Università degli studi di
Milano-Bicocca

Big Data in Public and Social Services

Final Project

Un sistema di auto-completamento semantico di skills

Authors:
Simone D'Amico - 850369
s.damico4@campus.unimib.it

Tutor:
Andrea Seveso
a.seveso6@campus.unimib.it

Introduzione

In questo report si descrivono le fasi di sviluppo di un sistema di auto-completamento di parole che sia in grado di suggerire quelle skills maggiormente simili, dal punto di vista semantico, a quelle inserite dall'utente.
Per raggiungere questo obiettivo si utilizzano dei vettori di word embeddings ottenuti con l'architettura FastText.
Nella sezione 1 si descrive il lavoro intorno allo sviluppo del sistema NEO da cui, in parte, nasce e si sviluppa l'idea di questo sistema di suggerimenti, sebbene il lavoro intorni a NEO sia molto ampio, in questa sede ci si concentra sulle modalità che hanno portato all'individuazione delle skills e dei relativi vettori di embeddings. Nella sezione 2 si definisce il modello di suggerimento in modo formale descrivendo l'analisi che ha portato poi alla definizione della metrica utilizzata per fornire i suggerimenti. Nella sezione 3 si descrive l'architettura del sistema e dei due moduli che la compongono. Nella sezione 4 NON LO SO e infine nella sezione 5 si presentano le conclusioni e i futuri sviluppi legati al sistema.

1 Related work

Il progetto si basa su sistema NEO e il relativo paper *NEO: A Tool for Taxonomy Enrichment with New Emerging Occupations*[1]; il cui scopo è quello di realizzare un sistema europeo per raccogliere e classificare annunci di lavoro online (OJVs) per l'intera UE e le sue 32 lingue dell'Unione[2] attraverso l'apprendimento automatico basandosi sulla tassonomia E.S.C.O. (*European Skills, Competences, Qualifications and Occupations*). La classificazione E.S.C.O. identifica e classifica abilità, competenze, qualifiche e occupazioni rilevanti per il mercato del lavoro europeo e l'istruzione e la formazione[3]. E.S.C.O. è stato sviluppato dalla Commissione Europea dal 2010 e fornisce descrizioni di 2942 professioni e 13.485 competenze legate a queste occupazioni, tradotte in 27 lingue (tutte le lingue ufficiali dell'UE più islandese, norvegese e arabo)[3].
Il lavoro descritto nel paper vuole realizzare un sistema che (1) impari i word embeddings di concetti e entità delle tassonomia preservando le relazioni tassonomiche, (2) suggerisca nuove entità per E.S.C.O. estratte da un corpus testuale e (3) valuti, tramite misure G.A.S.C., la loro idoneità come entità di differenti concetti tassonomici[1].
Per la realizzazione del sistema di auto-completamento ci si concentrerà sul descrivere il procedimento con cui sono stati imparati i vettori di embeddings e come sono state estratte le skills usate come suggerimenti.

Il corpus utilizzato contiene 2,119,025 annunci di lavoro pubblicati nel Regno Unito durante il 2018, con riferimento a concetti E.S.C.O. nel settore ICT, ogni annuncio è stato preprocessato eseguendo: (1) tokenization, (2) riduzione lower case, (3) rimozione punteggiatura e stopwords, (4) individuazione degli n-grams ottenedo l'insieme delle skills utilizzate nel sistema di auto completamento.
Nella fase sperimentale sono stati impiegati e valutati tre delle architetture più importanti per imparare word embedding da grandi corpora di testi: GloVe, Word2Vec, and FastText. Ognuno dei tre algoritmi è stato utilizzato con diversi parametri per generare un totale di 260 modelli. In conclusione, il modello con le migliori performance è risultato avere l'architettura FastText con algoritmo CBOW e 300 componenti per vettore. L'utilizzo di FastText come generatore di embeddings permette di avere a disposizione, non solo i vettori associati alle skills ma anche gli embeddings delle varie loro sotto-parole. Questo permette di gestire errori di type nell'inserimento di parole e poter calcolare la similarità in modo incrementale ad ogni nuovo input dell'utente.

2 Definizione del problema

In questa sezione si descrive il modello su cui è stato realizzato il sistema di auto-completamento da un punto di vista formale, si introducono i dati a disposizione, la modellazione del problema proposto e la metrica di valutazione utilizzata per individuare i suggerimenti più adatti.
Dal lavoro proposto da Giabelli et al[1] si sono ottenute un totale di 2319 skills, che possono essere singole parole o espressioni, inoltre si hanno a disposizione vettori composti da 50 componenti che rappresentano non solo le skills ma anche loro sotto-stringhe, per un totale di 113656 vettori.

2.1 Formalizzazione del problema

Nella formalizzazione del problema si identifica:

- L'insieme delle skills S.
- L'input dell'utente i che può essere l'esatta skill o una sua sotto-stringa.
- Il contesto $C = \{c_1, \ldots, c_n\}$ delle skill già inserite dall'utente.

Per ogni input dell'utente si considera la stringa inserita fino a quel momento, si calcola poi la similarità del coseno tra i e ogni skills ottenendo delle coppie

composte dalla skill e dalla sua similarità con l'input, si escludono le skills già inserite, cioè quelle contenute nel contesto:

$$A = \{(s, sim(i, s)) \mid s \in S \,\&\, s \notin C\} \tag{1}$$

Successivamente si considera il contesto, e quindi, per ogni skill già inserita, si calcola la sua similarità con le skill candidate come suggerimento:

$$B = \{(s, sim_{avg}(s, C)) \mid s \in S \,\&\, s \notin C\} \tag{2}$$

Dove $sim_{avg}(s, C)$ indica la media delle similarità del coseno della skill s con le skills nel contesto C ed ècalcolata per ogni skill candidata:

$$sim_{avg}(s, C) = \frac{\sum_{c \in C} sim(s, c)}{|C|} \tag{3}$$

Nell'insieme A viene calcolata quindi la similarità tra l'input dell'utente e le skills a disposizione mentre nell'insieme B si calcola la similarità tra le skill da poter suggerire e quelle già inserite.

2.2 Metrica di valutazione delle similarità

Calcolate le similarità sia considerando l'input che le skills già inserite, si passa poi ad identificare la skill maggiormente correlata da poter suggerire all'utente. Si utilizza una combinazione lineare pesata con un parametro $\alpha \in [0, 1]$, per ogni skill si calcola:

$$sim_s = \alpha \cdot sim_A + (1 - \alpha) \cdot sim_B \qquad \begin{array}{l} \forall s \in S \,\&\, s \notin C \\ (s, sim_A) \in A \\ (s, sim_B) \in B \end{array} \tag{4}$$

Utilizzando l'equazione (4) si calcolano le similarità finali di tutte le skills e si suggerisce quella con valore di similarità massimo.
L'utilizzo del parametro α permette al modello di essere flessibile e, a seconda dello scenario di utilizzo, dare maggiore importanza all'input corrente dell'utente o alle skills già inserite: per valori di α vicini a 1 aumenta l'importanza della skill che l'utente sta inserendo mentre per valori di α vicini a 0 il contesto delle skills già inserite assume sempre maggiore importanza nella scelta del suggerimento.

3 Implementazione del sistema

Il sistema è quindi in grado di fornire due tipi di suggerimenti: consiglia le skills più correlate sia con il contesto che con l'input dell'utente come

descritto nella sezione precedente, inoltre, fornisce suggerimenti ignorando l'input dell'utente e considerando solo le skills complete del contesto.
Il sistema è stato realizzato utilizzando il linguaggio di programmazione *Python* e l'applicazione open-source *Jupyter Notebook*.
Il sistema si compone di due componenti principali realizzate come classi *Python*: il modulo *AutoCompleteManager* implementa il modello di autocompletamento descritto nella sezione 3, mentre il modulo *FormUI* gestisce una piccola interfaccia utente con la quale interagire con il sistema. L'interazione avviene tramite un'area di testo nella quale inserire le skills di interesse, i suggerimenti vengono mostrati come bottoni, nel momento della pressione dello specifico bottone, la relativa skill viene aggiunta all'area di testo.
Per migliorare l'esperienza utente, i suggerimenti sono mostrati a partire dal quarto carattere inserito.
Si è cercato di implementare il sistema in modo che sia più flessibile possibile e facilmente riutilizzato in contesti diversi, questo si è tradotto nella scelta dei parametri di inizializzazione del sistema; si lascia quindi all'utente la scelta di alcuni parametri, come il numero di suggerimenti da mostrare (di default 4) e il valore di α usato per la pesatura delle similarità, inoltra l'utente può definire egli stesso sia la lista delle skills da utilizzare e sia i vettori di embeddings delle varie parole. In questo modo il sistema è applicabile ad una grande varietà di scenari.

3.1 Modulo AutoCompleteManager

Il modulo è stato realizzato come classe *Python* con diverse funzioni, ognuna dedicata ad uno specifico compito:

- Il costruttore della classe si occupa di inizializzare una serie di parametri come la lista delle skills, i vettori di FastText, il parametro α per il calcolo della similarità e crea un'istanza della classe *FormUI* da usare come interfaccia.

- La funzione get_skills_input_similarity inizializza l'insieme A definito nell'equazione (1). Utilizza un dizionario le cui chiavi sono le skills e i rispettivi valori sono le similarità delle skills con l'input dell'utente inserito fino a quel momento.

- La funzione get_skills_context_similarity inizializza l'insieme B definito nell'equazione (2). Utilizza un dizionario le cui chiavi sono le skills e i rispettivi valori sono le similarità delle skills con il contesto.

- La funzione get_best_similarity_skill calcola la similarità totale come specificato in (4) richiamando le due funzioni precedenti.

- suggest_interest_skill consiglia le skills maggiormente correlate al contesto, ignorando l'input corrente dell'utente.
- La funzione add_skill è eseguita ad ogni pressione del bottone e inserisce la skill associata al bottone cliccato nell'area di testo, inoltre resetta i bottoni permettendo di mostrare successivamente nuovi suggerimenti.
- La funzione suggests_manager è la funzione di listener associata all'area di testo nel quale l'utente inserisce le skills, viene invocata ogni volta che si inserisce un nuovo carattere nell'area di testo. Individua i migliori suggerimenti richiamando le varie funzioni.
- show_form richiama l'omonima funzione nel modulo *FormUI* che permette di visualizzare l'interfaccia.

3.2 Modulo FormUI

Il modulo *FormUI* si occupa di inizializzare il form che compone di fatto l'interfaccia del sistema. Come mostrato in figura 3.2, il form è composto

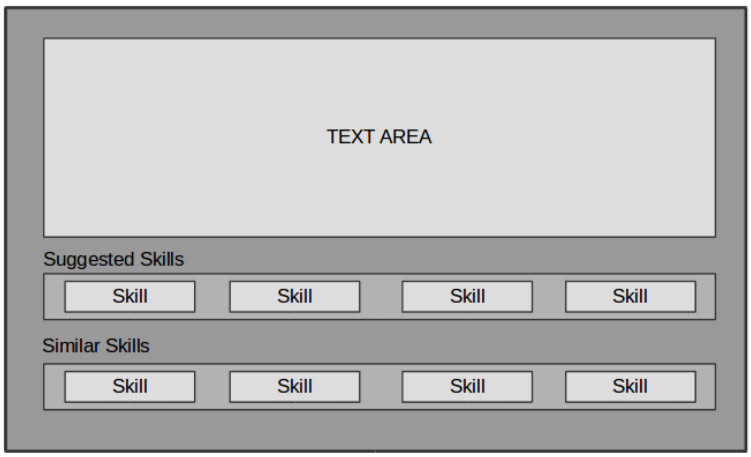

Figure 1: Struttura dell'interfaccia utente.

da tre sezioni distinte: la text area, con cui l'utente inserisce le skills di interesse e in cui si auto-completa l'input secondo il suggerimento scelto, la sezione *Suggested Skills* in cui sono presenti i bottoni etichettati con il nome della skill e la sua similarità; in questa sezione i suggerimenti sono calcolati considerando sia l'input che il contesto tramite la loro combinazione lineare e, infine, la sezione *Similar Skills* in, analogamente alla precedente, sono mostrati i bottoni associati alle skills che sono più simili solo al contesto.
La classe Python implementa una serie di funzioni:

- Il costruttore inizializza le tre aree sopra descritte associandovi anche un layout per la loro posizione.
- La funzione show_form mostra l'intero form.
- La funzione init_button con cui creare i bottoni dei suggerimenti, alla funzione vengono passati il testo da mostrare come descrizione del bottone e come tooltip al passaggio del mouse su di esso.
- Le funzioni close_suggest_buttons e close_suggest_buttons_context rimuovono i bottoni, rispettivamente della sezione *Suggested Skills* e *Similar Skills*, successivamente alla scelte dell'utente permettendo la visualizzazione di bottoni con nuovi suggerimenti.

Nella figura 3.2 si può vedere l'interfaccia utente in attività durante l'interazione, sono presenti le tre sezioni descritte con i relativi suggerimenti.

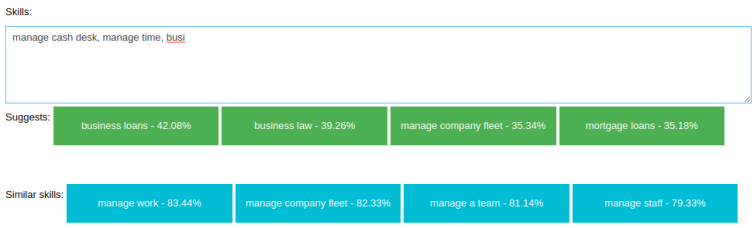

Figure 2: Interfaccia utente durante l'interazione.

4 Analisi delle prestazioni

5 Conclusioni

Il sistema presentato è in grado di fornire suggerimenti in modo dinamico secondo gli interessi dell'utente aiutandolo sia nel completare una skill che si sta inserendo e sia consigliando le skills più simili in assoluto rispetto a quelle già inserite.
Ulteriore oggetto di studio può essere il comportamento del sistema se si utilizzano differenti metriche di similarità, ad ora quella che viene calcolata è la similarità del coseno, ma possono essere impiegate anche altre misure come la distanza Euclidea o il dot product.
Il meccanismo di suggerimento è indipendente dal contesto ed è in grado di funzionare anche per parole e espressioni che non siano necessariamente skills

tecniche. Questo suo aspetto lo rende adatto ad essere impiegato anche in contesti reali, al di fuori quindi di quello universitario. La logica del sistema e separata dalla parte dell'interfaccia realizzata tramite i widgets di *Jupyter Notebook*, quindi, pensando ad un utilizzo professionale, questa può essere inserita all'interno di applicazioni web o di interfacce di programmi desktop.

References

[1] Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., & Seveso, A. (2020). NEO: A Tool for Taxonomy Enrichment with New Emerging Occupations.

[2] CEDEFOP: Real-time labour market information on skill requirements: Setting up the eu system for online vacancy analysis. https://goo.gl/5FZS3E (2016)

[3] Handbook, E. S. C. O. European Skills, Competences, Qualifications and Occupations (2017). EC Directorate E.