire in maniera diversa: sfruttando l'LSA.

6.1.1 Applicazione dell'LSA per il clustering di position

La Latent Semantic Analysis (LSA) è una tecnica molto utilizzata nel text mining. Fu inventata nel 1988 da S. Deerwester, S. Dumais, G. Furnas, R. Harshman, T. Landauer, K. Lochbaum e L. Streeter, [15]. Si parte da una matrice termini-documenti e la si trasforma facendo emergere associazioni semantiche latenti tra termini e documenti. Le associazioni semantiche latenti sono associazioni di ordine superiore non basate su match lessicale tra termini. Si effettua un mapping della matrice in uno spazio di vettori ridotto che approssima quello originale trascurando i dettagli, compreso l'eventuale errore. Nello spazio trasformato, i termini semanticamente simili o associati sono in posizioni limitrofe; la similarità semantica tra termini lessicalmente distinti emerge grazie alla co-occorrenza dei termini in documenti diversi. La trasformazione dello spazio avviene attraverso la Singular Value Decomposition (SVD).
La LSA è stata utilizzata per far emergere le associazioni semantiche latenti tra le position. I documenti sono rappresentati dagli skill mentre i termini sono rappresentati dalle position. La matrice termini-documenti sarà quindi la matrice di co-occorrenze position-skill su cui verrà effettuata la SVD.

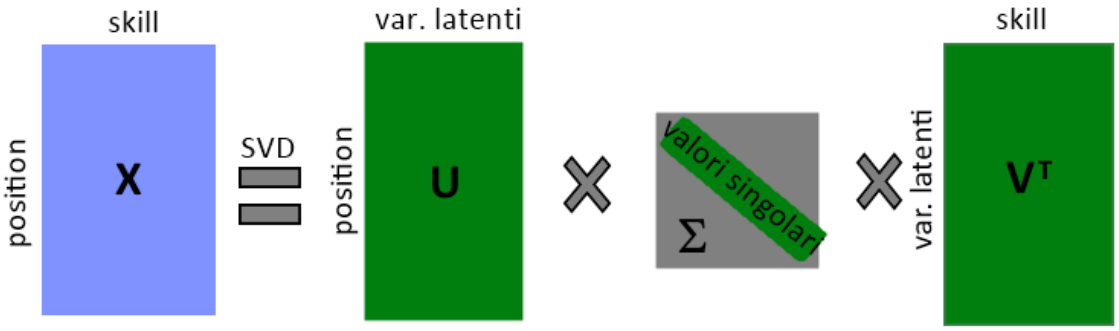

Figura 6.2: SVD applicata alla matrice delle co-occorrenze position-skill

Una volta applicata la SVD si ottengono tre matrici: matrice delle position (U), matrice degli autovalori ($\sum$) e matrice degli skill (V^T). Per ottenere le position nello spazio trasformato è necessario scalare la matrice delle position rispetto agli autovalori nella matrice degli autovalori. Ogni riga della matrice risultante rappresenta una position nel nuovo spazio.
A questo punto è possibile visualizzare le position nello spazio latente tramite per esempio un grafico 2D. In questo caso i termini sono proiettati su due dimensioni, la

seconda e la terza. Nello spazio latente le position sono tanto più simili quanto formano un angolo piccolo con l'origine. I grafici seguenti mostrano proprio la maggior similarità tra gruppi di position che appartengono alla stessa area semantica.

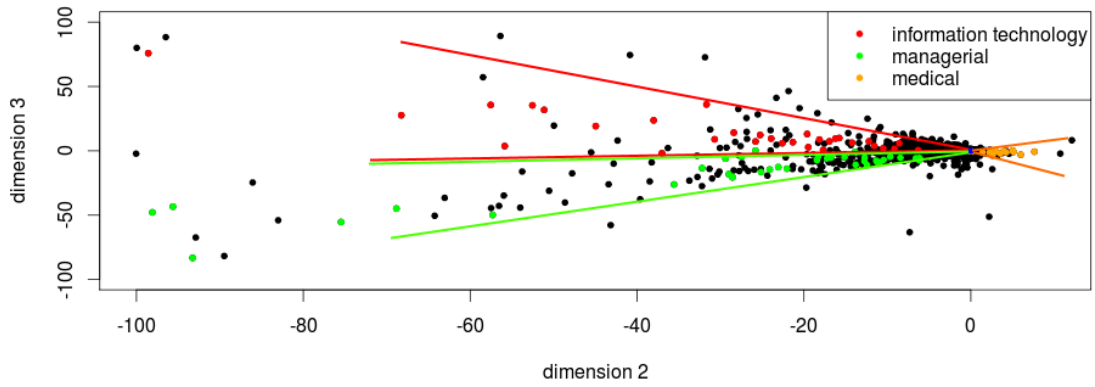

Figura 6.3: Position nello spazio latente proiettato sulle dimensioni 2 e 3

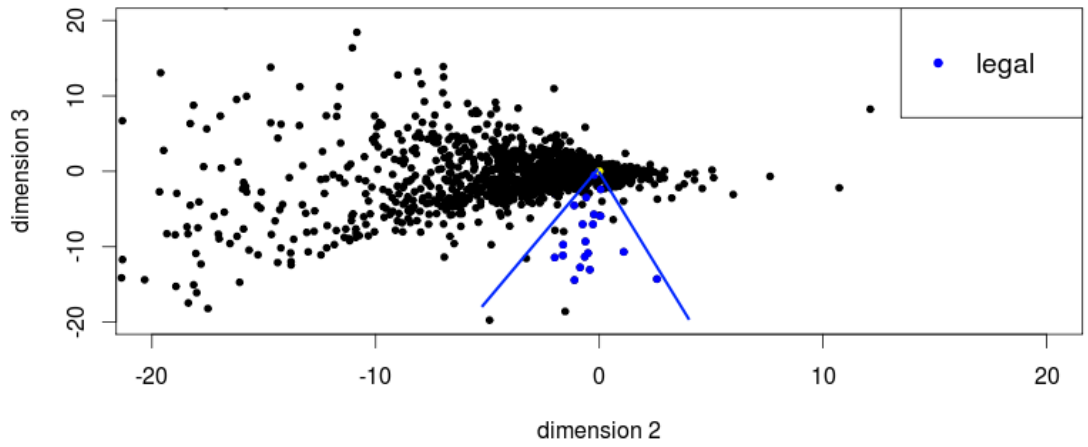

Figura 6.4: Position nello spazio latente proiettato sulle dimensioni 2 e 3

In figura 6.3 sono state visualizzate tutte le 2400 position. Data la numerosità delle position non è stato possibile etichettare ogni punto perché si sarebbe ottenuto un grafico completamente illeggibile. Per evidenziare l'efficacia dell'LSA sono state prese in esame le prime 500 position più frequenti e colorate quelle appartenenti alle tre aree semantiche più presenti, ovvero i settori Information Technology, Managerial e Medical. La colorazione è ovviamente parziale ma già rende l'idea della coerenza: più le position formano con l'origine un angolo piccolo, maggiore è la similarità tra esse. Anche per il settore legale sono state selezionate alcune position e rappresentate nel grafico 6.4. Come si può notare formano tra di loro angoli piccoli rispetto l'origine.

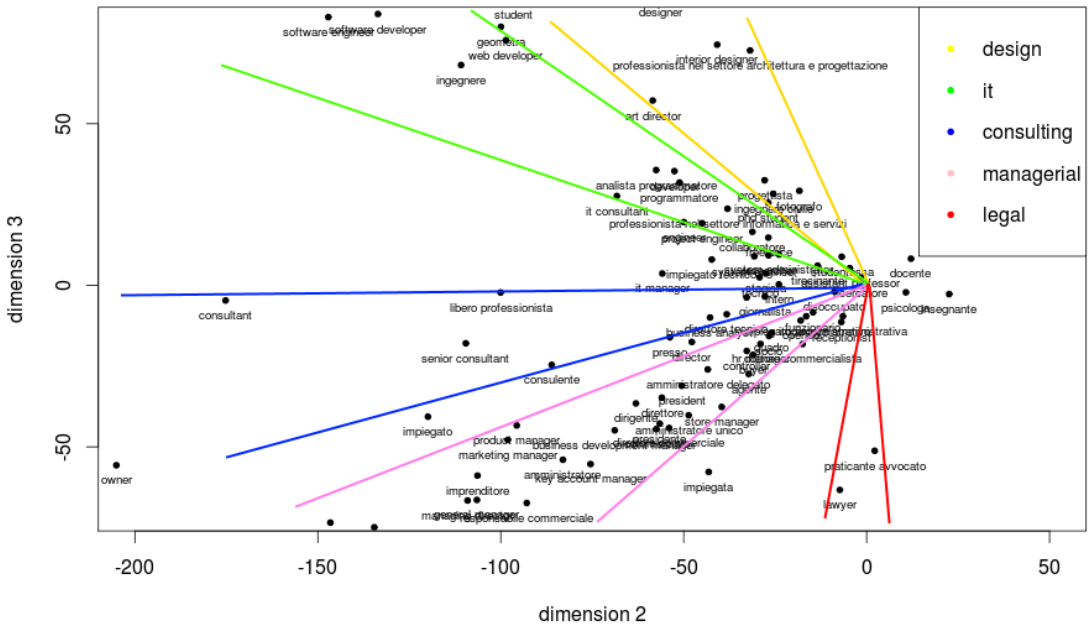

Figura 6.5: 100 position più frequenti nello spazio latente proiettato sulle dimensioni 2 e 3

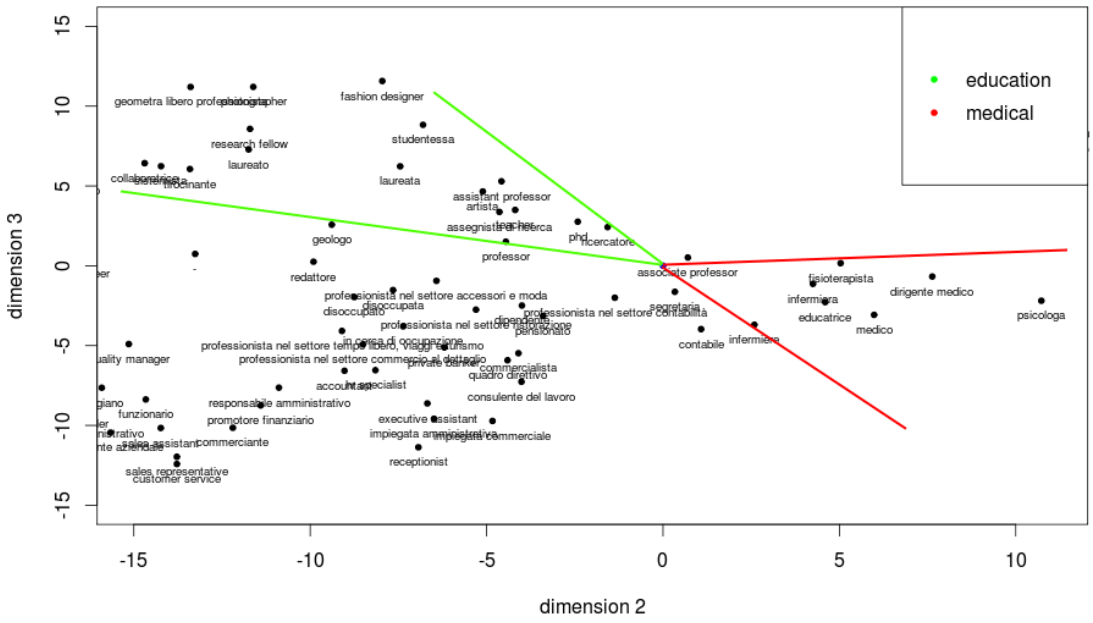

Figura 6.6: 200 position più frequenti nello spazio latente proiettato sulle dimensioni 2 e 3

In figura 6.5 sono state visualizzate soltanto le prime 100 position più frequenti per rendere leggibile il grafico ed etichettate. Sono state circoscritte alcune aree semantiche evidenti quali Design, Information Technology, Consulting, Managerial e Legal. In figura 6.6 sono visualizzate le prime 200 position per mostrare altre aree semantiche che emergono, ovvero Medical e Education.

6.1.2 International Standard Classification of Occupations

International Standard Classification of Occupations (ISCO) è un prodotto dell'International Labour Organization, un'agenzia facente parte delle Nazioni Unite. ISCO è uno strumento per organizzare le posizioni lavorative in un insieme ben definito di gruppi sulla base dei compiti e dei doveri che caratterizzano il lavoro. I suoi obiettivi principali sono di fornire: una base per reporting internazionale, comparazione e scambio di dati amministrativi e statistici sulle occupazioni; un modello

per lo sviluppo di classificazioni nazionali e regionali di occupazioni; un sistema che può essere usato direttamente in paesi in cui non sono state sviluppate le proprie classificazioni nazionali [26]. La prima versione di ISCO risale al 1958 [44], seguita da quelle del 1968, 1988 e l'ultima del 2008 [21].
La classificazione di ISCO è di tipo gerarchico e si suddivide in tre livelli: *Major Groups*, *Sub-major Groups* e *Minor Groups*. I *Major Groups* sono dieci e sono composti da:

1. Managers
2. Professionals
3. Technicians and associate professionals
4. Clerical support workers
5. Service and sales workers
6. Skilled agricultural, forestry and fishery workers
7. Craft and related trades workers
8. Plant and machine operators, and assemblers
9. Elementary occupations
10. Armed forces occupations

Questa classificazione ufficiale può essere utilizzata confrontandola con l'LSA di position realizzata precedentemente. I settori lavorativi visualizzati nei grafici 6.3, 6.5, 6.6, 6.4 possono essere mappati sui gruppi per mostrare che l'LSA effettivamente raggruppa le position in maniera coerente.
Il primo Major Group di ISCO comprende i manager, e le position associate al settore Managerial del grafico 6.3 sono:

manager	account manager
area manager	product manager
general manager	export manager
marketing manager	key account manager
business development manager	hr manager
export area manager	technical manager
senior manager	sales area manager
production manager	area sales manager
quality manager	program manager
brand manager	plant manager
senior project manager	operations manager
service manager	business manager
export sales manager	purchasing manager
event manager	senior account manager
social media manager	branch manager
supply chain manager	sales & marketing manager
assistant manager	country manager
customer service manager	sales account manager
district manager	finance manager
national account manager	office manager
credit manager	

Ovviamente tutte queste position fanno riferimento al Major Group Manager. Nel grafico 6.3 è evidenziato anche il settore dell'Information Technology. Le position corrispondenti sono:

software developer	sistemista
software engineer	senior software engineer
web developer	programmer
analista programmatore	it project manager
it consultant	software architect
project engineer	ingegnere informatico
it manager	analyst programmer
professionista nel settore informatica e servizi	sviluppatore software
programmatore	application engineer
system administrator	webmaster
developer	consulente informatico
system engineer	responsabile it
tecnico informatico	senior developer
web designer	sviluppatore
project leader	senior analyst
it specialist	senior system engineer

Questa categoria di position può essere mappata nel Sub-major group 25 che fa riferimento al Major Group 2 "Professionals", figura 6.7. Il Sub-major group 25 è suddiviso a sua volta in due Minor Groups: "Software and applications developers and analysts" e "Database and network professionals" che raggruppano esattamente le position del settore Information Technology evidenziate nel grafico.

2 Professionals

21 Science and engineering professionals

22 Health professionals

23 Teaching professionals

24 Business and administration professionals

25 Information and communications technology professionals

26 Legal, social and cultural professionals

Figura 6.7: Sub-major groups del Major group 2

Il settore Medical rappresentato nel grafico 6.3 è costituito dalle seguenti position:

dirigente medico	medico
fisioterapista	farmacista
infermiere	infermiera
odontoiatra	tecnico di laboratorio
psicoterapeuta	psicologa psicoterapeuta
informatore scientifico del farmaco	biologo
medico chirurgo	medico veterinario
biologa	professionista nel settore medicina
psicologo psicoterapeuta	

Questo gruppo può essere mappato nel Sub-major group 22 "Health professionals" facente parte del Major Group 2. Esso è suddiviso in diversi Minor Groups (figura 6.8) che racchiudono tutte le position.

22 Health professionals

221 Medical doctors

222 Nursing and midwifery professionals

223 Traditional and complementary medicine professionals

224 Paramedical practitioners

225 Veterinarians

226 Other health professionals

Figura 6.8: Minor groups del Sub-major group 22

L'ultimo settore preso in esame è quello Legale, che in figura 6.4 è rappresentato dalle seguenti position:

senior legal counsel	libero professionista studio legale
legal	avvocato
professionista nel settore servizi legali	praticante avvocato abilitato
professionista nel settore giudiziario	libero professionista avvocato
notaio	avvocato civilista
libero professionista servizi legali	lawyer
consulente legale	avvocato amministrativista
libero professionista giudiziario	avvocato cassazionista
independent legal services professional	avvocato - libero professionista
trainee lawyer	avvocato libero professionista
professionista nel settore studio legale	avvocato penalista
praticante avvocato abilitato al patrocinio	

Il Minor Group 261 "Legal professionals" raggruppa tutte queste position. Esso fa parte del Sub-major Group 26 "Legal, social and cultural professionals", figura 6.9.

26 Legal, social and cultural professionals

261 Legal professionals

262 Librarians, archivists and curators

263 Social and religious professionals

264 Authors, journalists and linguists

265 Creative and performing artists

Figura 6.9: Minor groups del Sub-major group 26

La mappatura dei settori evidenziati nei grafici dell'LSA con la classificazione ISCO-08 fornisce un'ulteriore prova della bontà dell'applicazione dell'LSA alla matrice delle co-occorrenze position-skill.
Le position nello spazio trasformato sono ora comparabili, perciò è stata calcolata la similarità coseno tra tutte le position. Questa misura, come la similarità di Jaccard nel caso degli skill, è la base su sui si è calcolata la distanza tra le position che serve all'algoritmo di clustering.

6.1.3 Clustering

Il tipo di clustering utilizzato per le position è lo stesso clustering gerarchico applicato agli skill. La misura di distanza utilizzata è stata ricavata dalla similarità coseno ottenuta dall'applicazione dell'LSA sulle co-occorrenze position-skill descritta sopra. Anche in questo caso sono stati ispezionati diversi livelli di taglio del dendrogramma e si è visto che le position vengono raggruppate in modo abbastanza coerente, molto meglio del caso iniziale in cui si utilizzava lo stesso metodo di calcolo delle distanze del clustering di skill. Di seguito alcuni esempi di cluster.

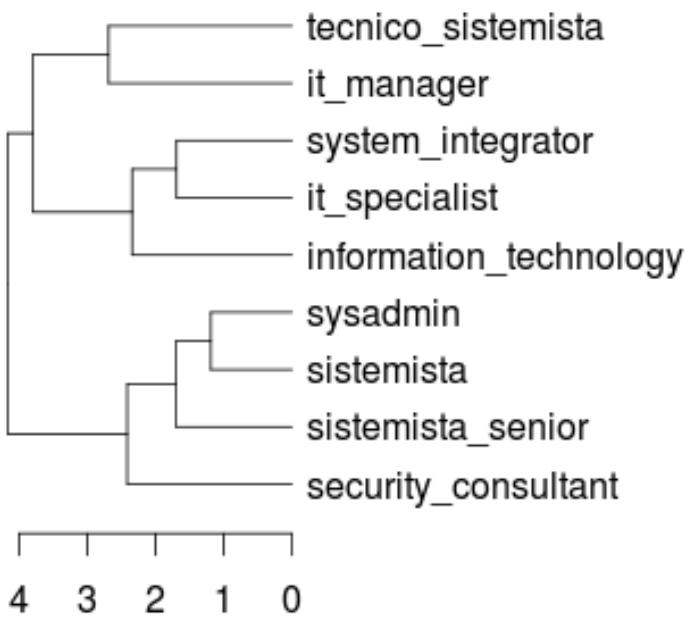

Figura 6.10: Cluster di position in ambito sistemistico

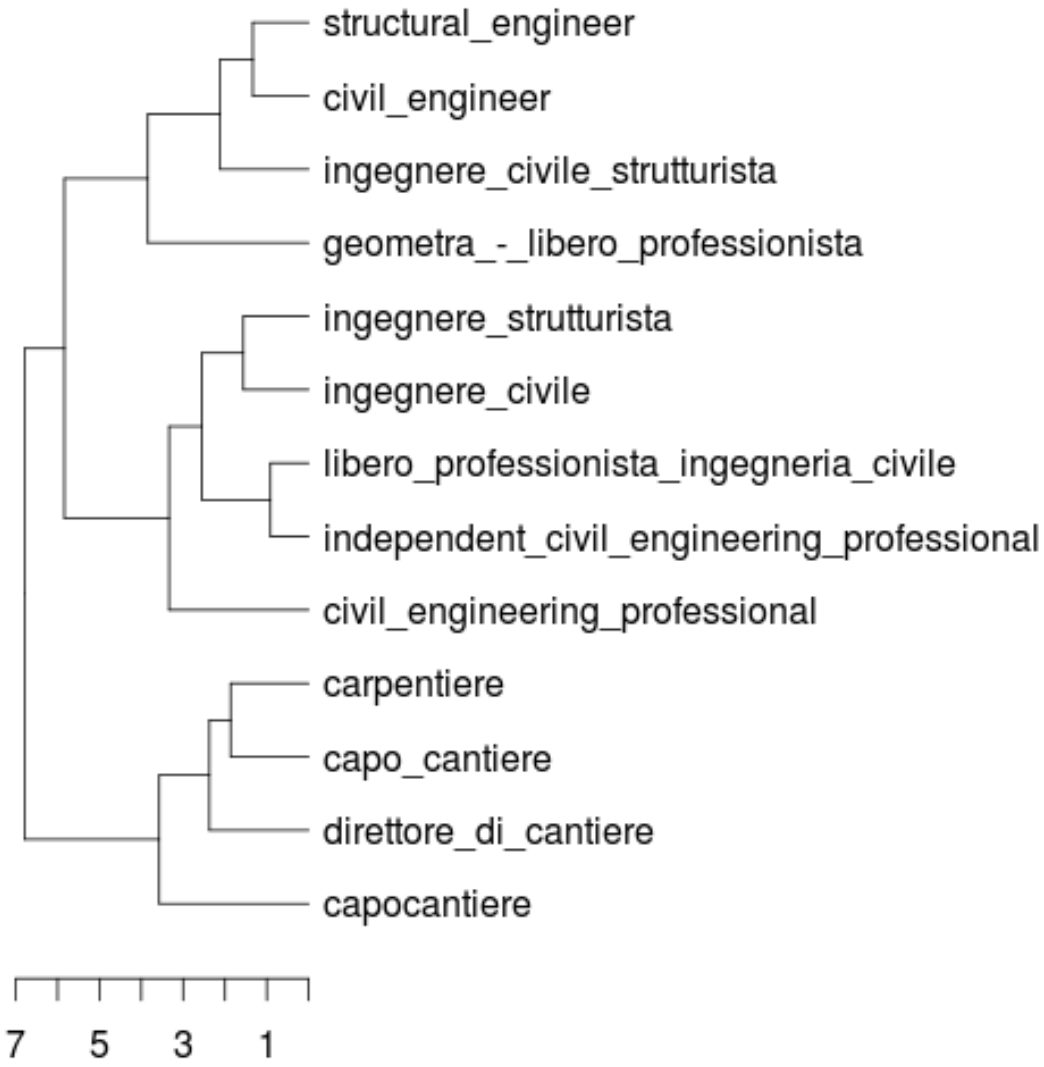

Figura 6.11: Cluster di position in ambito edile

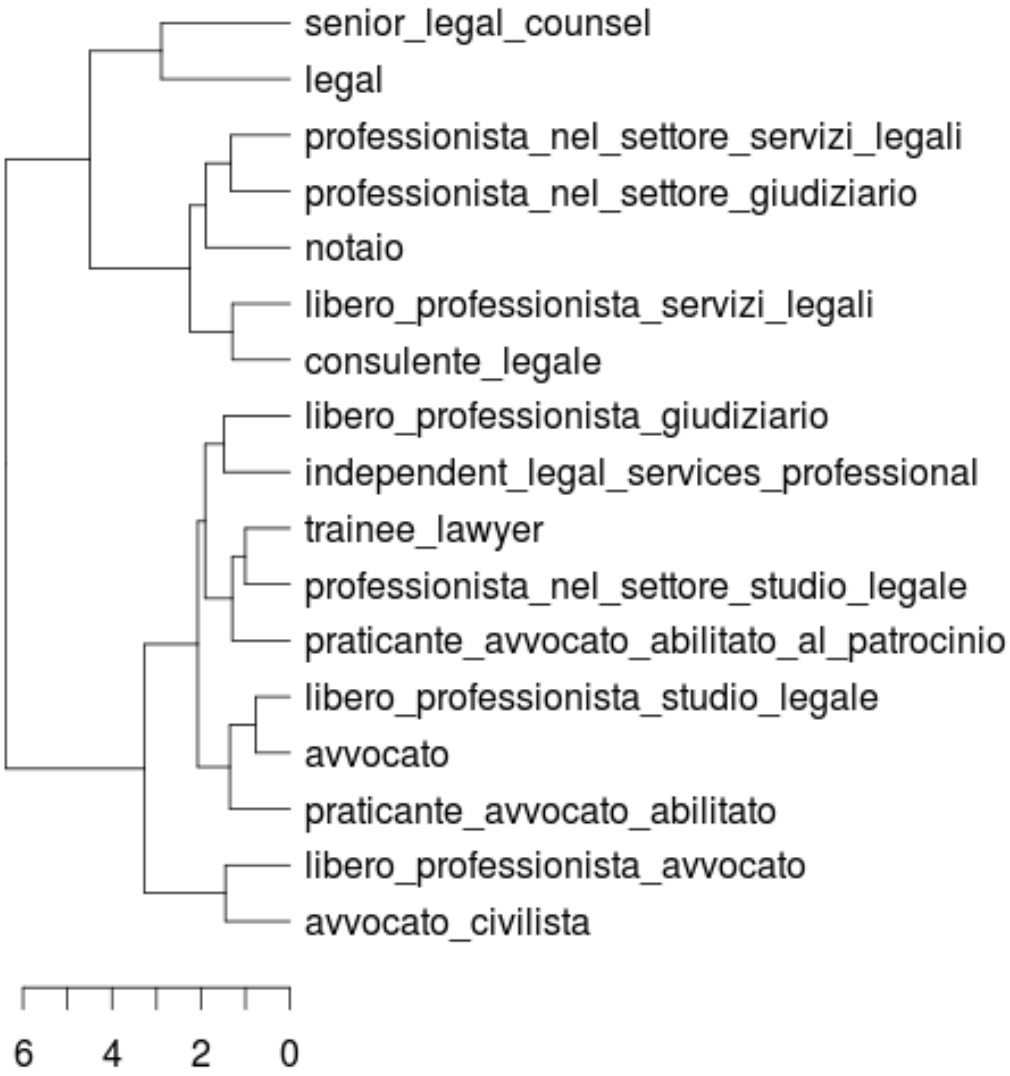

Figura 6.12: Cluster di position in ambito legale

6.2 Primo metodo di job recommendation

Il metodo proposto ha l'obiettivo di determinare l'affinità tra un insieme di skill e un insieme di position, il quale a sua volta può essere visto come un insieme di skill. La generalità del metodo fa sì che esso possa essere utile sia per chi cerca lavoro, sia per le aziende che ricercano personale.

6.2.1 Metodo

Il primo passo è quello di estrapolare le informazioni riguardanti gli skill e le position degli utenti, opportunamente filtrate in base alla frequenza per ottenere un dataset di dimensione appropriata. La soglia minima di occorrenza sia di skill che di position è stata fissata a 20.
Successivamente è stata calcolata la similarità di Jaccard tra ogni coppia skill-position tramite la stessa formula descritta in 5.1.1. Questo è il modello su cui si basa l'intero metodo.
Formalmente si definisce una matrice persone-skill T in cui $t_{i,j} = 1$ se la persona i possiede lo skill j, 0 altrimenti. Una persona ha un insieme di skill, dove ogni skill è una riga della matrice di Jaccard skill-position R. Per rappresentare una persona i si costruisce un vettore p_i che è la media di tutti i vettori di skill che la persona possiede, dato dal prodotto matriciale tra vettore t^i e R diviso il numero totale di skill posseduti.

$$p_i = \frac{1}{\sum_j t_{i,j}} t^i R$$

In figura 6.13 una visualizzazione grafica.

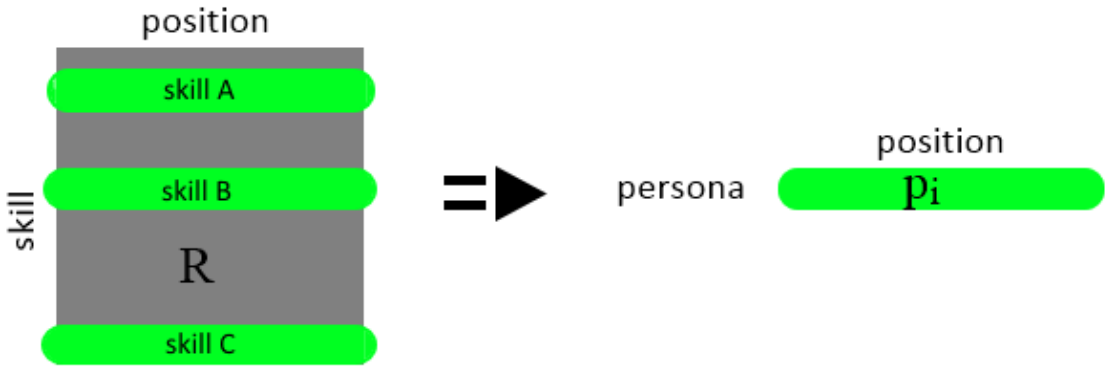

Figura 6.13: Costruzione del vettore che rappresenta una persona

Ogni elemento del vettore persona p_i è la similarità che ha quella persona con una precisa position. Per determinare le position più affini alla persona è sufficiente prendere quelle con similarità maggiore, ordinando i valori del vettore persona. Questa visione del metodo è pensata per chi cerca lavoro e ha un insieme di offerte di lavoro da prendere in considerazione; il sistema lo aiuterà a mettere il focus sulle offerte più affini al suo particolare profilo.
La visione del metodo dalla prospettiva aziendale è la seguente: l'azienda ha una determinata position aperta e un insieme di candidature tra cui deve scegliere. La position è una colonna della matrice di Jaccard skill-position ed ogni candidato è rappresentato da un vettore persona costruito come descritto sopra. In questo modo è possibile ordinare i candidati in base al valore del vettore persona in corrispondeza della position ricercata. Altro approccio possibile è quello di rappresentare la position aperta come un insieme di skill e costruire il relativo vettore tramite la matrice di Jaccard skill-position. A questo punto è sufficiente calcolare la similarità tra il vettore che rappresenta la position e i vettori dei candidati.
Il metodo in definitiva offre diverse soluzioni sia per chi cerca lavoro sia per le aziende che ricercano una particolare position o ricercano una figura con determinati skill. Tutti i dettagli sull'implementazione del metodo vengono rimandati al capitolo 7.

6.2.2 Risultati

Metodo

Il metodo presentato ha come obiettivo generale quello di far incontrare persone ed offerte di lavoro. Per misurare l'efficacia del metodo bisognerebbe possedere delle informazioni su persone che nella realtà hanno risposto a delle offerte di lavoro con esito positivo; queste informazioni non sono disponibili perciò è necessario formulare un altro tipo di misurazione. Si è pensato di confrontare le position suggerite dal metodo con la position che effettivamente ricopre quella specifica persona perché molto probabilmente gli skill che possiede sono in affinità con il lavoro che svolge.
Il dataset complessivo è stato suddiviso in training set e test set, costituiti rispettivamente di 51 mila e 37 mila profili. Le position distinte sono circa 2400. La misura calcolata è la recall@k, dove k è il numero di position suggerite per persona. La recall è calcolata come numero di volte che nelle k position suggerite compare quella vera diviso per il numero di persone totali di test. Nella tabella 6.1 sono riportati i risultati numerici, mentre nel grafico 6.14 viene visualizzato l'andamento della recall con l'aumentare di k. Come si può notare i primi k sono quelli che fanno aumentare più velocemente la recall: infatti con k = 20, ovvero 20 position suggerite, si arriva già ad una recall di 0.5, che è circa il 77% della massima recall raggiunta.

k	Recall	k	Recall	k	Recall	k	Recall	k	Recall
1	0.1264	11	0.4130	21	0.5103	31	0.5711	41	0.6149
2	0.1912	12	0.4252	22	0.5168	32	0.5757	42	0.6183
3	0.2338	13	0.4378	23	0.5242	33	0.5803	43	0.6215
4	0.2703	14	0.4484	24	0.5310	34	0.5847	44	0.6253
5	0.3008	15	0.4583	25	0.5379	35	0.5894	45	0.6291
6	0.3255	16	0.4680	26	0.5436	36	0.5937	46	0.6321
7	0.3463	17	0.4779	27	0.5496	37	0.5983	47	0.6355
8	0.3662	18	0.4871	28	0.5552	38	0.6028	48	0.6389
9	0.3838	19	0.4951	29	0.5606	39	0.6073	49	0.6419
10	0.3994	20	0.5028	30	0.5658	40	0.6112	50	0.6449

Tabella 6.1: Recall@k del metodo di job recommendation

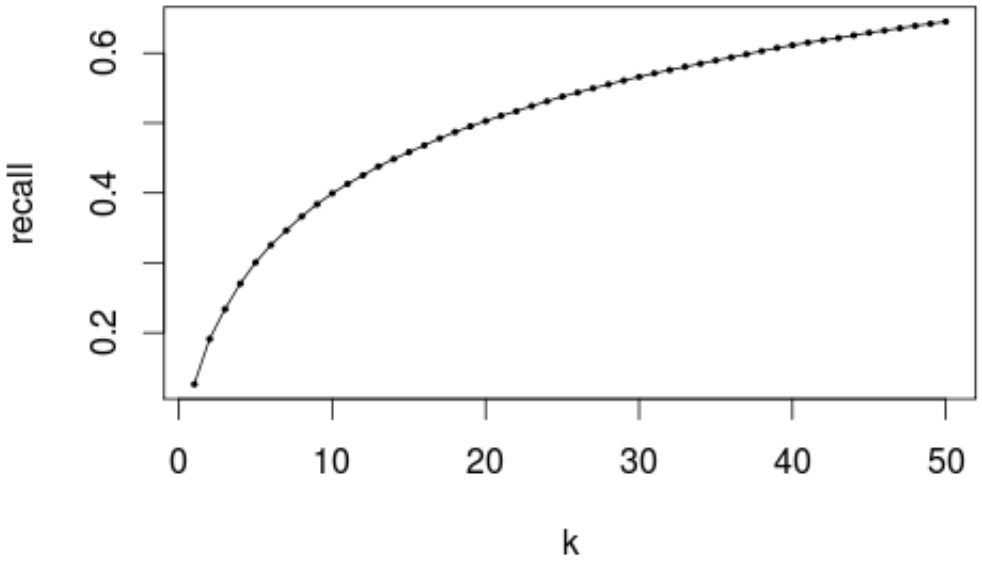

Figura 6.14: Recall in funzione di k

Classificatore basato sulla frequenza

L'efficacia del metodo di job recommendation può essere dimostrata, oltre che dalla recall, comparandolo con classificatori "dummy". Come nella recommendation di skill, 5.2.3, è stato realizzato un classificatore basato sulla frequenza delle position, che quindi suggerirà sempre le prime k position più frequenti. Nella tabella 6.2 i risultati.

k	Recall	k	Recall	k	Recall	k	Recall	k	Recall
1	0.04352	11	0.20786	21	0.28371	31	0.33364	41	0.37153
2	0.08684	12	0.21791	22	0.28871	32	0.33775	42	0.37485
3	0.11095	13	0.22679	23	0.29400	33	0.34191	43	0.37847
4	0.12729	14	0.23622	24	0.29921	34	0.34604	44	0.38122
5	0.14181	15	0.24513	25	0.30432	35	0.35009	45	0.38419
6	0.15618	16	0.25350	26	0.30926	36	0.35433	46	0.38727
7	0.16995	17	0.26044	27	0.31469	37	0.35795	47	0.39059
8	0.18113	18	0.26714	28	0.32011	38	0.36102	48	0.39291
9	0.19034	19	0.27299	29	0.32495	39	0.36453	49	0.39564
10	0.1993	20	0.27853	30	0.32984	40	0.36856	50	0.39875

Tabella 6.2: Recall@k classificatore basato sulla frequenza

Classificatore casuale basato sulla frequenza

Anche il classificatore casuale basato sulla frequenza implementato per la recommendation di skill, 5.2.3, è stato applicato per la recommendation di position. Nella tabella 6.3 i risultati.

k	Recall	k	Recall	k	Recall	k	Recall	k	Recall
1	0.00675	11	0.06518	21	0.11171	31	0.14970	41	0.18102
2	0.01231	12	0.07050	22	0.11632	32	0.15316	42	0.18383
3	0.11095	13	0.07471	23	0.12062	33	0.15683	43	0.18634
4	0.02527	14	0.07946	24	0.12445	34	0.15972	44	0.18899
5	0.03097	15	0.08416	25	0.12777	35	0.16277	45	0.19188
6	0.03683	16	0.08873	26	0.13155	36	0.16596	46	0.19461
7	0.04279	17	0.09337	27	0.13533	37	0.17001	47	0.19720
8	0.04833	18	0.09815	28	0.13906	38	0.17254	48	0.19995
9	0.05381	19	0.10274	29	0.14225	39	0.17522	49	0.20249
10	0.05956	20	0.10730	30	0.14592	40	0.17794	50	0.20465

Tabella 6.3: Recall@k classificatore casuale basato sulla frequenza

6.3 Secondo metodo di job recommendation basato sul clustering gerarchico

Dopo aver implementato un primo metodo di job recommendation ed ottenuto buoni risultati, si è pensato ad una evoluzione del sistema che sfrutti il clustering gerarchico realizzato in precedenza.

6.3.1 Metodo

Il metodo presentato si avvale della gerarchia di position realizzata precedentemente, e può essere considerato una classificazione gerarchica. La classificazione gerarchica, e differenza di quella classica, non considera completamente incorretta una classificazione di una istanza in una classe che è vicina o simile alla classe vera, ma attraverso misure di similarità o distanza calcola il grado di correttezza della classificazione. Il metodo si ispira proprio alla classificazione gerarchica in cui le istanze da classificare sono le persone e le classi sono le position. L'implementazione del metodo verrà discussa nel capitolo 7.
Il metodo è caratterizzato da diverse fasi:

1. *Vettore persona.* La costruzione del vettore persona è simile al primo metodo ma con qualche differenza: dato il set di skill che caratterizza la persona si calcola la media delle righe della matrice di Jaccard skill-skill J (e non skill-position) e si ottiene il vettore p_i (figura 6.15)

2. *Calcolo centroidi cluster.* Si parte dal nodo radice dell'albero ottenuto dal clustering gerarchico, ogni nodo avrà esattamente due figli. Si calcola il centroide dei due cluster figli in questo modo: ogni cluster è costituito da un insieme di position, per ogni position si prende la colonna corrispondente della matrice di Jaccard skill-position R e il centroide è la media di tutte le position (figure 6.16 e 6.17)

3. *Distanza euclidea.* Si calcola la distanza euclidea tra i due centroidi (rappresentati da un vettore di skill) e il vettore persona p_i. Il nodo del cluster con distanza minore è quello scelto e si scende al livello successivo

4. *Confronto con position vera.* La vera position della persona sarà contenuta in uno dei due possibili cluster. Se si sceglie il nodo giusto lo step è considerato corretto e si prosegue navigando l'albero (fase 2) finché o si arriva al nodo

foglia che è esattamente la vera position della persona oppure si sbaglia scelta. Se si sceglie il nodo sbagliato, che non contiene la vera position, l'algoritmo si ferma

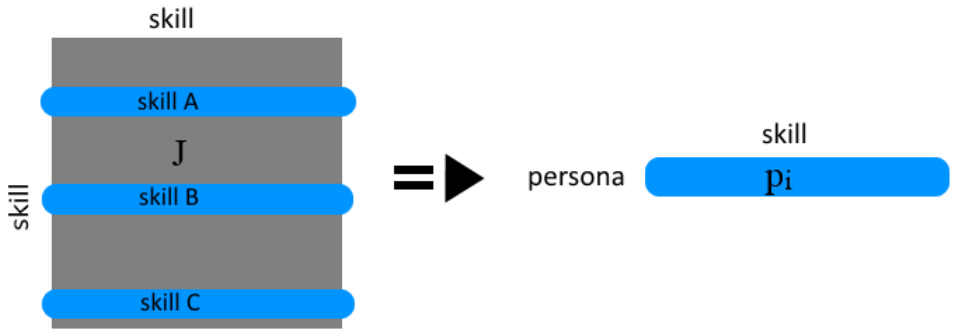

Figura 6.15: Costruzione vettore persona

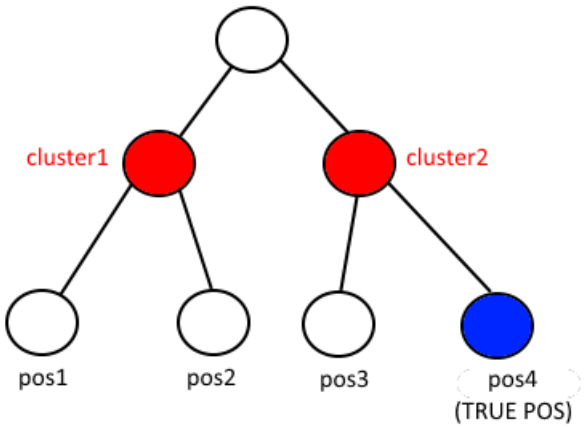

Figura 6.16: Gerarchia di position

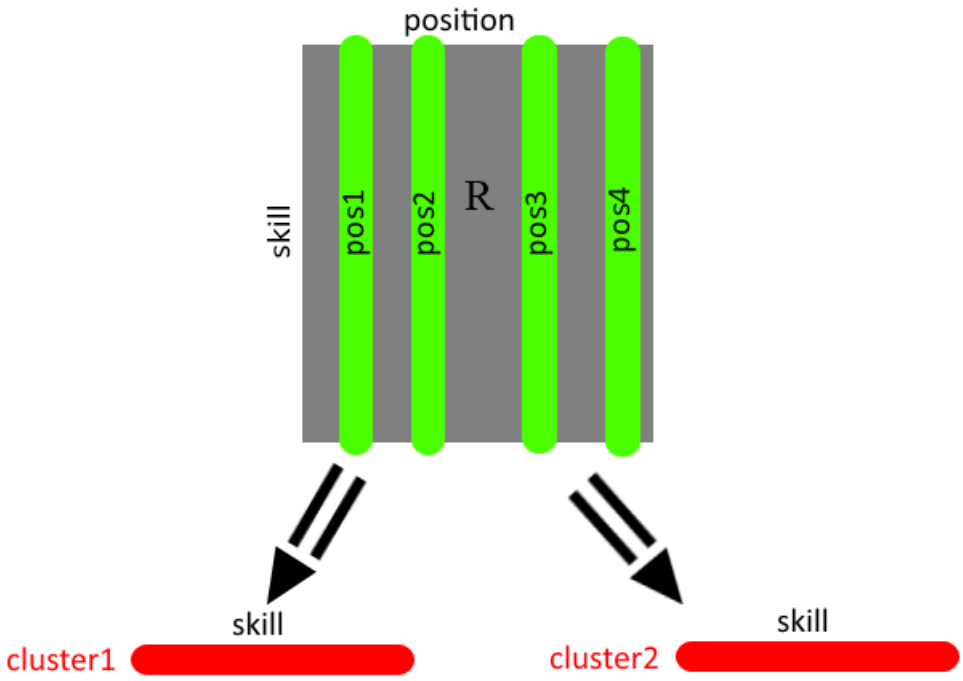

Figura 6.17: Calcolo dei centroidi dei cluster

6.3.2 Risultati

La misurazione dell'accuratezza del metodo, come nella classificazione gerarchica, avviene calcolando il grado di correttezza. In particolare la recall viene calcolata come numero di passi corretti diviso per il numero di passi totali per arrivare alla vera position. La divisione dei dati in training e test è esattamente la stessa di quella realizzata per il primo metodo.

Metodo

La recall media massima si attesta a 0.248. Per fare un confronto con il primo metodo bisogna prendere in considerazione la recall@1, perché nel secondo metodo, seppur in diversi step, si arriva a suggerire una sola position. La recall@1 del primo metodo si ferma a 0.126, perciò si registra un notevole miglioramento rispetto alla soluzione precedente.

Ricalcolo della recall del primo metodo

Dato il miglioramento dei risultati sfruttando la gerarchia di position, è stato pensato di ricalcolare l'accuratezza del primo metodo di recommendation utilizzando la stessa strategia utilizzata per il secondo. Ad esempio, prendendo k=5, sono state prese le prime 5 position suggerite per ogni persona di test e per ogni position suggerita è stato navigato l'albero calcolando quanti step in comune avevano la position suggerita e la position vera. Sulle 5 position suggerite si è presa la recall maggiore e così via per k da 1 a 50. In figura 6.18 un esempio grafico dove lo step in comune tra position suggerita e position vera è 1, mentre gli step totali per arrivare alla position vera sono 3. La recall in questo caso è 1/3=0.33.

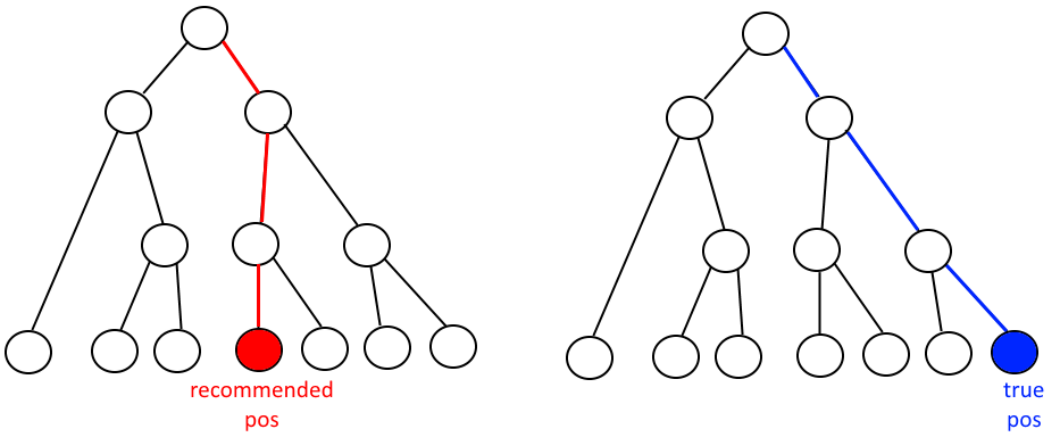

Figura 6.18: Esempio di navigazione dell'albero e confronto tra position suggerita e position vera

Nella tabella 6.4 sono riportate le recall medie massime per k da 1 a 50. Confrontando questi risultati con quelli della recall binaria del primo metodo (6.1) è evidente l'ampio miglioramento, dovuto al fatto che la classificazione di una position vicina a quella vera è considerata comunque buona, anche se non ottima. Nel grafico 6.19 l'andamento della recall con l'aumentare di k. In questo caso il grafico è ancora più a curva rispetto a quello della recall binaria (6.14), infatti con k=20 si raggiunge già l'88.5% della recall massima ottenuta, contro il 77%.

k	Recall	k	Recall	k	Recall	k	Recall	k	Recall
1	0.27921	11	0.67919	21	0.76944	31	0.81517	41	0.84412
2	0.39244	12	0.69183	22	0.77536	32	0.81898	42	0.84620
3	0.45923	13	0.70398	23	0.78070	33	0.82221	43	0.84841
4	0.51100	14	0.71486	24	0.78578	34	0.82502	44	0.85081
5	0.54970	15	0.72424	25	0.79065	35	0.82795	45	0.85296
6	0.57995	16	0.73325	26	0.79532	36	0.83101	46	0.85513
7	0.60512	17	0.74209	27	0.79973	37	0.83384	47	0.85726
8	0.62826	18	0.74974	28	0.80380	38	0.83651	48	0.85911
9	0.64790	19	0.75694	29	0.80780	39	0.83916	49	0.86111
10	0.66480	20	0.76328	30	0.81171	40	0.84170	50	0.86282

Tabella 6.4: Recall@k del primo metodo ricalcolata utilizzando la gerarchia

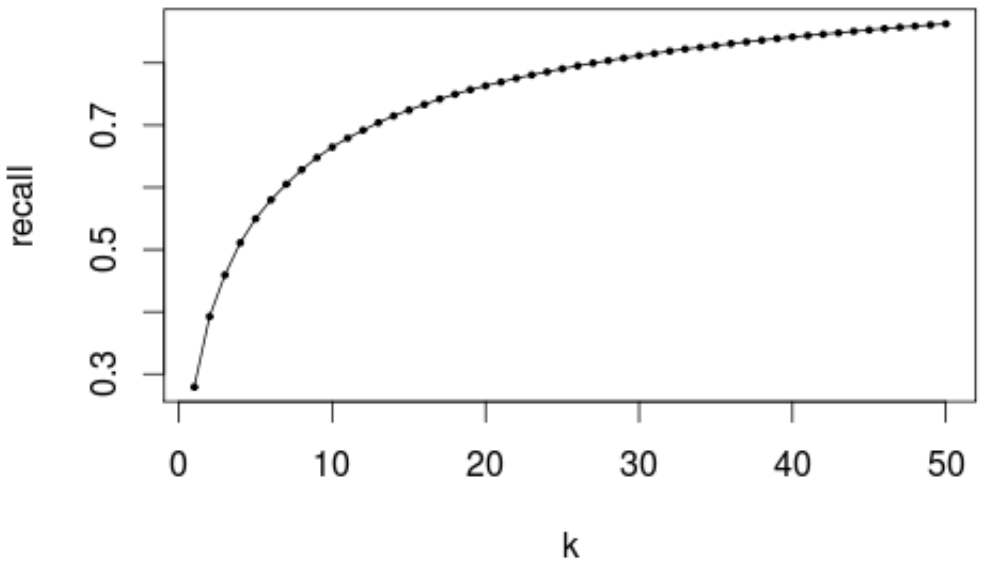

Figura 6.19: Recall in funzione di k

6.4 Considerazioni

Il lavoro svolto in questo capitolo consiste nello sviluppo di un sistema che possa affiancare le aziende nella selezione di candidati e le persone nella ricerca di lavoro. L'obiettivo finale è quello di dimostrare l'effettiva utilità del sistema e la sua versatilità: nonostante la mancanza di dati riguardanti offerte di lavoro e persone che rispondendo all'offerta sono state selezionate, è stato mostrato che il sistema, in entrambe le versioni, è riuscito ad acquisire conoscenza semantica su skill e position. Il problema è stato affrontato sotto diversi aspetti, in particolare:

- *Skill recommendation.* Seguendo il lavoro dei ricercatori LinkedIn, [6], è stato dimostrato nella sezione 5.2 che il sistema è in grado di suggerire skill agli utenti sulla base di informazioni di profilo, e quindi ha acquisito conoscenza semantica sugli skill e le relazioni tra essi

- *Position clustering.* Prendendo ispirazione dal lavoro sugli skill, nella sezione 6.1 è stata applicata la LSA e poi realizzato un clustering di position mostrando come posizioni semanticamente simili vengano associate, prova del fatto che il sistema ha acquisito conoscenza sulle position e relazioni tra esse

- *Position recommendation.* I metodi sviluppati nelle sezioni 6.2 e 6.3 riescono ad associare insiemi di skill con position, insiemi di skill con altri insiemi di skill e position tra loro, dando prova della loro grande versatilità

Tutta la conoscenza acquisita dal sistema offre un modello generale, che può esser sfruttato in diverse maniere sia dalle persone che cercano lavoro che dalle aziende che lo offrono, aprendo la strada ad un grande numero di possibili applicazioni.

6.5 Classificazione supervisionata

I metodi di job recommendation presentati sopra non hanno termini di confronto come quello di recommendation di skill in cui il paper forniva risultati confrontabili, perciò si è pensato di effettuare delle classificazioni utilizzando algoritmi supervisionati. La classificazione supervisionata parte da una collezione di record (training set), in cui ogni record è composto da un insieme di attributi, di cui uno esprime la classe di appartenenza del record. Essa prevede di trovare un modello per l'attributo di classe che esprima il valore dell'attributo in funzione dei valori degli altri attributi. L'obiettivo è quello di assegnare una classe ai record del test set per determinare

l'accuratezza. Applicando questi concetti al problema della job recommendation i record sono costituiti dalle persone, mentre gli attributi dei record sono l'insieme di tutti gli skill possibili più l'attributo classe. Gli attributi skill sono di tipo binario, ovvero se la persona possiede quello skill l'attributo avrà valore 1, altrimenti avrà valore 0. L'attributo classe è di tipo nominale e rappresenta la position della persona.
La suddivisione in training e test set è la medesima utilizzata per i metodi precedenti. Sono stati addestrati diversi tipi di algoritmi e i risultati possono essere visionati nelle tabelle seguenti, in particolare: k-nearest neighbor con k=100 (6.5), k-nearest neighbor con k=50 (6.6), RandomForest (6.7), Support Vector Machine (SVM) con kernel lineare (6.8).

k	Recall	k	Recall	k	Recall	k	Recall	k	Recall
1	0.097	11	0.323	21	0.389	31	0.433	41	0.463
2	0.158	12	0.332	22	0.394	32	0.436	42	0.465
3	0.190	13	0.342	23	0.398	33	0.439	43	0.468
4	0.218	14	0.350	24	0.403	34	0.442	44	0.470
5	0.243	15	0.356	25	0.408	35	0.445	45	0.472
6	0.260	16	0.362	26	0.413	36	0.448	46	0.475
7	0.274	17	0.368	27	0.418	37	0.451	47	0.477
8	0.289	18	0.373	28	0.422	38	0.454	48	0.479
9	0.302	19	0.379	29	0.425	39	0.458	49	0.481
10	0.314	20	0.384	30	0.429	40	0.461	50	0.483

Tabella 6.5: Recall@k utilizzando k-nearest neighbor con k=100

k	Recall	k	Recall	k	Recall	k	Recall	k	Recall
1	0.112	11	0.330	21	0.388	31	0.431	41	0.462
2	0.172	12	0.338	22	0.393	32	0.434	42	0.464
3	0.206	13	0.346	23	0.397	33	0.437	43	0.466
4	0.234	14	0.353	24	0.402	34	0.440	44	0.469
5	0.255	15	0.358	25	0.406	35	0.444	45	0.471
6	0.272	16	0.364	26	0.411	36	0.447	46	0.473
7	0.285	17	0.368	27	0.415	37	0.450	47	0.475
8	0.299	18	0.373	28	0.419	38	0.453	48	0.476
9	0.310	19	0.378	29	0.423	39	0.456	49	0.478
10	0.322	20	0.383	30	0.427	40	0.459	50	0.480

Tabella 6.6: Recall@k utilizzando k-nearest neighbor con k=50

k	Recall	k	Recall	k	Recall	k	Recall	k	Recall
1	0.092	11	0.250	21	0.258	31	0.262	41	0.264
2	0.130	12	0.251	22	0.258	32	0.262	42	0.265
3	0.158	13	0.253	23	0.259	33	0.262	43	0.265
4	0.178	14	0.253	24	0.259	34	0.263	44	0.265
5	0.195	15	0.254	25	0.260	35	0.263	45	0.265
6	0.210	16	0.255	26	0.260	36	0.263	46	0.266
7	0.224	17	0.255	27	0.261	37	0.263	47	0.266
8	0.235	18	0.256	28	0.261	38	0.264	48	0.266
9	0.243	19	0.256	29	0.261	39	0.264	49	0.267
10	0.248	20	0.257	30	0.261	40	0.264	50	0.267

Tabella 6.7: Recall@k utilizzando l'algoritmo RandomForest

k	Recall	k	Recall	k	Recall	k	Recall	k	Recall
1	0.115	11	0.370	21	0.455	31	0.512	41	0.550
2	0.180	12	0.380	22	0.462	32	0.516	42	0.554
3	0.218	13	0.391	23	0.468	33	0.521	43	0.557
4	0.250	14	0.400	24	0.474	34	0.524	44	0.560
5	0.275	15	0.410	25	0.480	35	0.529	45	0.563
6	0.296	16	0.418	26	0.486	36	0.532	46	0.566
7	0.315	17	0.427	27	0.491	37	0.536	47	0.569
8	0.330	18	0.434	28	0.496	38	0.540	48	0.572
9	0.345	19	0.442	29	0.501	39	0.540	49	0.575
10	0.358	20	0.448	30	0.507	40	0.546	50	0.578

Tabella 6.8: Recall@k utilizzando SVM (linear kernel)

Tutti e cinque gli algoritmi proposti devono essere confrontati con la recall@k binaria del primo metodo, perché gli algoritmi di classificazione non misurano il grado di accuratezza o di similarità della position suggerita rispetto alla position reale, riportano soltanto se la position suggerita è esattamente quella vera. In nessun caso gli algoritmi raggiungono il metodo messo a punto; l'unico algoritmo degno di nota è SVM che per i primi k riesce a tenere testa al metodo, mentre aumentando il numero di position suggerite il divario si allarga.

Capitolo 7

Architettura e implementazione

In questo capitolo verranno presentate le elaborazioni effettuate in tutto il lavoro dal punto di vista implementativo, ovvero come sono stati realizzati in pratica le applicazioni e i sistemi discussi nei capitoli precedenti, descrivendo anche gli strumenti utilizzati, quali tool e ambienti di sviluppo.

7.1 Tecnologie utilizzate

Eclipse

Eclipse è un IDE (Integrated Development Environment) incentrato sulla piattaforma con vari tool di sviluppo integrati. Ogni nuovo ambiente di sviluppo può essere scritto come plug-in ed incorporato nell piattaforma Eclipse. Quando il modulo è inserito, Eclipse avrà la capacità di eseguire nuovi task [10].
Eclipse è un IDE open source ed estensibile sostenuto da IBM ed altre aziende quali HP, Intel, SAP, Borland e Oracle. È diventato velocemente una grande community open-source attorno alla quale persone da tutto il mondo sviluppano diversi progetti. La versione Kepler di Eclipse, quella precedente all'attuale Luna, conta più di 5 milioni e 700 mila download.
L'IDE Java di Eclipse rende lo sviluppo Java più efficiente e produttivo. Tra le funzionalità più utili, offre l'evidenziazione della sintassi per le parole chiave, autocompletamento di codice per terminare nomi di variabili lunghi e nomi di metodi definiti precedentemente, assistenza del codice che fornisce suggerimenti per i metodi, import dei package necessari, ricerca di dichiarazioni e utilizzi di codice referenziato, compilazione real time che fornisce indicazioni per trovare errori o errori potenziali

come variabili non dichiarate, parentesi non corrette, parametri di input o valori di ritorno non validi [10].
Eclipse è stato utilizzato come ambiente di sviluppo per la realizzazione delle applicazioni Java riguardanti: crawler per l'ottenimento di profili pubblici di utenti LinkedIn, estrazione delle informazioni sui profili in oggetti Java, mapping tra modello dei dati ad oggetti e modello relazionale.

PostgreSQL

Le origini di PostgreSQL si rifanno ad un progetto realizzato dall'università della California nel 1970, un database relazionale di nome Ingres. Dopo l'implementazione di Ingres in un prodotto commerciale, cambi di proprietà ed aggiunta di feature, nel 1996 nasce PostgreSQL. Oggi PostgreSQL è sviluppato da un gruppo internazionale chiamato Global Development Group. PostgreSQL è un prodotto open-source; Red Hat ha commercializzato PostgreSQL creando il Red Hat Database, ma PostgreSQL rimane free [14].
PostgreSQL è uno dei più avanzati database disponibili. Dispone di un grande numero di feature, tra cui: elaborazione delle transazioni, PostreSQL protegge i dati e coordina la presenza di multipli utenti concorrenti; integrità referenziale, PostgreSQL implementa la completa integrità referenziale supportando le relazioni tra chiavi primarie ed esterne così come i trigger; linguaggi procedurali multipli, i trigger e altre procedure possono essere scritte in diversi linguaggi procedurali; API client, PostgreSQL supporta lo sviluppo di applicazioni client in numerosi linguaggi attraverso delle interfacce; tipi di dato unici, PostgreSQL offre una varietà di tipi di dato oltre a quelli usuali, come i tipi geometrici o i booleani; estensibilità, se non si trova un tipo di dato, una funzione o un operatore specifico lo si può aggiungere autonomamente. Ci sono anche numerosi package disponibili su Internet [14].
PostgreSQL è stato utilizzato come unico database per i profili degli utenti LinkedIn.

R

R è un sistema per il calcolo statistico e la grafica. Fornisce, tra le altre cose, un linguaggio di programmazione, grafici ad alto livello, interfacce ad altri linguaggi e funzionalità per il debugging. Il linguaggio R è un dialetto di S che fu sviluppato negli anni '80 e il suo utilizzo si è subito diffuso nella comunità statistica. Il progetto R fu iniziato da Ross Ihaka e Robert Gentlemen all'università di Auckland nel 1996 [25]. Il linguaggio R è in continuo aggiornamento dall'R Core Team [49], la versione attualmente in sviluppo è la 3.2.

Oggi R è una suite integrata di funzionalità software per manipolazione di dati, calcolo e rappresentazione grafica che include: funzionalità per la gestione e memorizzazione dei dati; suite di operatori per il calcolo su array, in particolare matrici; una vasta collezione integrata di tool per l'analisi dei dati; funzionalità grafiche per l'analisi dei dati e la visualizzazione; un linguaggio di programmazione che include condizionali, loop, funzioni ricorsive definite dall'utente e funzionalità di input e output. R è pensato come un ambiente in cui vengono implementate tecniche statistiche e può essere esteso facilmente attraverso dei package disponibili nei siti della famiglia CRAN che coprono una vasta gamma di tecniche statistiche moderne [19]. R è stato utilizzato nella tesi per: effettuare l'analisi preliminare dei dati; eseguire tutte le computazioni per la realizzazione di clustering, LSA e metodi di recommendation; creare i grafici presentati.

7.2 Crawler Java

Il crawler Java è stato creato per ottenere automaticamente pagine HTML di profili pubblici di utenti italiani di LinkedIn. Nel listato 7.1 il codice corrispondente.

Listato 7.1: Crawler Java per il download dei profili pubblici su LinkedIn

```
1  public class LinkedinUserDownloader {
2
3          public static void crawl(URL url) {
4                  System.out.println("Scanning: " + url);
5                  Document doc;
6                  while(true){
7                          try {
8                                  doc = Jsoup.parse(url, 0);
9                                  break;
10                         }catch(UnknownHostException ex){
11                                 System.out.println("Non connesso.");
12                         }catch(SocketException es){
13                                 System.out.println("Non connesso.");
14                         }catch (IOException e) {
15                                 e.printStackTrace();
16                                 return;
17                         }
18                 }
19                 Elements links =
```

```
20              doc.select("ul.directory").first().getElementsByTag("a");
21              for (Element link: links) {
22                      URL dest;
23                      try {
24                              dest = new URL(link.absUrl("href"));
25                      } catch (MalformedURLException e) {
26                              e.printStackTrace();
27                              continue;
28                      }
29                      if (dest.getPath().startsWith("/pub/dir")){
30                              crawlNamesake(dest);
31                      }else if (dest.getPath().startsWith("/directory")){
32                              crawl(dest);
33                      }else{
34                              savePage(dest);
35                      }
36              }
37      }
38
39      public static void crawlNamesake(URL url) {
40              System.out.println("Scanning: " + url);
41              Document doc;
42              try {
43                      doc = Jsoup.parse(url, 0);
44              } catch (IOException e) {
45                      e.printStackTrace();
46                      return;
47              }
48              Elements italianLinks = doc.select("a[class=region-filter]");
49              if(!italianLinks.isEmpty()){
50                      try {
51                      doc = Jsoup.parse(
52                      new URL(italianLinks.first().absUrl("href")),0);
53                      } catch (IOException e) {
54                      e.printStackTrace();
55                      return;
56                      }
57                      Elements links = doc.select("ol[id=result-set]");
58                      if(!links.isEmpty()){
```

```
59                                  Elements l =
60                                          links.first().select("a[class=profile-photo]");
61                                  for(Element link: l){
62                                          URL dest;
63                                          try {
64                                                  dest = new URL(
65                                                  link.absUrl("href"));
66                                          } catch (MalformedURLException e) {
67                                                  e.printStackTrace();
68                                                  continue;
69                                          }
70                                          savePage(dest);
71                                  }
72                          }
73                  }
74          }
75
76          public static void savePage(URL dest) {
77                  PrintWriter writer;
78                  try {
79                          String url = dest.toString();
80                          url = url.replaceAll("[^a-zA-Z0-9.-]", "_");
81                          url = url.substring(7);
82                          InputStream is = dest.openStream();
83                          writer = new PrintWriter(
84                                  "D:/crawlerJava/Users/T/"+url+".html");
85                          BufferedReader br =
86                                  new BufferedReader(new InputStreamReader(is));
87                          String line;
88                          while ((line = br.readLine()) != null) {
89                                  writer.println(line);
90                          }
91                          is.close();
92                          writer.close();
93                  } catch (IOException e) {
94                          e.printStackTrace();
95                  }
96          }
97
```

```
98          public static void main(String[] args) throws IOException {
99                  //URL pagina di partenza
100                 URL seed = new
101                 URL("http://it.linkedin.com/directory/people-a");
102                 System.out.println(seed);
103                 crawl(seed);
104         }
105 }
```

7.3 Applicazione Java per la memorizzazione dei dati sul database

L'applicazione Java implementata nell'ambiente di sviluppo Eclipse ha come risultato finale la creazione di un database in cui sono memorizzati tutti i profili ottenuti tramite il crawler descritto in 2.2. Il numero di profili è di 1 milione e 200 mila perciò è stato necessario realizzare un'applicazione multithread che parallelizzasse i lavoro il più possibile, per rendere i tempi di esecuzione accettabili.
L'applicazione è costituita da diversi thread, ognuno con un compito ben preciso:

- *Main thread.* Il main ha il compito di istanziare tutti i thread necessari alle computazioni, attendere il completamento di tutte le operazioni ed infine terminare l'applicazione.

- *DatabaseInteractor.* Questo thread effettua ciclicamente una get su una coda condivisa con il thread *DataExtractorFromHTML*. La coda è composta da oggetti di tipo PersonProfile che rappresentano il profilo di un utente LinkedIn. Dato il profilo utente, il thread effettua una connessione al database e memorizza le tutte le informazioni (comprese quelle sulle aziende) sfruttando le funzionalità di Hibernate presentate nella sezione 3.3. Nel listato 7.2 la macro-struttura del codice.

- *DataExtractorFromHTML.* La parallelizzazione delle computazioni è realizzata tramite un pool di thread di questo tipo, che accetta un file HTML contenente un profilo utente, effettua il parsing HTML presentato nella sezione 3.2 ed incapsula le informazioni nell'oggetto PersonProfile. Durante il parsing del file HTML, se l'utente ha inserito il link delle aziende presso cui lavora o ha lavorato, il thread inserisce l'URL corrispondente in una coda condivisa con

il thread *RequestThread* ed estrae le informazioni sull'azienda. Infine inserisce l'oggetto nella coda condivisa con il thread *DatabaseInteractor*. Nel listato 7.3 la marco-struttura del codice e un esempio di parsing HTML per l'estrazione delle informazioni sulle lingue.

- *RequestThread*. Ha il compito di effettuare delle get cicliche sulla coda condivisa con il thread *DataExtractorFromHTML*. La coda è composta da stringhe che corrispondono all'URL dell'azienda. Il thread apre uno stream su questo URL e memorizza il file HTML sull'azienda. Nel listato 7.4 il codice completo.

Listato 7.2: DatabaseInteractor: thread che memorizza i profili utente sul database

```
public void run() {
        while(true){
                try {
                        PersonProfile p = queue.take();
                        if(p.getId().equals("FINE")){
                                break;
                        }else{
                                saveProfile(p);
                        }
                        } catch (InterruptedException e) {
                                e.printStackTrace();
                                continue;
                        } catch (org.hibernate.exception.DataException e1){
                                e1.printStackTrace();
                                continue;
                }
        }
}

private void saveProfile(PersonProfile profile) {

        session = sessionFactory.openSession();

        System.out.println("PROFILE: "+profile.getId());
        saveEmployments(profile);
        saveHonorAwards(profile);
        saveLanguages(profile);
        saveSkills(profile);
```

```
29          saveEducations(profile);
30          saveCertifications(profile);
31          saveInterests(profile);
32          saveIndustry(profile);
33          saveTitle(profile);
34          saveGroups(profile);
35
36          session.beginTransaction();
37          session.save(profile);
38          session.getTransaction().commit();
39
40          session.close();
41  }
42
43  private void getLanguages() {
44
45          Elements langs = d.select("div[id=profile-languages]");
46          if(!langs.isEmpty()){
47                  Elements la =
48                  langs.first().select("li[class=competency language]");
49
50                  for(Element e: la){
51                          //language
52                          String language = "";
53                          Elements ls = e.select("h3");
54                          if(!ls.isEmpty()){
55                                  language =
56                                  ls.text().toLowerCase(Locale.ENGLISH);
57                          }
58
59                          //proficiency
60                          String proficiency = "";
61                          Elements p= e.select("span[class=proficiency]");
62                          if(!p.isEmpty()){
63                                  proficiency =
64                                  p.text().toLowerCase(Locale.ENGLISH);
65                          }
66
67                          Language lang =
```

```
68                              new Language(new LanguageID(language, proficiency));
69
70                              profile.addLanguage(lang);
71                      }
72              }
73      }
```

Listato 7.3: DataExtractorFromHTML: thread che incapsula le informazioni sul profilo utente estrapolandole dal file HTML

```
1   public void run() {
2           try {
3                   if(htmlFile.getName().equals("FINE")){
4                           queue.put(new PersonProfile("FINE"));
5                   }else{
6                           d = Jsoup.parse(htmlFile, null);
7                           createProfile();
8                           queue.put(profile);
9                   }
10          } catch (IOException e) {
11                  e.printStackTrace();
12          } catch (InterruptedException e) {
13                  e.printStackTrace();
14          }
15  }
16
17  private void createProfile() {
18
19          getCurrentPositions();
20          getPastPositions();
21          getHonorsAwards();
22          getLanguages();
23          getSkills();
24          getEducationsAndEstimatedAge();
25          getWebSites();
26          getInterests();
27          getGroups();
28          getConnectionsNumber();
29          getCertifications();
30          getIndustry();
```

```
31          getTitle();
32
33  }
```

Listato 7.4: RequestThread: thread che ottiene i file HTML delle aziende

```
1  public void run() {
2          while(true){
3                  try {
4                          AbstractMap.SimpleEntry<String, String> profile =
5                          queue.peek();
6                          if(profile!=null){
7                                  if(profile.getKey().equals("FINE")){
8                                          break;
9                          }else{
10                                 URL url = new URL(profile.getValue());
11                                 InputStream is = url.openStream();
12                                 PrintWriter writer =
13                                 new PrintWriter("D:/crawlerJava/Companies/"+
14                                 profile.getKey());
15                                 System.out.println("COMPANY: "+
16                                 profile.getKey());
17                                 BufferedReader br = new BufferedReader(
18                                 new InputStreamReader(is));
19                                 String line;
20                                 while ((line = br.readLine()) != null) {
21                                         writer.println(line);
22                                 }
23                                 is.close();
24                                 writer.close();
25                                 queue.take();
26                         }
27                 }
28                 }catch (MalformedURLException e) {
29                         e.printStackTrace();
30                 }catch(SocketException ex){
31                         System.out.println("Non connesso.");
32
33                 }catch (java.io.FileNotFoundException e){
34                         try {
```

```
35                                  queue.take();
36                          } catch (InterruptedException e1) {
37                                  e1.printStackTrace();
38                          }
39                  } catch (InterruptedException e) {
40                          e.printStackTrace();
41                  }
42          }
43  }
```

7.4 Clustering di skill

Il clustering di skill, implementato tramite un script R e descritto nella sezione 5.1.2, si ottiene attraverso diverse computazioni: calcolo delle co-occorrenze tra skill; calcolo della similarità di Jaccard tra ogni coppia di skill; clustering gerarchico basato sulla similarità di Jaccard. Nel listato 7.5 il codice corrispondente.

Listato 7.5: Clustering di skill

```
1  library(Matrix)
2
3  #reads skill occurrences and person-skills pairs from file
4
5  skills <- read.csv2("/home/pasini/Desktop/TESI/topic extraction/
       skill/skill maggiore di 20 linkedindb.csv", stringsAsFactors=
       FALSE)
6
7  row.names(skills) <- skills$skills_skill
8
9  skills <- skills[ order((rownames(skills))),]
10
11 person_skills <- read.csv2("/home/pasini/Desktop/TESI/topic
       extraction/skill/personprofiles skills maggiore di 20
       linkedindb.csv", stringsAsFactors=FALSE)
12 person_skills <- person_skills[order(person_skills$personprofiles
       _id),]
13 row.names(person_skills) <- NULL
14
15 #calculates skill co-occurrences
```

```
16
17 person.fac <- factor(person_skills[,1])
18 skill.fac <- factor(person_skills[,2])
19
20 s <- sparseMatrix(
21 as.numeric(person.fac),
22 as.numeric(skill.fac),
23 dimnames = list(
24 as.character(levels(person.fac)),
25 as.character(levels(skill.fac))),
26 x = 1)
27
28 cooccurrences <- as.matrix(t(s) %*% s)
29
30 diag(cooccurrences) <- 0
31
32 occurrences <- skills[,2]
33
34 occ_matrix_1 <- matrix(occurrences, nrow=nrow(skills), ncol=nrow(
    skills))
35
36 occ_matrix_2 <- matrix(occurrences, nrow=nrow(skills), ncol=nrow(
    skills), byrow=TRUE)
37
38 #calculates jaccard similarity between every skill pairs
39
40 jaccard_similarity <- cooccurrences/(occ_matrix_1+occ_matrix_2-
    cooccurrences)
41
42 diag(jaccard_similarity) <- 1
43
44 #hierarchical clustering of skills based on jaccard similarity
45
46 matrix1 <- matrix(1, nrow=nrow(jaccard_similarity), ncol=nrow(
    jaccard_similarity))
47
48 complementary_jaccard <- matrix1 - jaccard_similarity
49
50 dist <- dist(complementary_jaccard)
```

```
51
52 hc_complete <- hclust(dist)
```

7.5 Skill recommendation

Lo script relativo al metodo di skill recommendation, illustrato nella sezione 5.2, si divide in: divisione del dataset in training e test; calcolo delle co-occorrenze tra feature set e skill; calcolo della probabilità di una feature a_i dato lo skill s $P(a_i|s)$; calcolo delle probabilità a priori degli skill; realizzazione della funzione che calcola il vettore degli skill suggeriti per ogni persona. Nel listato 7.6 è mostrato il codice con feature set costituito dall'industry.

Listato 7.6: Skill recommendation

```
1 library(Matrix)
2 library(pbapply)
3
4 #reads industry and person-skill pairs from file
5
6 m_i <- read.csv2("/home/pasini/Desktop/TESI/recommendation system
      paper/mutual information/members_industry_freq.csv",
    stringsAsFactors=FALSE)
7
8 colnames(m_i) <- c('member', 'feature')
9
10 m_i <- within(m_i, feature <- paste('i_', feature))
11
12 members_skills <- read.csv2("/home/pasini/Desktop/TESI/
    recommendation system paper/mutual information/members_skill_
    freq.csv", stringsAsFactors=FALSE)
13
14 colnames(members_skills) <- c('member', 'skill')
15
16 members_skills <- within(members_skills, skill <- paste('s_',
    skill))
17
18 #divides dataset into training and test sets
19
20 member.fac <- factor(members_skills[,1])
```

```
21 feature.fac <- factor(members_skills[,2])
22
23 s <- sparseMatrix(
24 as.numeric(member.fac),
25 as.numeric(feature.fac),
26 dimnames = list(
27 as.character(levels(member.fac)),
28 as.character(levels(feature.fac))),
29 x = 1)
30
31 member.fac <- factor(m_i[,1])
32 feature.fac <- factor(m_i[,2])
33
34 f <- sparseMatrix(
35 as.numeric(member.fac),
36 as.numeric(feature.fac),
37 dimnames = list(
38 as.character(levels(member.fac)),
39 as.character(levels(feature.fac))),
40 x = 1)
41
42 difference <- (rownames(f) %in% rownames(s))
43 difference <- which(difference == FALSE)
44
45 i <-!c(1:nrow(f)) %in% difference
46 f <- f[i,]
47
48 difference <- (rownames(s) %in% rownames(f))
49 difference <- which(difference == FALSE)
50 i <-!c(1:nrow(s)) %in% difference
51 s <- s[i,]
52
53 training_set <- sample(c(1:nrow(s)), as.integer(nrow(s)*55/100))
54
55 training_set <- sort.int(training_set)
56
57 f_training <- f[training_set,]
58
59 s_training <- s[training_set,]
```

```
60
61 test_set <- c(1:nrow(s))[-training_set]
62
63 f_test <- f[test_set,]
64
65 s_test <- s[test_set,]
66
67 #calculates skill-feature cooccurrences
68
69 skill_feature_cooccurrences <- Matrix(t(s_training) %*% f_
       training, sparse=TRUE)
70
71 #calculates skill and feature occurrences
72
73 skill_occurrences <- pbsapply(c(1:ncol(s_training)), function(x)
       sum(s_training[,x]))
74 feature_occurrences <- pbsapply(c(1:ncol(f_training)), function(x
       ) sum(f_training[,x]))
75
76 #computes P(a|s)
77
78 p_a_s <- (1+(skill_feature_cooccurrences))/(length(feature_
       occurrences)+skill_occurrences)
79
80 #computes prior skill probability
81
82 prior_skill_prob <- skill_occurrences/nrow(s_training)
83
84 #calculates vector of recommended skills per person
85
86 get_recommendation_vector <- function(x){
87
88 feature <- which(f_test[x,] == 1)
89
90 p_a_s_vectors <- p_a_s[,feature]
91 if(length(feature)==1){
92 product_vector <- p_a_s_vectors
93 }else{
94 product_vector <- apply(p_a_s_vectors, 1, prod)
```

```
95 }
96
97 recommendation_vector <- product_vector*(prior_skill_prob)
98
99 recommendation_vector <- sort.int(recommendation_vector, index.
       return=TRUE, decreasing=TRUE)
100
101 true <- which(s_test[x,]==1)
102
103 recommended_skill <- recommendation_vector$ix[c(1:50)]
104
105 matches <- length(intersect(true,recommended_skill))
106
107 return (c(matches,length(true)))
108 }
109
110 recommendation_results <- pbsapply(c(1:nrow(s_test)), get_
       recommendation_vector)
```

7.6 Position clustering

Il clustering di position, descritto nella sezione 6.1, è stato realizzato eseguendo precedentemente la LSA sulla matrice di co-occorrenze position-skill per far emergere le associazioni semantiche tra position. Il risultato è stato poi utilizzato come misura di similarità per il clustering gerarchico. Nel listato 7.7 il codice corrispondente.

Listato 7.7: Position clustering

```
1 library(Matrix)
2 library(pbapply)
3 library(lsa)
4
5 #applies LSA to position-skill cooccurrences matrix
6
7 lsa_p <- lsa(position_skill_cooccurrences)
8
9 #terms (positions) matrix scaled compared to the eigenvalues
10
11 lsa_terms = lsa_p$tk %*% diag(lsa_p$sk)
```

```
12
13 #compares every position by cosine similarity
14
15 cosine_matrix <- cosine(t(lsa_terms))
16
17 #hierarchical clustering of position based on LSA
18
19 matrix1 <- matrix(1, nrow=nrow(cosine_matrix), ncol=nrow(cosine_
       matrix))
20
21 distance_matrix <- matrix1 - cosine_matrix
22
23 dist <- dist(distance_matrix)
24
25 hc_complete <- hclust(dist)
```

7.7 Metodo 1 job recommendation

Il primo metodo di job recommendation si basa sulla matrice di Jaccard skill-position come spiegato nella sezione 6.2. Il listato 7.8 mostra la funzione che rappresenta il metodo con cui si suggeriscono le position in base agli skill dell'utente. Il calcolo della matrice di Jaccard skill-position e la divisione del dataset in training e test sono gli stessi mostrati nei listati precedenti.

Listato 7.8: Metodo 1 job recommendation

```
1  library(Matrix)
2  library(pbapply)
3
4  recommend_position <- function(x){
5    index <- which(s_test[x,] == 1)
6    person_vectors <- skill_position_jaccard[index,]
7    if(length(index) == 1){
8      person_vector <- person_vectors
9    }else{
10     person_vector <- colMeans(person_vectors)
11   }
12
```

```
13    similarities <- sort.int(person_vector, index.return=TRUE,
          decreasing=TRUE)
14
15    return (similarities$ix[c(1:50)])
16  }
17
18  recommended_position <- pbsapply(c(1:nrow(s_test)), recommend_
      position)
```

7.8 Metodo 2 job recommendation

Il clustering gerarchico di position è stato utilizzato nella job recommendation in due maniere: come base per la valutazione della recall del primo metodo di job recommendation e per lo sviluppo di un secondo metodo di job recommendation.

Job recommendation gerarchica

Nella sezione 6.3 è stato proposto un nuovo metodo di job recommendation. Nel listato 7.9 il codice relativo.

Listato 7.9: Metodo 2 di job recommendation

```
1  library(Matrix)
2  library(pbapply)
3
4  #hierarchical clustering of training set cutted at height 12
5
6  hc_complete <- hclust(dist)
7
8  cutted_hc_9 <- cutree(hc_complete, h=12)
9
10 skill_cooccurrences <- t(s_training) %*% s_training
11
12 occ_1 <- matrix(skill_occurrences, ncol=ncol(skill_cooccurrences)
       , nrow=nrow(skill_cooccurrences), byrow=TRUE)
13
14 occ_2 <- matrix(skill_occurrences, ncol=ncol(skill_cooccurrences)
       , nrow=nrow(skill_cooccurrences))
15
```

```
skill_jaccard <- skill_cooccurrences / (occ_1 + occ_2 - skill_
    cooccurrences)

#function that returns subtree of a input node

get_subtree <- function(x){
  clu_to_merge <- hc_complete$merge[x,]

  if(clu_to_merge[1] < 0 && clu_to_merge[2] < 0){
    return (clu_to_merge)
  }else if(clu_to_merge[1] > 0 && clu_to_merge[2] < 0){
    return (c(clu_to_merge[2],get_subtree(clu_to_merge[1])))
  }else if(clu_to_merge[1] < 0 && clu_to_merge[2] > 0){
    return (c(clu_to_merge[1],get_subtree(clu_to_merge[2])))
  }else{
    return (c(get_subtree(clu_to_merge[2]),get_subtree(clu_to_
        merge[1])))
  }
}

# method 2 of job recommendation

hierarchical_classification <- function(x,tree_index){

  clu <- hc_complete$merge[tree_index,]

  if(clu[1] < 0 && clu[2] < 0){
    position_1 <- -clu[1]
    position_2 <- -clu[2]
  }else if(clu[1] > 0 && clu[2] < 0){
    position_1 <- -get_subtree(clu[1])
    position_2 <- -clu[2]
  }else if(clu[1] < 0 && clu[2] > 0){
    position_1 <- -clu[1]
    position_2 <- -get_subtree(clu[2])
  }else{
    position_1 <- -get_subtree(clu[1])
    position_2 <- -get_subtree(clu[2])
  }
```

```
53
54    index <- which(s_test[x,] == 1)
55    person_vectors <- skill_jaccard[index,]
56    if(length(index) == 1){
57      person_vector <- person_vectors
58    }else{
59      person_vector <- colMeans(person_vectors)
60    }
61
62    if(length(position_1) > 1){
63      centroid_1 <- colMeans(position_skill_jaccard[position_1,])
64    }else{
65      centroid_1 <- position_skill_jaccard[position_1,]
66    }
67
68    if(length(position_2) > 1){
69      centroid_2 <- colMeans(position_skill_jaccard[position_2,])
70    }else{
71      centroid_2 <- position_skill_jaccard[position_2,]
72    }
73
74    dist_1 <- dist(rBind(person_vector,centroid_1))
75
76    dist_2 <- dist(rBind(person_vector,centroid_2))
77
78    true_pos <- which(p_test[x,] == 1)
79
80    if(dist_1 <= dist_2){
81      if(clu[1] > 0){
82        return (c(true_pos %in% position_1,hierarchical_
              classification(x,clu[1])))
83      }else{
84      return (true_pos %in% position_1)
85      }
86    }else{
87      if(clu[2] > 0){
88        return (c(true_pos %in% position_2,hierarchical_
              classification(x,clu[2])))
89      }else{
```

```
90        return (true_pos %in% position_2)
91      }
92    }
93  }
94
95  recommendation_hierarchical <- pbsapply(c(1:nrow(p_test)),
      function(x) hierarchical_classification(x,2409))
```

Recall del primo metodo basandosi sul clustering gerarchico

Nella sezione 6.3.2 è stato descritto come è stata ricalcolata la recall del primo metodo sfruttando la gerarchia di position. Il listato 7.10 mostra il codice corrispondente.

Listato 7.10: Calcolo recall primo metodo di job recommendation

```
1  library(Matrix)
2  library(pbapply)
3
4  first_method_hierarchical_classification <- function(true_pos,
       tree_index, rec_pos){
5
6    clu <- hc_complete$merge[tree_index,]
7
8    if(clu[1] < 0 && clu[2] < 0){
9      position_1 <- -clu[1]
10     position_2 <- -clu[2]
11   }else if(clu[1] > 0 && clu[2] < 0){
12     position_1 <- -get_subtree(clu[1])
13     position_2 <- -clu[2]
14   }else if(clu[1] < 0 && clu[2] > 0){
15     position_1 <- -clu[1]
16     position_2 <- -get_subtree(clu[2])
17   }else{
18     position_1 <- -get_subtree(clu[1])
19     position_2 <- -get_subtree(clu[2])
20   }
21
22   if(rec_pos %in% position_1){
23     if(true_pos %in% position_1){
24       if(length(position_1) == 1){
```

```
25          return (1)
26        }else{
27          return (c(1+first_method_hierarchical_classification(true
                _pos,clu[1],rec_pos)))
28        }
29      }else{
30        return (0)
31      }
32    }else{
33      if(true_pos %in% position_2){
34        if(length(position_2) == 1){
35          return (1)
36        }else{
37          return (c(1+first_method_hierarchical_classification(true
                _pos,clu[2],rec_pos)))
38        }
39        }else{
40          return (0)
41      }
42    }
43  }
44
45  first_method_hierarchical <- pbsapply(c(1:nrow(p_test)), function
      (x) sapply(c(1:50), function(y) first_method_hierarchical_
      classification(which(p_test[x,] == 1),2409,recommended_
      position[y,x])))
```

Conclusioni

Il lavoro presentato nella tesi consiste nella realizzazione di un sistema di recommendation applicato alla tematica della ricerca ed offerta di lavoro. Non avendo dati su cui lavorare la prima fase è stata quella dell'ottenimento dei dati dal social media LinkedIn, particolarmente adatto al tema di interesse perché anch'esso orientato al lavoro. I profili pubblici degli utenti LinkedIn contengono diversi tipi di informazioni, l'attenzione è stata posta in particolar modo sugli skill, le capacità di una persona, e sulle position, le posizioni lavorative. Sono stati sviluppati diversi metodi di recommendation:

- *Skill Recommendation* che suggerisce agli utenti i possibili skill da aggiungere al profilo sulla base delle altre informazioni di profilo inserite dall'utente. Seguendo il lavoro di un paper di ricercatori LinkedIn [6], sono state fatte diverse prove utilizzando vari feature set; la recall@50 (suggerendo 50 skill) massima ottenuta è di 0.516

- *Job Recommendation 1.* Questo metodo utilizza la similarità di Jaccard tra skill e position per inferire le position più affini ad un set di skill, dove il set di skill può rappresentare una persona che cerca lavoro o un particolare profilo ricercato da una azienda. Misurando la recall binaria, ovvero 1 se la position suggerita è quella vera, 0 altrimenti, si raggiunge un valore @50 di 0.644. Misurando la recall sfruttando la gerarchia di position realizzata, ovvero calcolando il grado di correttezza della position suggerita, si ottiene un valore @50 di 0.862.

- *Job Recommendation 2.* Questo metodo sfrutta la gerarchia di position realizzata applicando l'LSA alla matrice di co-occorrenze skill-position. La recall è misurata come numero di scelte corrette all'interno dell'albero diviso per il numero di step per arrivare alla position vera. Il valore ottenuto è 0.248, mol-

to buono considerando che a differenza dell'altro metodo che suggerisce più position (da 1 a 50), questo ne suggerisce una sola.

La linea guida di tutto il lavoro è stata quella di cercare di sviluppare un recommender system generale, che offra diverse opportunità in base all'utente finale del sistema. Le possibili applicazioni infatti sono molteplici, per esempio se il fruitore del sistema è una persona che cerca lavoro esso potrebbe supportarlo nella scelta dell'offerta di lavoro più affine alle sue caratteristiche, se invece è un'azienda il sistema potrebbe aiutarla nella scelta di candidati per una precisa posizione oppure per una posizione caratterizzata da un insieme specifico di skill.

Bibliografia

[1] Gediminas Adomavicius e Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A Inkeri Verkamo, e altri. Fast discovery of association rules. *Advances in knowledge discovery and data mining*, 12(1):307–328, 1996.

[3] Scott W Ambler. Mapping objects to relational databases: What you need to know and why. *Ronin International*, 2000.

[4] Sitaram Asur e Bernardo A Huberman. Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pp. 492–499. IEEE, 2010.

[5] Marko Balabanović e Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[6] Mathieu Bastian, Matthew Hayes, William Vaughan, Sam Shah, Peter Skomoroch, Hyungjin Kim, Sal Uryasev, e Christopher Lloyd. Linkedin skills: large-scale topic extraction and inference. In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 1–8. ACM, 2014.

[7] Chumki Basu, Haym Hirsh, William Cohen, e altri. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pp. 714–720, 1998.

[8] Christian Bauer e Gavin King. *Java Persistence with Hibernate*. Manning Publications, 2006.

[9] John S Breese, David Heckerman, e Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43–52. Morgan Kaufmann Publishers Inc., 1998.

[10] Zhixiong Chen e Delia Marx. Experiences with eclipse ide in programming courses. *Journal of Computing Sciences in Colleges*, 21(2):104–112, 2005.

[11] Chen-Fu Chien e Li-Fei Chen. Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry. *Expert Systems with applications*, 34(1):280–290, 2008.

[12] Courtney Corley, Armin R Mikler, Karan P Singh, e Diane J Cook. Monitoring influenza trends through mining social media. In *BIOCOMP*, pp. 340–346, 2009.

[13] Pedro Domingos. Mining social networks for viral marketing. *IEEE Intelligent Systems*, 20(1):80–82, 2005.

[14] Korry Douglas e Susan Douglas. *PostgreSQL: a comprehensive guide to building, programming, and administering PostgreSQL databases.* SAMS publishing, 2003.

[15] Susan T Dumais, George W Furnas, Thomas K Landauer, Scott Deerwester, e Richard Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 281–285. ACM, 1988.

[16] Sudheep Elayidom, Sumam Mary Idikkula, Joseph Alexander, e Anurag Ojha. Applying data mining techniques for placement chance prediction. In *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT'09. International Conference on*, pp. 669–671. IEEE, 2009.

[17] James Elliott, Timothy M O'Brien, e Ryan Fowler. *Harnessing hibernate.* O'Reilly Media, Inc., 2008.

[18] Usama Fayyad, Gregory Piatetsky-Shapiro, e Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[19] The R foundation. What is r? `http://www.r-project.org/about.html`.

[20] David Goldberg, David Nichols, Brian M Oki, e Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[21] Adriana Mata Greenwood. *Updating the International Standard Classification of Occupations, ISCO-08*. UN, 2004.

[22] Pritam Gundecha e Huan Liu. Mining social media: A brief introduction. *Tutorials in Operations Research*, 1(4), 2012.

[23] Ido Guy, Naama Zwerdling, Inbal Ronen, David Carmel, e Erel Uziel. Social media recommendation based on people and tags. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp. 194–201. ACM, 2010.

[24] Will Hill, Larry Stead, Mark Rosenstein, e George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[25] Ross Ihaka e Robert Gentleman. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996.

[26] ILO. Isco international standard classification of occupations. `http://www.ilo.org/public/english/bureau/stat/isco/`, 2015.

[27] Christopher Ireland, David Bowers, Michael Newton, e Kevin Waugh. A classification of object-relational impedance mismatch. In *Advances in Databases, Knowledge, and Data Applications, 2009. DBKDA'09. First International Conference on*, pp. 36–43. IEEE, 2009.

[28] Andreas M Kaplan e Michael Haenlein. Users of the world, unite! the challenges and opportunities of social media. *Business horizons*, 53(1):59–68, 2010.

[29] Ioannis Konstas, Vassilios Stathopoulos, e Joemon M Jose. On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 195–202. ACM, 2009.

[30] George Lekakos e Petros Caravelas. A hybrid approach for movie recommendation. *Multimedia tools and applications*, 36(1-2):55–70, 2008.

[31] Kristina Lerman. Social networks and social information filtering on digg. *arXiv preprint cs/0612046*, 2006.

[32] Greg Linden, Brent Smith, e Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[33] Marek Lipczak. Tag recommendation for folksonomies oriented towards individual users. *ECML PKDD discovery challenge*, 84, 2008.

[34] Duen-Ren Liu e Ya-Yueh Shih. Hybrid approaches to product recommendation based on customer lifetime value and purchase preferences. *Journal of Systems and Software*, 77(2):181–191, 2005.

[35] Pasquale Lops, Marco De Gemmis, e Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pp. 73–105. Springer, 2011.

[36] Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, e John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pp. 263–266. ACM, 2003.

[37] Hokey Min e Ahmed Emam. Developing the profiles of truck drivers for their successful recruitment and retention: a data mining approach. *International Journal of Physical Distribution & Logistics Management*, 33(2):149–162, 2003.

[38] Deep Nishar. The next three billion [infographic]. `http://blog.linkedin.com/2014/04/18/the-next-three-billion/`, 2014.

[39] Elizabeth J O'Neil. Object/relational mapping 2008: hibernate and the entity data model (edm). In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1351–1356. ACM, 2008.

[40] Neelamadhab Padhy, Dr Mishra, Rasmita Panigrahi, e altri. The survey of data mining applications and feature scope. *arXiv preprint arXiv:1211.5723*, 2012.

[41] Zizi Papacharissi. The virtual geographies of social networks: a comparative analysis of facebook, linkedin and asmallworld. *New media & society*, 11(1-2):199–220, 2009.

[42] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[43] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, e John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175–186. ACM, 1994.

[44] EDWARD C RILEY. International standard classification of occupations. *Journal of Occupational and Environmental Medicine*, 1(11):615, 1959.

[45] Matthew A Russell. *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. O'Reilly Media, Inc., 2013.

[46] Börkur Sigurbjörnsson e Roelof Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th international conference on World Wide Web*, pp. 327–336. ACM, 2008.

[47] Meredith M Skeels e Jonathan Grudin. When social networks cross boundaries: a case study of workplace use of facebook and linkedin. In *Proceedings of the ACM 2009 international conference on Supporting group work*, pp. 95–104. ACM, 2009.

[48] Ian Soboroff e Charles Nicholas. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI*, volume 99, pp. 86–91, 1999.

[49] R Core Team. R language definition, 2000.

[50] D Michael Titterington, Adrian FM Smith, Udi E Makov, e altri. *Statistical analysis of finite mixture distributions*, volume 7. Wiley New York, 1985.

[51] Sholom M Weiss e Casimir A Kulikowski. Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning and exp. 1990.

[52] Wikipedia. Hibernate — wikipedia, l'enciclopedia libera. `http://it.wikipedia.org/w/index.php?title=Hibernate&oldid=66336951`, 2014.

[53] Wikipedia. Linkedin — wikipedia, l'enciclopedia libera. `http://it.wikipedia.org/w/index.php?title=LinkedIn&oldid=70891637`, 2015.