Università degli Studi di Milano-Bicocca

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di Laurea in Informatica

RETI NEURALI PER L'APPRENDIMENTO DEI TRATTI DELLA PERSONALITÀ DAL LINGUAGGIO NATURALE

Relatore: Prof. Stella Fabio Antonio
Co-relatore: Dott. Marelli Marco

Relazione della prova finale di:
Giorgia Adorni
Matricola 806787

Anno Accademico 2017-2018

A mio padre, per il suo sostegno quotidiano.
Ad Elia, per tutto il supporto e l'amore dimostrato.

Ringraziamenti

Grazie al mio relatore Fabio Stella, per avermi trasmesso la sua passione e per avermi fornito gli strumenti necessari per intraprendere questo percorso.
Grazie ai ragazzi del Laboratorio MAD (Models and Algorithms for Data & Text Mining), per tutti i loro consigli.

Abstract

La personalità è considerata come uno degli argomenti di ricerca più influenti in psicologia poiché predittiva di molti esiti consequenziali come la salute mentale e fisica, ed è in grado di spiegare il comportamento umano. Grazie alla diffusione dei Social Network come mezzo di comunicazione, sta diventando sempre più importante sviluppare modelli che possano leggere automaticamente e con precisione l'essenza di individui basandosi esclusivamente sulla scrittura.
In particolare, la convergenza tra scienze sociali e informatiche ha portato i ricercatori a sviluppare approcci automatici per estrarre e studiare le informazioni "nascoste" nei dati testuali presenti in rete. La natura di questo progetto di tesi è altamente sperimentale, e la motivazione alla base di questo lavoro è presentare delle analisi dettagliate sull'argomento, in quanto allo stato attuale non esistono importanti indagini che si basino interamente su testo in linguaggio naturale.
L'obiettivo è identificare un adeguato spazio semantico che permetta di definire sia la personalità dell'oggetto a cui un determinato testo si riferisce, sia quella dell'autore. Punto di partenza è un dizionario di aggettivi che la letteratura psicologica definisce come *marker* dei cinque grandi tratti di personalità, i Big Five.
In questo lavoro siamo partiti dall'implementazione di reti neurali fully-connected come base per capire come modelli semplici di Deep Learning possano fornire informazioni sulle caratteristiche nascoste della personalità.
Infine, utilizziamo una classe di algoritmi distribuzionali inventati nel 2013 da *Tomas Mikolov*, che consistono nell'utilizzo di una rete neurale convoluzionale in grado di imparare, in modo non supervisionato, i contesti delle parole. In questo modo costruiamo un embedding in cui sono contenute le informazioni semantiche del testo, ottenendo una sorta di "geometria del significato" in cui i concetti sono tradotti in relazioni lineari. Con quest'ultimo esperimento ipotizziamo che uno stile di scrittura individuale sia in gran parte accoppiato con i tratti della sua personalità.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

La personalità è un fattore chiave che influenza le interazioni, i comportamenti e le emozioni delle persone. Al giorno d'oggi, essa viene considera come uno degli argomenti di ricerca più influenti in psicologia.

La crescente immersione negli ambienti digitali e la diffusione dei social network come mezzo di comunicazione, ha contribuito alla creazione di un enorme quantità di dati, anche chiamati *Big Data*, aprendo la necessità allo sviluppo di modelli automatici in grado di leggere con precisione l'essenza degli individui basandosi esclusivamente sulla scrittura.

L'esigenza di produrre analisi sempre più velocemente ha imposto lo sviluppo di metodi meccanici per selezionare e interpretare i dati, favorendo la ricerca nel campo dell'apprendimento automatico o *Machine Learning* [1].

Nello specifico, il *Data Mining* è un approccio che consiste nell'individuazione d'informazioni significative tramite l'applicazione di algoritmi in grado di determinare le associazioni "nascoste" tra di esse [2, 3].

Una sua forma particolare è il *Text Mining*, nell'ambito del quale si sono sviluppate metodologie che consentono ai computer di confrontarsi con il linguaggio umano, di elaborarlo e comprenderlo [4].

L'interesse nello studio delle informazioni digitali e le abilità necessarie per farlo non sempre coincidono tra gli scienziati sociali. Di conseguenza, tale ricerca viene generalmente affidata a scienziati e ingegneri informatici, facilitando la scoperta di modelli che non sarebbe possibile individuare ed offrendo l'opportunità di instaurare collaborazioni interdisciplinari.

La maggior parte degli attuali studi automatici di rilevamento della personalità si sono concentrati sulla teoria dei *Big Five* come quadro per studiare le caratteristiche intrinseche dell'essere umano [5]. Secondo questo modello esistono cinque dimensioni fondamentali dei tratti, stabili nel tempo e condivisi a livello interculturale. Le cinque caratteristiche,

note appunto come i “Grandi Cinque”, sono Openness (apertura all'esperienza), Conscientiousness (coscienziosità), Extraversion (estroversione), Agreeableness (gradevolezza), Neuroticism (nevroticismo), riconosciuti dall'acronimo OCEAN.

Sviluppare un modello accurato e aprire questa domanda di ricerca avrebbe implicazioni significative in diversi ambiti della sociologia, ma non solo.

Struttura della tesi

Di seguito si passano in rassegna gli argomenti affrontati capitolo per capitolo.

Nel capitolo Contesto vengono introdotti i concetti teorici alla base del lavoro, in particolare viene introdotta la teoria dei Big Five.

Nel capitolo Reti Neurali si descrivono le principali tecniche di Deep Learning, in particolare soffermandosi sulle architetture adottate negli esperimenti.

Nel capitolo Formulazione viene definito il problema ed illustrati approcci e strumenti risolutivi.

Nel capitolo Esperimenti e risultati viene presentata una panoramica degli esperimenti effettuati e una relativa analisi dei risultati.

Nel capitolo Conclusioni vengono esposte le considerazioni finali.

Motivazioni

La natura di questo progetto di tesi è altamente sperimentale. Le motivazioni che hanno portato alla realizzazione di questo lavoro sono la presentazione di analisi dettagliate sull'argomento, in quanto allo stato attuale non esistono importanti indagini di questo tipo.

Contributi

I dati che verranno utilizzati per definire lo spazio semantico e testare la sua funzionalità sono messi a disposizione da Yelp Dataset Challenge, che contiene 5 200 000 recensioni relative a 174 000 attività commerciali di 11 aree metropolitane nel mondo.

Capitolo 1

Contesto

Con il termine *personalità* si intende l'insieme delle caratteristiche psichiche e dei comportamentali abituali — inclinazioni, interessi e passioni — che definiscono e differenziano ogni individuo, nei vari contesti ed ambienti in cui la condotta umana si sviluppa [6, 7].

La tradizione di studi psicologici relativi alla personalità è una delle più rilevanti della psicologia contemporanea, un campo in cui si susseguono studi empirici, teorici e storici, volti a comprendere la natura dell'identità personale nel contesto biologico e sociale di sviluppo. Essa tenta di spiegare le tendenze che sono alla base delle differenze comportamentali, ed ogni gruppo di pensiero tenta di concettualizzare la personalità entro modelli diversi — adoperando metodi, approcci, obiettivi e modalità d'analisi — anche molto dissonanti fra loro.

Una significativa parte della psicologia che studia le differenze individuali, analizza e valuta la personalità attraverso specifici test volti ad individuarne i tratti.

1.1 Big Five

Le teorie della personalità basate sui tratti definiscono la personalità come l'insieme delle caratteristiche che stabiliscono il comportamento di una persona.

La teoria dei Grandi Cinque (o Big Five) risulta essere uno dei modelli più condivisi e testati, sia a livello teorico che empirico. McCrae e Costa identificano cinque grandi dimensioni in cui può essere suddivisa la personalità [8, 9]:

- L'*apertura all'esperienza* o "openness" (creativo/curioso vs. coerente/cauto) è intesa come attitudine alla ricerca di stimoli culturali e di pensiero esterni al proprio contesto ordinario. Essa riflette il grado di curiosità intellettuale, la creatività o una preferenza per la novità; può inoltre essere percepita come imprevedibilità o

mancanza di concentrazione.
Individui con un'elevata apertura perseguono l'auto-realizzazione, cercando esperienze intense ed euforiche. Viceversa, coloro che hanno una bassa apertura cercano di ottenere soddisfazione attraverso la perseveranza.

- La *coscienziosità* o "conscientiousness" (organizzato vs. negligente) è una tendenza caratterizzata dall'organizzazione, precisione e affidabilità. Un soggetto contraddistinto da questa attitudine, preferisce un comportamento pianificato piuttosto che spontaneo.
  Spesso l'alta coscienziosità viene percepita come testardaggine e ossessione, mentre la bassa coscienziosità è associata alla flessibilità e alla spontaneità, ma può anche apparire come mancanza di affidabilità.

- L'*estroversione* o "extraversion" (estroverso/energetico vs. solitario/riservato) è intesa come grado di entusiasmo negli atteggiamenti che si adottano e tendenza a cercare la stimolazione in compagnia degli altri.
  L'alta estroversione è spesso percepita come una ricerca di attenzioni e prepotenza. La bassa estroversione causa una personalità riservata, riflessiva, che può essere avvertita come distaccata.

- La *gradevolezza* o "agreeableness" (amichevole/compassionevole vs. provocatorio/-distaccato) è indicata come quantità e qualità delle relazioni interpersonali che la persona intraprende, orientate al prendersi cura dell'altro. È una tendenza ad essere compassionevoli e collaborativi piuttosto che sospettosi e antagonisti.
  L'alta gradevolezza è spesso vista come ingenuità o sottomissione. Le persone con scarsa gradevolezza sono spesso competitive o sfidanti, e possono essere intese come inaffidabili.

- Il *nevroticismo* o "neuroticism" (sensibile/nervoso vs. sicuro/fiducioso), è una misura di resistenza a stress di tipo psicologico, come l'ansietà e l'irritabilità, ma si riferisce anche al grado di solidità emotiva e di controllo degli impulsi.
  Un'alta stabilità si manifesta in una personalità calma che però può essere vista come poco interessante e indifferente. Una bassa stabilità esprime reattività e dinamicità in individui che spesso possono essere percepiti come instabili o insicuri.

Queste dimensioni sono state individuate a partire da studi psico-lessicali, secondo cui le cinque dimensioni corrisponderebbero alle macro-categorie più usate nel linguaggio per descrivere le diversità fra individui.

Le regioni cerebrali che codificano i vari tratti di personalità sono spesso collegate alle regioni responsabili della comunicazione verbale e scritta.

Un grande numero di prove di ricerca hanno supportato il modello a cinque fattori, che sembra essere condiviso a livello interculturale — Cina, Giappone, Italia, Ungheria, Turchia [10].

Le dimensioni di Big Five predicono accuratamente il comportamento e vengono utilizzate sempre più spesso per aiutare i ricercatori a comprendere l'estensione dei disturbi psicologici come ansia e depressione [11].

Uno dei vantaggi principali di questo approccio è che consente di concentrare l'attenzione solo sulle dimensioni di base piuttosto che studiare centinaia di tratti.

Capitolo 2

Reti Neurali

Nel campo dell'apprendimento automatico, o *machine learning*, una rete neurale artificiale in inglese *Artificial Neural Network* (ANN), è un modello matematico basato sulla semplificazione delle reti neurali biologiche [1].

Una rete neurale può essere considerata come un sistema dinamico avente la topologia di un grafo orientato, i cui nodi modellano i neuroni in un cervello biologico, mentre gli archi rappresentano le sinapsi (interconnessioni di informazioni).

Ogni connessione può trasmettere un segnale da un neurone artificiale a un altro, i quali sono tipicamente aggregati in strati. Gli stimoli vengono ricevuti da un livello di nodi d'ingresso, detto unità di elaborazione, che elabora il segnale e lo trasmette ad altri neuroni ad esso collegati.

2.1 Modello

Le reti neurali possono essere viste come semplici modelli matematici che definiscono una funzione $f : X \to Y$.

La funzione di rete di un neurone $f(x)$ è definita come una composizione di altre funzioni $g_i(x)$, che possono a loro volta essere scomposte in altre funzioni.

Una rappresentazione ampiamente utilizzata per la descrizione di ANN tradizionali è la *somma ponderata*, mostrata nell'equazione 2.1.

$$f(x) = K\left(\sum_i w_i x_i + b\right) \tag{2.1}$$

Ogni segnale in ingresso x_i viene moltiplicato ad un corrispondente peso w_i, che assume valore positivo o negativo a seconda che si voglia eccitare o inibire il neurone. Il bias b varia secondo la propensione del neurone ad attivarsi, influenzandone l'uscita. Inoltre, viene

applicata una funzione predefinita K, detta anche *funzione di attivazione*, illustrata nella seguente sezione.

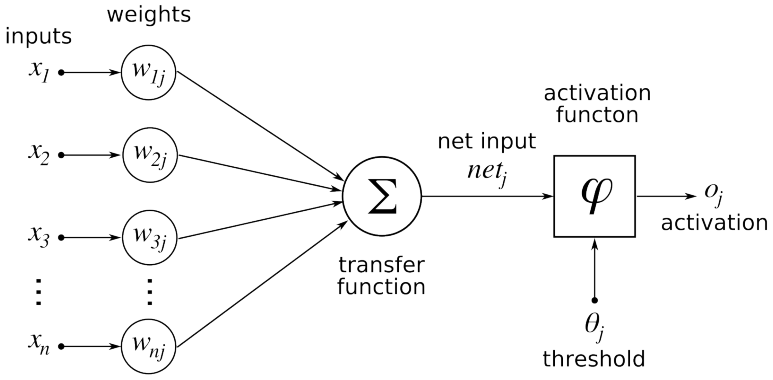


Figura 2.1: Artificial Neuron Model

2.1.1 Funzioni di attivazione

Una funzione di attivazione è una componente fondamentale del modello. Essa consente alla rete di imparare trasformazioni non lineari, in modo da essere in grado di calcolare problemi non banali utilizzando un limitato numero di nodi.

Una delle funzioni più utilizzate è la *sigmoide* $\sigma(x)$, la quale modella la frequenza degli stimoli emessi, da neurone inattivo, $\sigma(x) = 0$, a neurone completamente saturo con una frequenza di attivazione massima, $\sigma(x) = 1$.

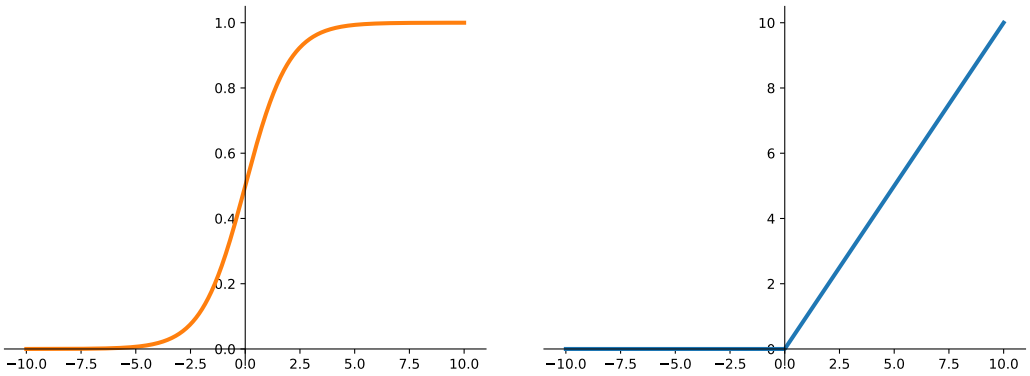
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Negli ultimi anni è diventata molto popolare la *Rectified Linear Unit* (ReLU) [12, 13, 14, 15], definita dalla seguente equazione:

$$f(x) = \max(0, x) = \begin{cases} x & \text{se } x > 0 \\ 0 & \text{altrimenti} \end{cases} \tag{2.3}$$

Questa funzione azzera tutti i valori negativi, mentre ritorna invariati quelli positivi.

Essa viene utilizzata per la sua capacità di accelerare notevolmente il processo di ottimizzazione, inoltre la sua implementazione risulta semplice ed efficiente.



(a) *Sigmoide* (b) *ReLU*

Figura 2.2: Andamento di due funzioni di attivazione

2.2 Architetture

I neuroni vengono organizzati in una struttura detta architettura della rete. I dati, partendo da un livello iniziale, chiamato layer di input, attraversano i multipli strati interni della rete, gli hidden layer, raggiungendo l'ultimo livello detto layer di output.

Quando i collegamenti tra i neuroni formano una struttura senza cicli si parla di reti *feed-forward* [16].

2.2.1 Layer fully-connected

Un'architettura molto comune nelle reti neurali è una struttura "densa", che utilizza *layer fully-connected*, in cui tutti i neuroni del livello precedente sono collegati ad ogni neurone dello strato successivo [17].

Lo scopo di un layer completamente connesso è imparare combinazioni non lineari di feature ad alto livello provenienti dal layer precedente. Una struttura di questo tipo è però caratterizzata da un numero di connessioni che cresce molto velocemente, causando un accrescimento del numero di parametri che la rete deve apprendere. Questo comporta un aumento del costo computazionale e un alto rischio di overfitting, approfondito nella sezione 2.4.2.

Per questo motivo questi vengono spesso sostituiti dai layer convoluzionali.

2.2.2 Layer convoluzionali

Una rete neurale convoluzionale, in inglese *Convolutional Neural Network* (CNN), è una classe di reti artificiali avanzate e *feed-forward*, composte da uno o più strati convoluzionali seguiti da una serie di livelli completamente connessi [18].

Una convoluzione può essere considerata come una funzione a finestra scorrevole, detta *kernel* o filtro, applicata a una matrice. Ogni livello applica filtri diversi, in genere centinaia o migliaia, e combina i loro risultati. Durante la fase di addestramento, una CNN impara automaticamente i valori dei suoi filtri in base all'attività che si desidera eseguire.

Due parametri che definiscono il comportamento di un layer convoluzionale sono lo *stride*, che rappresenta il passo di convoluzione e viene utilizzato per ridurre le dimensioni spaziali dell'output e il *padding*, che definisce il comportamento dei neuroni lungo i bordi dei dati di input. Se viene scelto un padding *valid*, l'output contiene solo i neuroni la cui regione di convoluzione è completamente contenuta nei dati; nel caso di padding *same*, invece, l'output mantiene la stessa dimensione dell'input, utilizzando zero come valore per i dati mancanti.

Le reti convoluzionali sono adatte per elaborare dati visivi e altri dati bidimensionali, ed hanno mostrato ottimi risultati nel riconoscimento di immagini e nella manipolazione del linguaggio naturale [19]. In quest'ultimo caso, l'input è costituito da frasi o documenti rappresentati come una matrice, in cui ogni riga corrisponde a un token, in genere una parola. Tipicamente, questi vettori sono dei word embeddings (rappresentazioni a bassa dimensione) come word2vec [20], ma potrebbero anche essere vettori unici che indicizzano la parola in un vocabolario.

Nel *Natural Language Processing* (NLP) utilizziamo filtri che scorrono su righe complete della matrice (parole), pertanto, la loro "larghezza" è solitamente uguale alla quella della matrice di input. L'altezza, o la dimensione della regione, può variare, ma le finestre tipicamente scorrono su 2-5 parole per volta.

Risulta che le CNN applicate ai problemi di NLP funzionino abbastanza bene, un esempio è il modello *bag-of-words* che è stato l'approccio standard per anni e ha portato a risultati piuttosto buoni [21].

Le CNN sono più facili da addestrare e hanno molti meno parametri da stimare. Il minor numero di connessioni e pesi di questa architettura, rende gli strati convoluzionali relativamente economici in termini di memoria e potenza di calcolo necessari.

2.2.3 Layer di pooling

Il *pooling* è un processo alquanto comune nelle reti neurali, la cui funzione è ridurre progressivamente la dimensionalità spaziale per diminuire la quantità di parametri e la complessità computazionale della rete, mantenendo le informazioni più salienti e controllando anche l'overfitting.

Il layer di pooling opera indipendentemente su ogni slice di profondità dell'input e lo ridimensiona spazialmente, usando l'operazione MAX sul risultato di ogni filtro [22].

Una proprietà del pool è che fornisce una matrice di output a dimensione fissa — in genere richiesta per la classificazione. Ciò consente di utilizzare frasi di dimensioni variabili e filtri di dimensioni variabili, ma ottenere sempre le stesse dimensioni di output da inserire in un classificatore.

2.2.4 Batch Normalization

Durante la fase di addestramento del modello, i parametri di ogni substrato vengono ottimizzati al fine di minimizzare l'errore finale. Ad ogni iterazione, in ciascuno strato avviene una variazione dell'output, corrispondente ad una variazione nei valori in ingresso al livello successivo. Questo può rappresentare un problema per la rete, che deve adattare i propri strati ad un continuo cambiamento nell'input.

Per aumentare la stabilità della rete neurale, velocizzare il training e migliorarne le performance, generalmente viene applicata ad ogni strato una *Batch Normalization* [23], la quale normalizza l'uscita di un precedente livello sottraendo il valore medio di un batch e dividendo il risultato per la sua deviazione standard.

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}} \tag{2.4}$$

L'applicazione di questa operazione ad ogni input potrebbe cambiare ciò che il layer può rappresentare. Per questo motivo viene assicurato che la trasformazione inserita nella rete possa rappresentare l'identità, in modo da poter annullare il potenziale effetto della Batch Normalization, nel caso in cui fosse l'azione ottimale. Dunque viene prevista l'aggiunta di due parametri — "deviazione standard" γ e "media" β — che la rete impara assieme ai parametri del modello originale, come mostrato nell'equazione 2.5

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}. \tag{2.5}$$

La *Batch Normalization* può essere utilizzata sia su reti *feed-forward*, sia sulle *reti convoluzionali*. In questo secondo caso, media e varianza vengono calcolate per ogni filtro.

2.3 Apprendimento

Per insegnare alla rete a risolvere un determinato problema, occorre una fase di addestramento in cui vengono condotte una serie di osservazioni per stabilire quali valori assegnare ad ogni parametro della rete e trovare un modello ottimale.

Questo processo di apprendimento viene strutturato come un problema di ottimizzazione in cui lo scopo è minimizzare una *funzione di costo*, che misura la distanza tra una soluzione particolare ed una ottima.

2.3.1 Funzione di costo

Una funzione di costo mappa un evento ad un numero reale, il quale ne rappresenta intuitivamente il "costo".

Nella strategia adottata si utilizzeranno due diverse funzioni obiettivo: l'*errore quadratico medio* per risolvere il problema di regressione, e la *Softmax Cross Entropy* per il compito di classificazione. Mentre per la costruzione dell'embedding verrà applicata la *Noise Contrastive estimation*.

Mean Squared Error

Per il task il cui compito è prevedere dei valori reali è comune calcolare lo scostamento tra la quantità prevista dalla rete ($\hat{Y}$) e i valori osservati Y (ground truth).

L'*errore quadratico medio* (MSE) di uno stimatore misura la media dei quadrati degli errori, e viene calcolato come

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2. \tag{2.6}$$

con n numero delle previsioni [24].

Softmax Cross Entropy

Softmax è una funzione di loss comunemente utilizzata per la classificazione. In particolare viene applicata allo strato finale della rete ed addestrata in un regime di entropia incrociata [25].

L'entropia incrociata è un indicatore che può essere utilizzato per misurare l'accuratezza delle previsioni.

Date due variabili casuali discrete p e q definiamo l'entropia nel modo seguente:

$$H(p,q) = -\sum_{x} p(x) \log q(x) \tag{2.7}$$

in cui p_x è la "vera" probabilità o distribuzione, mentre q_x è la distribuzione "innaturale" ottenuta a partire dal modello corrente.

Nella pratica la *Cross Entropy* viene calcolata empiricamente ipotizzando l'equiprobabilità degli eventi, poiché p è ignota. Di conseguenza, è ridefinibile come segue

$$H(q) = -\frac{1}{N} \sum_{x} \log q(x) \tag{2.8}$$

dove N è il numero di eventi osservati.

L'obiettivo principale di questa funzione è rendere il risultato della Softmax campionata uguale a quella vera. L'algoritmo si concentra sulla selezione di campioni specifici dalla distribuzione data per ottenere la loss desiderata [26]. L'utilizzo della funzione Softmax ha un effetto considerevole sulle prestazioni.

Noise Contrastive estimation

La stima contrastiva del rumore è una strategia utilizzata nell'ambito della modellazione linguistica o per la generazione di word embedding dati in input dei corpus molto ampi.

La funzione obiettivo del modello *skip-gram* cerca di trovare rappresentazioni di parole che siano utili per predire le parole circostanti, meglio chiamati contesti, in una frase o in un documento. Data una sequenza di parole di addestramento, la funzione obiettivo massimizza la probabilità media di log

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_T) \tag{2.9}$$

dove c è la dimensione del contesto di training [20].

Nell'implementazione del modello `word2vec`, la formulazione standard dello *skip-gram* definisce la precedente probabilità di log ricorrendo alla funzione Softmax:

$$p_\theta(w_O | w_I) = \frac{\exp({v'_{w_O}}^\top v_{w_I})}{\sum_{w=1}^{W} \exp({v'_w}^\top v_{w_I})} \tag{2.10}$$

dove v_w e v'_w sono le rappresentazione vettoriali di input e output e W è il numero delle parole del vocabolario [27].

In questo modo la previsione di una data parola a partire da un contesto risulta essere un compito computazionalmente intenso, poiché vi sono operazione che coinvolgono l'intero dizionario.

Di conseguenza, un alternativa alla funzione Softmax è l'applicazione della *Noise Contrastive estimation* con campionamento negativo [28, 29]. Essa consente un allenamento più veloce e rappresentazioni vettoriali migliori per le parole frequenti.

2.3.2 Algoritmi di ottimizzazione

Gli algoritmi di ottimizzazione sono necessari per minimizzare il risultato di una determinata funzione obiettivo, la quale dipende dai parametri che il modello deve imparare durante l'addestramento.

Vengono utilizzate varie strategie e algoritmi di ottimizzazione per aggiornare e calcolare i valori appropriati e ottimali di tale modello, i quali influenzano fortemente l'efficacia del processo di apprendimento.

L'entità dell'aggiornamento è determinata dal tasso di apprendimento η, in inglese *learning rate*, che garantisce la convergenza al minimo globale, per superfici di errore convesse, e ad un minimo locale, per superfici non convesse.

Stochastic Gradient Descent

La discesa del gradiente, in inglese *Gradient Descent* (GD), è un algoritmo iterativo per l'ottimizzazione di funzioni [30].

Viene utilizzato principalmente per eseguire gli aggiornamenti dei pesi in un modello di rete neurale nel seguente modo

$$\theta = \theta - \eta \nabla J(\theta) \tag{2.11}$$

dove η rappresenta il tasso di apprendimento, $\nabla J(\theta)$ è il gradiente della funzione di loss $J(\theta)$ rispetto al parametro θ.

La tradizionale discesa gradiente, o *Batch Gradient Descent* (GD), calcola il gradiente dell'intero set di dati, eseguendo un solo aggiornamento. Di conseguenza il processo di addestramento può risultare lento e difficile da controllare per i set di dati che sono molto grandi.

I problemi che si verificano con questo algoritmo vengono risolti applicando una sua variante: la discesa stocastica del gradiente, in inglese *Stochastic Gradient Descent* (SGD). Questa tecnica esegue un aggiornamento alla volta dei parametri per ognuno degli esempi

di training

$$\theta = \theta - \eta \nabla J(\theta; x(i); y(i)) \tag{2.12}$$

dove $x(i)$ e $y(i)$ sono le coppie di esempi usati per l'addestramento.

Questa tecnica risulta essere molto più veloce di quella classica. A causa dei frequenti aggiornamenti, i parametri presentano un'alta varianza, inoltre la funzione loss oscilla tra diverse intensità. Questo favorisce la scoperta di nuovi minimi locali, complicando però la convergenza all'ottimo globale.

Adagrad

Adagrad è un metodo di apprendimento adattativo che aggiusta il learning rate sulla base dei parametri [31] . In questo algoritmo, la dimensione degli aggiornamenti è grande per parametri associati a caratteristiche poco ricorrenti e piccola per quelli più frequenti. Per questo motivo viene considerato adatto alla gestione di dati sparsi.

Adagrad modifica il tasso di apprendimento generale η ad ogni istante di tempo t per ogni parametro $\theta(i)$, sulla base dei gradienti che sono stati calcolati per $\theta(i)$.

Il vantaggio principale di questo metodo è che non è necessario regolare manualmente la frequenza di apprendimento e nella maggior parte delle implementazioni viene usato un valore predefinito — per esempio 0,001 — e lasciato invariato.
La principale debolezza consiste nell'accumulo dei "gradienti quadrati" nel denominatore, finché ogni termine aggiunto è positivo [30]. Di conseguenza il tasso di apprendimento si riduce fino al punto in cui l'algoritmo non è più in grado di acquisire ulteriori conoscenze.

2.3.3 Paradigmi di apprendimento

Gli algoritmi di apprendimento sono principalmente suddivisi in due categorie:

supervisionato — alla rete viene presentato un training set preparato da un "insegnante esterno", composto da coppie significative di valori (input, output atteso).

Quando alla rete neurale viene fornito l'input dall'ambiente, l'insegnante calcola l'output desiderato corrispondente, addestrando la rete mediante un algoritmo (tipicamente quello di back propagation [32]).

La rete impara a riconoscere la relazione incognita che lega le variabili di ingresso e uscita, in modo da prevedere il valore di output per qualsiasi valore di ingresso, basandosi solo su una casistica di corrispondenze (coppie input-output).

non supervisionato — alla rete vengono presentati solo i valori di input, mentre non sono messe a disposizione le informazioni di ritorno dell'ambiente sui valori obiettivo che si vogliono ottenere in risposta o riguardo la correttezza dell'output fornito.

La rete è in grado di individuare da sola pattern, caratteristiche, similarità e regolarità statistiche nei dati di input, acquisendo la capacità di dividerli in cluster rappresentativi che sviluppino delle rappresentazioni interne, senza usare confronti con output noti.

In questo caso, gli algoritmi che modificano i pesi della rete fanno riferimento solo ai dati contenuti nelle variabili di ingresso.

Questo è un tipo di apprendimento autonomo senza controllo esterno sull'errore. È un approccio adatto per ottimizzare le risorse nel caso in cui non si conoscano a priori i gruppi in cui dividere l'input.

2.4 Train, Validation e Test Sets

Per misurare le prestazioni di una rete neurale dopo la fase di apprendimento, viene creato un test set formato da coppie non utilizzate per il training e validation set.
Vengono generalmente definiti:

- Training set — sul quale viene eseguito l'algoritmo di apprendimento.
- Validation set — viene utilizzato per regolare i parametri, selezionare le features e prendere decisioni per quanto riguarda l'algoritmo di apprendimento.
- Test set — si utilizza per valutare le performance dell'algoritmo, ma non per prendere decisioni su quale algoritmo di apprendimento o parametri utilizzare.

Una volta definiti i set, ci si concentrerà sul miglioramento delle prestazioni del training e validation set.

Generalmente la dimensione del test set è un terzo di quella del training. Esso è composto da input critici su cui la risposta della rete deve essere buona. Questo funziona bene quando sono messi a disposizione un numero limitato di esempi, ma nell'era dei Big Data, dove i problemi di apprendimento automatico consistono di più di un miliardo di campioni, la frazione di dati allocati agli insiemi di sviluppo e test è ridotta, nonostante il valore assoluto di esempi sia maggiore.

Vengono utilizzate diverse tecniche statistiche per valutare la bontà di un modello.

2.4.1 Evaluation metric

Le metriche di valutazione misurano le prestazioni di un modello, discriminando la bontà dei risultati ottenuti. Vengono presi in considerazione diversi tipi di metriche per valutare i modelli. La scelta della metrica dipende completamente dal tipo di modello e dal piano di implementazione.

I modelli predittivi si distinguo in due principali categorie: si parla di *regressione* quando l'output da prevedere è continuo, o di *classificazione* nel caso in cui l'output sia nominale o binario.

Regressione

L'scarto quadratico medio, in inglese *Root Mean Squared Error* (RMSE), è la metrica di valutazione più popolare utilizzata nei problemi di regressione. Questo parametro aiuta a fornire risultati affidabili, mostrando correttamente la grandezza del termine di errore. La metrica RMSE è definita dalla seguente equazione

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{N}}. \tag{2.13}$$

con $\hat{Y}$ quantità prevista dalla rete, Y i valori osservati e N numero delle previsioni.

Classificazione

Nei problemi di classificazione, in particolare quella binaria, gli output sono 0 o 1.
Una *matrice di confusione*, nota anche come matrice di errore, è una tabella 2×2, generalizzabile ad una $N \times N$ per problemi ad N classi, che consente la visualizzazione delle prestazioni di un algoritmo di apprendimento supervisionato.

Ogni colonna della matrice rappresenta le istanze previste di una classe mentre ciascuna riga quelle osservate.

L'individuazione di falsi positivi (casi negativi identificati come positivi), falsi negativi (casi positivi identificati come negativi), veri positivi (casi positivi correttamente identificati) e veri negativi (casi negativi correttamente identificati), mostrati nella figura 2.3, consentono un'analisi più dettagliata della semplice proporzione di classificazioni corrette.

È possibile estrarre da questa tabella le seguenti misure di performance:

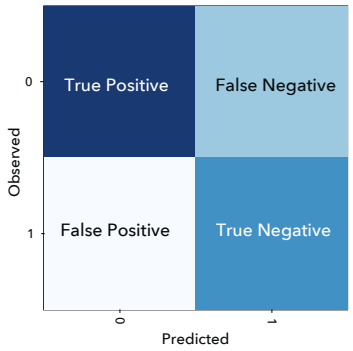


Figura 2.3: Visualizzazione di una matrice di confusione

- l'*accuracy*, o accuratezza, del modello, che consiste nella porzione rispetto al totale delle previsioni corrette

$$\frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}} \tag{2.14}$$

- la *precision*, o precisione, cioè la porzione dei casi positivi identificati correttamente

$$\frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2.15}$$

- la *recall*, ovvero la porzione dei casi positivi reali correttamente identificati

$$\frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2.16}$$

2.4.2 Overfitting

L'*overfitting* è "la produzione di un'analisi che corrisponde esattamente a un particolare insieme di dati o ne presenta forti similarità; può determinare l'impossibilità di adattamento a nuovi dati o compromettere l'affidabilità delle predizioni sulle osservazioni future".

La rete neurale deve avere la capacità di comprensione del modello statistico dei dati. In presenza di overfitting essa memorizza i dati del training set e non è quindi in grado di generalizzare su nuovi dati. L'essenza di questo problema consiste nell'estrarre inconsapevolmente parte della variazione residua (cioè il rumore) come se quella variazione rappresentasse la sottostante struttura del modello [33].

Per ridurre la possibilità di overfitting esistono diverse tecniche, come la convalida incrociata (Cross Validation), la regolarizzazione o l'*early stopping*, che consiste nell'utilizzo di un validation set di coppie non usate nel training set per la misurazione dell'errore.

Capitolo 3

Formulazione

3.1 Definizione del problema

Partendo da materiale testuale presente in rete, in particolare un dataset messo a disposizione da *Yelp Dataset Challenge* contenente 5 200 000 reviews relative a 174 000 business di 11 aree metropolitane nel mondo, l'obiettivo è quello di estrarre da questi dati caratteristiche di personalità.
Il nostro scopo è di identificare un adeguato spazio semantico che permetta di definire la personalità dell'oggetto target a cui un determinato testo si riferisce.

3.2 Descrizione del dataset

Come punto di partenza, viene messo a disposizione un dizionario di 637 aggettivi, che la letteratura psicologica definisce come marker dei cinque grandi tratti di personalità noti come *Big Five*. In particolare, questo vocabolario associa ad ogni aggettivo un vettore di cinque elementi in cui ogni elemento corrisponde al grado di presenza o assenza di una determinata caratteristica.

Adjective	**OCEAN**				
Active	0,053194	0,237406	0,365915	0,116700	-0,058669
Angry	-0,004604	-0,038453	0,020755	-0,294754	0,590114
Boring	-0,069877	-0,099754	-0,478821	-0,236462	0,118821
...	...	...	...	...	...

Tabella 3.1: Esempio di dizionario OCEAN

3.3 Calcolo della Ground Truth

Per poter addestrare correttamente il modello è necessario avere una Ground Truth, ovvero una label associata ad ogni input della rete. Senza questa informazione sarebbe impossibile ottimizzare il modello e valutarne la validità.

Si procede eliminando dal dataset tutte le sentences non contenenti almeno uno degli aggettivi presenti nel dizionario OCEAN. In seguito verranno utilizzati i dati contenuti all'interno del vocabolario per creare una mappatura diretta tra ogni frase e il corrispondente vettore di personalità, in cui ogni elemento sarà calcolato come la media del valore di ogni aggettivo presente nel testo.

3.4 Preprocessing

Gli algoritmi di apprendimento automatico non sono in grado funzionare direttamente con il testo non elaborato, è quindi necessario eseguire in primis un preprocessamento del database di testi e in seguito convertire i dati in numeri, nello specifico, in vettori di numeri.
Alla fine di questo processo il dataset ottenuto sarà suddiviso in tre corpora:

- 70 % training set : contenente 4 351 900 reviews utilizzate dalla rete per l'apprendimento;
- 10 % validation set: contenente 621 700 reviews;
- 20 % testing set: contenente 1 243 000 reviews.

La divisione dei tre dataset dovrà mantenere una buona distribuzione fra le diverse classi.

3.4.1 Natural Language Processing

Il *Natural Language Processing* (NLP) è un insieme di tecniche di computer science e linguistica che ricorrono a dei calcolatori per analizzare il linguaggio umano.

Dal punto di vista sintattico, al dataset viene applicata la seguente serie di operazioni:

- **Rottura della frase**: dato un pezzo di testo vengono trovati i limiti della frase, spesso contrassegnati da punti o altri segni di punteggiatura.
- **Stemming**: alcune parole vengono ridotte alla loro forma radice (ad esempio "argue, argued, argues, arguing, and argus" sono mappati alla parola "argu").

- **Segmentazione di parole**: un blocco di testo o sentence viene separato in parole. Per una lingua come l'inglese, questo è abbastanza banale, poiché le parole sono solitamente separate da spazi.

Dal punto di vista semantico invece si interviene nel seguente modo:

- **Semantica lessicale**: tenta di comprendere il significato computazionale delle singole parole nel loro contesto.
- **Comprensione del linguaggio naturale**: i blocchi di testo vengono convertiti in rappresentazioni più formali e più facili da manipolare per i computer.

Ricorrendo, dove possibile, alle tecniche sopraelencate, è possibile costruire un dizionario delle 60 000 parole più frequenti nel corpus di training, basato sulla frequenza assoluta di una parola, avendo cura di eliminare tutti gli aggettivi presenti nel dataset OCEAN.

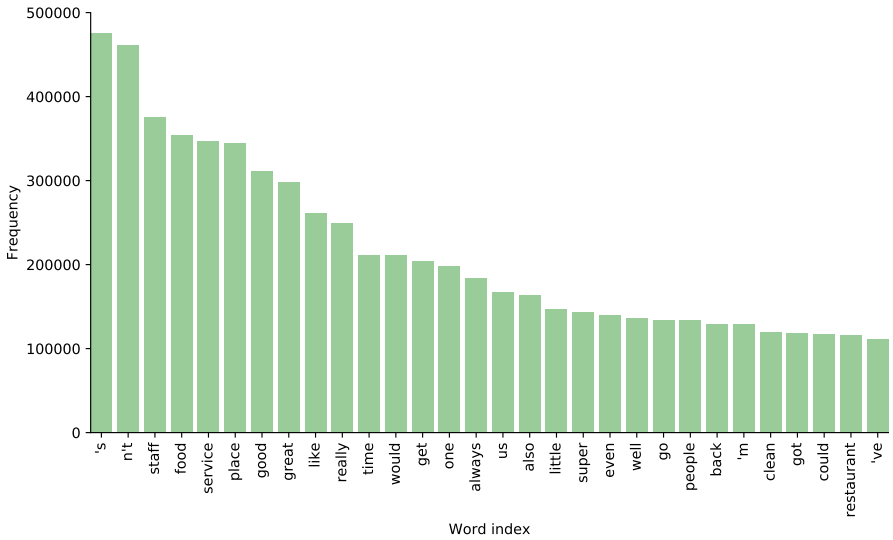


Figura 3.1: Istogramma rappresentativo delle 30 parole più frequenti nel dizionario

In questo modo non verrà influenzata la rete mostrando l'associazione tra gli aggettivi e le label associate, ed il modello sarà "costretto" ad imparare il legame esistente tra il contesto di una frase e il relativo valore del tratto di personalità.

Inoltre sarà necessario anche rimuovere tutte le *stopwords* relative alla lingua inglese, ovvero parole considerate poco significative perché usate troppo frequentemente all'interno delle frasi — per esempio gli articoli e le congiunzioni — filtrando i termini comuni e senza uno specifico significato semantico dalle parole che trasportano vere informazioni.

In seguito ogni parola del dizionario verrà codificata con un valore intero univoco, mentre quelle non presenti, tra cui gli aggettivi mappati nel vocabolario OCEAN, verranno indicizzati al valore “-1” e gli verrà associato il token “UNK” come mostrato nella figura 3.2

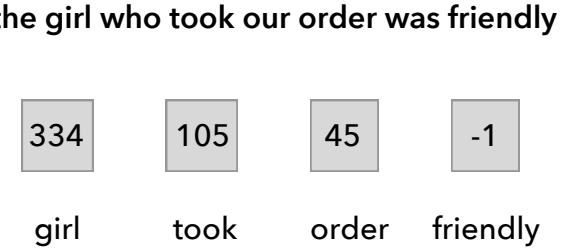


Figura 3.2: Visualizzazione del preprocessing

Capitolo 4

Esperimenti e risultati

La maggior parte degli attuali studi di previsione della personalità si sono concentrati sull'applicazione di tecniche generali di apprendimento automatico per predire i tratti di personalità Big Five. In particolare verranno utilizzate diverse strutture di rete, combinando differenti domini.

4.1 Spazi di rappresentazione

Per affrontare questo problema di *Text Mining*, gli esperimenti si concentrano su due principali metodi per l'estrazione delle caratteristiche del testo:

- Un approccio supervisionato, in cui viene utilizzato come strumento di generazione di feature un vettore *bag-of-words* o brevemente BoW, una rappresentazione di testo che descrive la presenza delle parole all'interno di un documento, in questo caso nel dizionario delle occorrenze [21].
- Un approccio non supervisionato, in cui viene costruito un embedding, tramite l'algoritmo `word2vec` di Tomas Mikolov [20]. Insegnando alla rete il significato delle parole e la relazione tra di esse, è possibile rappresentare, sotto forma di vettori, le mappature tra le parole e i contesti.

In seguito viene posta l'attenzione su tre diverse architetture neurali:

- Reti fully-connected;
- Reti neurali convoluzionali CNN;
- Classificatori multi-label binari.

4.2 Esperimento 1

4.2.1 Input Features

Nel primo approccio proposto, per rappresentare i dati testuali viene utilizzato il modello *bag-of-words*, un tipo di descrizione semplificata, spesso utilizzata nell'elaborazione del linguaggio naturale e nel campo dell'*Information Retrivial* (IR).

Sfruttando il dizionario delle occorrenze precedentemente costruito, il testo viene modellato come fosse una "borsa di parole", in cui grammatica e ordine delle parole vengono trascurate. Ogni frase viene ridotta ad un vettore in cui ogni elemento identifica una parola del dizionario. Nella posizione corrispondente ad un determinato termine vi sarà il valore 1 se la parola è contenuta nella frase, 0 altrimenti.

Il modello riguarda solo le parole conosciute, di conseguenza i vocaboli che compaiono nel testo ma sono assenti nel dizionario vengono trascurati.

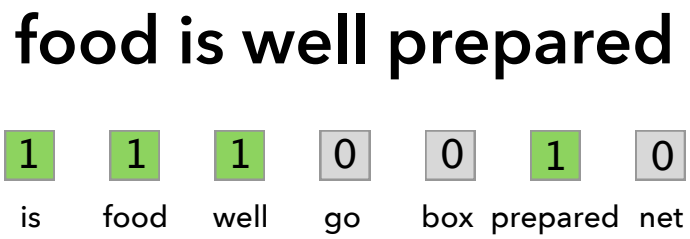


Figura 4.1: Visualizzazione del modello *bag-of-words*

4.2.2 Architettura della rete

Una volta estratte, le features posso essere passate in input alla rete neurale, che le elabora calcolando le risposte dei neuroni dal livello di input verso il livello di output.

In questa prima strategia viene utilizzata una rete *feed-forward* con struttura densa o *fully-connected*. Per i dettagli di queste architetture si rimanda alla sezione 2.2.1.

Ad ogni livello viene applicata la funzione di attivazione non lineare $ReLU$, descritta nella sezione 2.1.1.

Inoltre, per accelerare l'apprendimento ed aumentare la stabilità della rete, viene effettuata dopo ogni layer una *Batch Normalization*, definita nella sezione 2.2.4.

Vengono presentate nella tabella 4.1 tre architetture implementate, con differenti strati e numero di neuroni che caratterizza ciascuno.

Layer	Modello 1	Modello 2	Modello 3
Input	60 000	60 000	60 000
fc1	300	300	100
fc2	200	200	50
fc3	-	100	20
Output	5	5	5

Tabella 4.1: Confronto delle architetture di tre differenti modelli

Tutte le simulazioni sono state addestrate per 10 epoche: la fase di addestramento viene in genere portata avanti fino a quando le performance sul test non producono alcun miglioramento.

L'ottimizzatore scelto è *Adagrad*, introdotto nella sezione 2.3.2, con learning rate 0,001.

Come funzione di loss è stato scelto l'errore quadratico medio, in inglese *Mean Squared Error (MSE)*, presentato nella sezione 2.3.1. Mentre la metrica di valutazione utilizzata per misurare le prestazioni predittive del modello è la *Root Mean Squared Error* (RMSE), introdotta nella sezione 2.4.1.

In ogni epoca si alternano una fase di training ed una fase di test in modo tale da monitorare costantemente i miglioramenti o i peggioramenti del modello sul test set.

4.2.3 Performance

Prendendo in considerazione le tre diverse architetture implementate, vengono presentati i risultati ottenuti in termini di loss.

	Train loss	Test loss	Tempo di training
Modello 1	0,061	0,062	235 min
Modello 2	0,090	0,061	250 min
Modello 3	0,068	0,062	265 min

Tabella 4.2: Confronto dei risultati in termini di loss ottenuti dalle tre diverse reti

Per valutare l'efficacia di questi modelli, è fondamentale eseguire un analisi dettagliata, in particolare ponendo l'attenzione sui valori di RMSE per ogni tratto di personalità. Calcolando il valore medio assunto da ogni caratteristica durante la fase di addestramento, è possibile stabilire qual è il valore di Root Mean Squared Error di un modello concettuale, chiamato "Modello 0", che per ogni tratto predice sempre il suo valore medio.

Modelli		Root Mean Squared Error				
		O	C	E	A	N
Modello 1	Modello	0,148	0,227	0,224	0,251	0,351
	Modello 0	0,145	0,224	0,213	0,218	0,318
Modello 2	Modello	0,147	0,226	0,225	0,251	0,341
	Modello 0	0,141	0,227	0,213	0,208	0,305
Modello 3	Modello	0,147	0,226	0,225	0,262	0,348
	Modello 0	0,233	0,307	0,262	0,373	0,546

Tabella 4.3: Confronto dei risultati in termini di Root Mean Squared Error delle architetture contro il modello basato sulla media di training

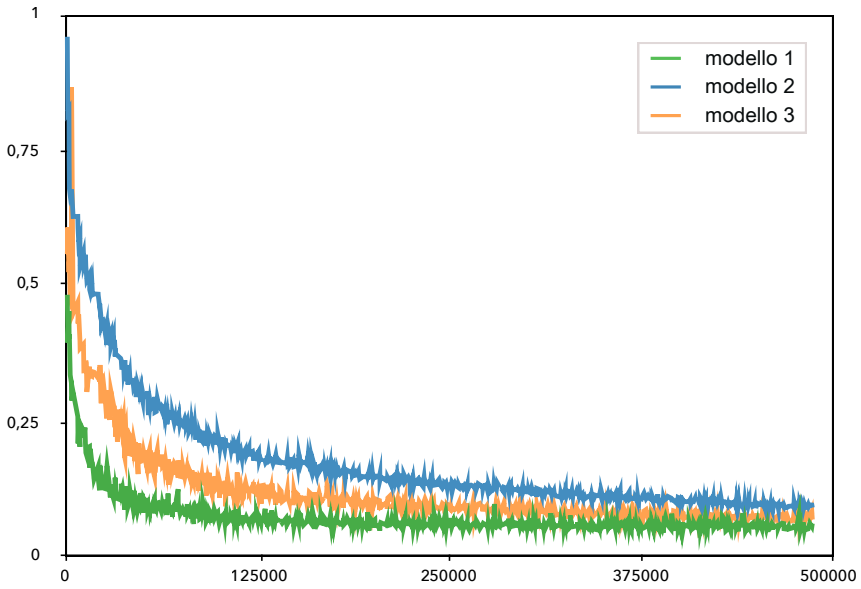


Figura 4.2: Visualizzazione delle training loss dei tre modelli

Mettendo a confronto le due metriche, si nota che i modelli realmente implementati imparano nella maggioranza dei casi a predire un valore con lo stesso errore commesso dai modelli banali che calcolano la media. Risulta allora evidente che i risultati ottenuti non siano ottimali.

Nonostante ciò, il terzo modello sembrerebbe essere leggermente migliore degli altri due, in particolare confrontandolo con il corrispondente modello nullo si nota la presenza di un piccolo margine di miglioramento.

La scarsa efficienza di questi modelli potrebbe dipendere dall'efficacia con cui vengono codificate le feature in input alla rete. Infatti le limitazioni dell'approccio bag-of-words

derivano in parte dalla progettazione del vocabolario e della sua dimensione, che può causare una scarsa "descrizione" del documento. Scartare l'ordine delle parole e ignorare il contesto non consente di determinare la differenza tra le stesse parole disposte diversamente, i sinonimi ecc. Inoltre, un tipo di rappresentazione sparsa risulta più difficile da modellare quando si cercano modelli in grado di sfruttare poche informazioni in uno spazio rappresentativo ampio, sia per ragioni computazionali (spazio e complessità temporale) sia per ragioni di informazione.

4.3 Esperimento 2

Durante l'applicazione di tecniche di apprendimento automatico, tutte le "informazioni" vengono rappresentate per mezzo di identificativi unici e discreti.

Nel caso dell'approccio BoW, la codifica utilizzata non fornisce alcuna informazione utile al sistema riguardo le relazioni che possono sussistere tra i singoli elementi. Ciò significa che quando sta elaborando i dati, il modello può sfruttare molto poco di ciò che ha appreso su un determinato termine.

Inoltre la "raffigurazione" utilizzata nel precedente esperimento, ha portato alla creazione di dati sparsi. Di conseguenza, per ottenere un modello di successo, un'alternativa valida sarebbe quella di sfruttare modelli spaziali vettoriali, in inglese *Vector Space Model* (VSM), per rappresentare le parole in uno spazio continuo [34, 35]. Questi metodi dipendono dall'ipotesi distributiva, la quale afferma che le parole che appaiono negli stessi contesti condividono lo stesso significato semantico [36].

4.3.1 Input Features

Nel secondo approccio, viene applicato l'algoritmo non supervisionato `word2vec` di Tomas Mikolov [37]. `Word2vec` è un modello predittivo particolarmente efficiente dal punto di vista computazionale per l'apprendimento degli embedding di parole a partire dal testo non elaborato. Esso è basato su una rete neurale artificiale a due strati, addestrati a ricostruire i contesti linguistici delle parole.

A partire dal corpus di testo, la rete prende in input un set formato dall'accoppiamento di ogni parola target e i contesti in cui appare e restituisce un insieme di vettori che rappresentano la distribuzione semantica delle parole nel testo.

Viene considerato come "contesto" l'insieme delle "parole a sinistra" e delle "parole alla destra" dell'obiettivo, ovvero la finestra di dimensione 1 attorno all'elemento target. Ogni coppia di destinazione del contesto viene trattata come se fosse una nuova osservazione, incrementando le informazioni distribuzionali. Viene così prodotto uno spazio vettoriale di

diverse centinaia di dimensioni, in cui ogni parola univoca viene assegnata a un vettore corrispondente nello spazio.

Per l'implementazione viene utilizzato il modello *skip-gram*, una versione di `word2vec` che vuole predire le parole del contesto di origine (label) a partire dalle parole target (features).

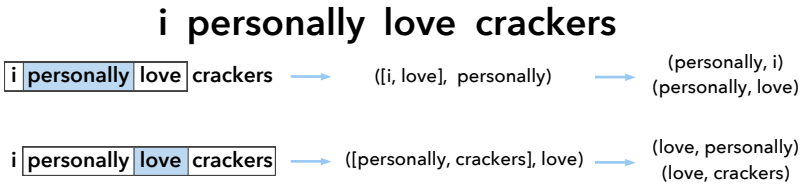
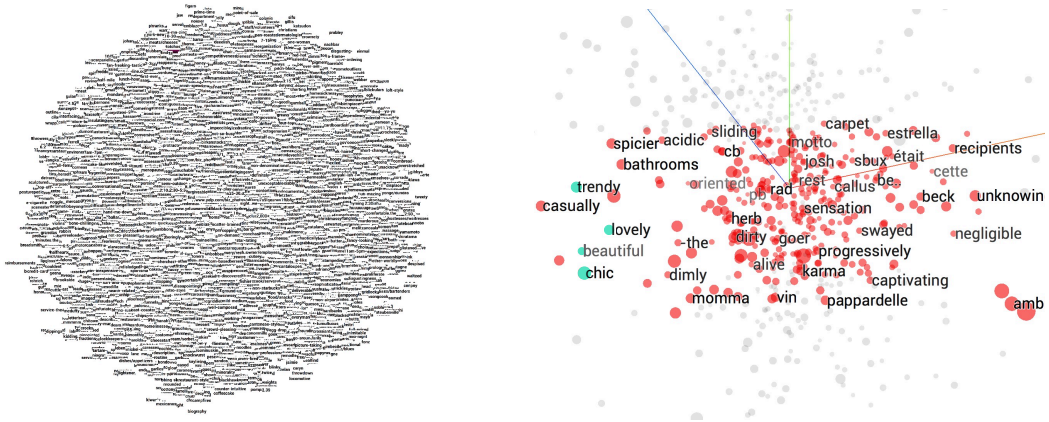


Figura 4.3: Visualizzazione del modello skip-gram

In questo tipo di apprendimento, i vettori si posizionano nello spazio in modo tale che le parole che condividono contesti comuni nel corpo siano situate in stretta prossimità l'una dell'altra. Un esempio interessante viene illustrato nella figura 4.4(b) — si ponga particolare attenzione alle parole "beautiful", "lovely", "chic", "trendy" ecc.. — in cui i vocaboli semanticamente simili si trovano vicini dello spazio.



(a) *Visualizzazione 2D* (b) *Dettaglio di una visualizzazione 3D*

Figura 4.4: Proiezione dell'embedding di Mikolov nello spazio 2-3 dimensionale, tramite la *tecnica di riduzione della dimensionalità* (t-SNE) [38]

La funzione obiettivo utilizzata dalla rete per la costruzione dell'embedding viene definita sull'intero set di dati, ed ottimizzata con la *Stochastic Gradient Descent* (SGD), definita nella sezione 2.3.2.

Nella seguente tabella vengono presentati i due embedding realizzati e i relativi parametri.

Embedding	Parameters	
	Embedding Size	Num Sampled
Embedding 1	40	20
Embedding 2	250	50

Tabella 4.4: Confronto dei parametri della rete impostati per la realizzazione dell'embedding

4.3.2 Architettura della rete

La rappresentazione spazio-vettoriale viene utilizzata come feature della modello che si andrà a costruire. In questo secondo approccio verrà utilizzata un *rete convoluzionale*, in cui ogni neurone è collegato solo a pochi neuroni vicini nel livello precedente, e lo stesso insieme di pesi viene utilizzato per ogni neurone.

In questo tipo di rete, il modello di connessione locale e lo schema di peso condiviso possono essere interpretati come un filtro (o un insieme di filtri) che accettano un sottoinsieme dei dati di input alla volta, ma vengono applicati all'intero input.

Uno strato convoluzionale è molto più specializzato ed efficiente di uno completamente connesso.

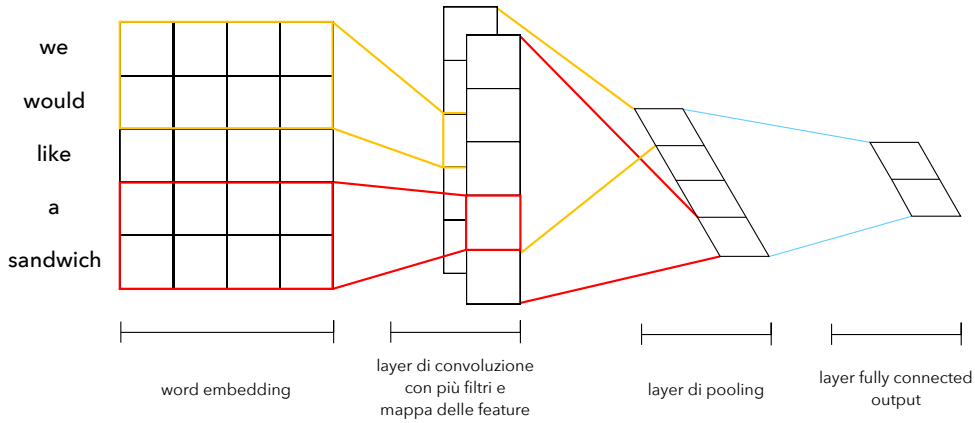


Figura 4.5: Esempio di architettura convoluzionale per la manipolazione del linguaggio naturale

Le operazioni eseguite da questi strati vengono trasformate in "moltiplicazioni" non lineari tramite l'applicazione della funzione di attivazione $ReLU$, introdotta nella sezione 2.1.1.

Ad ogni livello convoluzionale viene applicata una *Batch Normalization*, definita nella sezione 2.2.4, utilizzata sull'input per il ridimensionamento delle funzionalità e la normalizzazione batch nei livelli nascosti.

Dopo il primo strato convoluzionale viene inserito un *layer di pooling*, introdotto nella sezione 2.2.3, necessario per ridurre in modo efficace i campioni dell'output del livello precedente, riducendo il numero di operazioni richieste per tutti i livelli successivi, ma passando comunque le informazioni valide.

Come ultimo strato della rete viene scelto un *layer fully-connected*, definito nella sezione 2.2.1, corrispondente ad un'operazione lineare sul vettore di input del livello che esegue una serie di trasformazioni sulla rappresentazione profonda al fine di emettere i punteggi di ogni classe.

Layer	**Modello 4**	**Modello 5**	**Modello 6**
conv1	10×5, 10	5×5, 150, same pad	3×3, 100, same pad
mpool1		4×4, stride 2, same pad	
conv2	18×18, 10	5×20, 100, same pad	3×20, 75, same pad
conv3	—	1×20, 50,	1×20, 50
fcout		1×1, 5	

Tabella 4.5: Architetture implementate a partire dall'embedding 1

Layer	**Modello 7**	**Modello 8**
conv1	3×3, 100, stride 2, same pad	—
mpool1	4×4, stride 2, same pad	—
conv2	3×63, 75, stride 2, same pad	—
conv3	1×32, 50, stride 2	—
fc1	—	1×1, 100
fc2	1×32, 50, stride 2	1×1, 50
fc3	1×32, 50, stride 2	1×1, 20
fcout	1×1, 5	1×1, 5

Tabella 4.6: Architetture implementate a partire dall'embedding 2

Vengono presentate nelle tabelle 4.5 e 4.6 le diverse configurazioni dei layer convoluzionali delle architetture implementate per i due embedding.

L'input della rete è costituito da un numero di caratteristiche pari a $n \times p$ dove n è la dimensione dell'embedding e p il numero delle parole di ogni sentence.

Modelli		**Learning Rate**
Embedding 1	**Modello 4**	0,001 0
	Modello 5	0,000 1
	Modello 6	0,005 0
Embedding 2	**Modello 7**	0,005 0
	Modello 8	0,000 1

Tabella 4.7: Learning rate delle simulazioni effettuate nell'esperimento 2

Come algoritmo di ottimizzazione viene scelto sempre *Adagrad*, illustrato nella sezione 2.3.2, mentre il valore di *learning rate* settato per ogni modello viene mostrato nella tabella 4.7.

Viene mantenuta anche in questa simulazione la funzione di costo MSE, approfondita nella sezione 2.3.1.

Nel caso del secondo embedding si è voluto provare anche un modello che sfruttasse solo livelli di rete *fully-connected*, senza applicare alcuna convoluzione.

4.3.3 Performance

Prendendo in considerazione i due diversi embedding, vengono messe a confronto a confronto le diverse architetture implementate, mostrando i risultati ottenuti in termini di loss.

Modelli		**Train loss**	**Test loss**	**Tempo di training**
Embedding 1	**Modello 4**	0,061	0,058	200 min
	Modello 5	0,052	0,060	310 min
	Modello 6	0,042	0,060	540 min
Embedding 2	**Modello 7**	0,038	0,057	225 min
	Modello 8	0,058	0,117	250 min

Tabella 4.8: Confronto dei risultati in termini di *loss* ottenuti nelle diverse reti con i due diversi embedding

Valutando le prestazioni su train e test set viene rilevato il "Modello 7" come il migliore tra quelli proposti, presentando i valori di loss più bassi. Oltre a ciò, anche rispetto al precedente approccio il "Modello 7" mostra una performance migliorata circa del 3 %.

Vengono analizzati anche i valori di RMSE, relativi ad ogni tratto di personalità, riassunti nella tabella 4.9, e viene inoltre visualizzata la curva RMSE nella figura 4.6.

Modelli		Root Mean Squared Error				
		O	C	E	A	N
Embedding 1	**Modello 4**	0,148	0,226	0,232	0,252	0,313
	Modello 5	0,146	0,227	0,230	0,251	0,336
	Modello 6	0,146	0,225	0,224	0,251	0,337
Embedding 2	**Modello 7**	0,147	0,223	0,222	0,251	0,320
	Modello 8	0,399	0,275	0,257	0,271	0,457

Tabella 4.9: Confronto dei risultati in termini di Root Mean Squared Error delle architetture sul test set

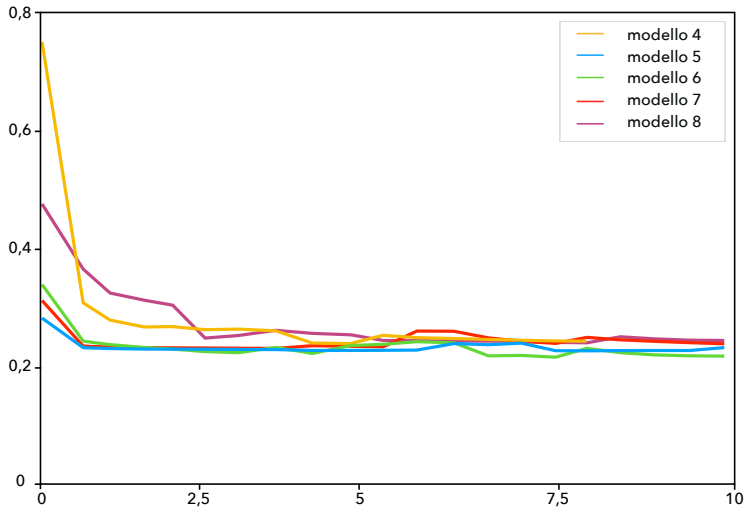


Figura 4.6: Visualizzazione delle training RMSE dei modelli

4.4 Esperimento 3

Nel terzo approccio si decide di valutare una rappresentazione dell'input alternativa.

4.4.1 Input Features

Ricorrendo nuovamente al modello *skip-gram*, l'input della rete questa volta comprende il set formato dall'accoppiamento tra gli aggettivi contenuti nel dizionario OCEAN e i loro contesti.

La finestratura che si andrà a definire in torno al target sarà di dimensione 2 e considererà le due parole a sinistra e le due parole a destra dell'aggettivo. La window non sarà più

quindi incentrata su ogni elemento della sentence. Dunque nell'embedding che si andrà a costruire non verranno appresi i contesti di tutte le parole, ma solamente quelli di nostro interesse, ovvero relativi agli aggettivi.

In questo esperimento viene realizzato un solo embedding, la cui dimensione è 250 e con numero di etichette negative da campionare pari a 50 [28].

4.4.2 Architettura della rete

Viene utilizzato l'embedding estratto per costruire due modelli di reti neurali, le cui architetture vengono presentate nella tabella 4.10. Come nel precedente esperimento si ricorrerà alle *reti convoluzionali*; si rimanda alla sezione 4.3.2 per i dettagli.

Layer	**Modello 9**	**Modello 10**
conv1	7×5, 100, stride 2, same pad	3×3, 100, stride 2, same pad
mpool1	4×4, stride 2, same pad	
conv2	5×63, 75, stride 2, same pad	3×63, 75, stride 2, same pad
conv3	3×32, 50, stride 2, same pad	1×32, 50, stride 2
conv4	1×16, 25, stride 2	—
fcout	1×1, 5	

Tabella 4.10: Architetture dei due modelli implementati a partire dall'embedding 3

L'algoritmo di apprendimento utilizzato è *Adagrad*, definito nella sezione 2.3.2, con *learning rate* 0,000 5 per il "Modello 9" e 0,005 per il "Modello 10". La funzione di *loss* scelta è la MSE, introdotta nella sezione 2.3.1.

4.4.3 Performance

Vengono presentati nella tabella 4.11 i valori ottenuti dalla funzione obiettivo dei due modelli, mentre nella tabella 4.12 vengono riassunti i valori di RMSE per ogni tratto di personalità.

	Train loss	**Test loss**	**Tempo di training**
Modello 9	0,043	0,060	18 h
Modello 10	0,050	0,059	17 h

Tabella 4.11: Confronto dei risultati in termini di loss ottenuti nell'implementazione dei due modelli

Modelli	Root Mean Squared Error				
	O	C	E	A	N
Modello 9	0,146	0,223	0,223	0,251	0,331
Modello 10	0,147	0,225	0,224	0,252	0,339

Tabella 4.12: Confronto dei risultati in termini di RMSE delle due architetture sul test set

Rispetto al precedente approccio non vengono ottenuti dei miglioramenti.

4.5 Esperimento 4

L'ultimo approccio provato riutilizza i precedenti metodi di estrazione di feature che hanno ottenuto le migliori prestazioni per realizzare un modello predittivo di classificazione.

Nello specifico il problema viene trasformato in un compito di classificazione binaria multi-label, in cui si cercano di predire cinque diverse etichette per ogni istanza.

Per ogni dimensione di personalità l'output della rete sarà "0", se il valore reale che assume un determinato tratto è inferiore a 0, o "1" altrimenti.
Viene stabilito 0 come "punto di neutralità assoluta" perché corrisponde alla media dei valori osservati in ognuna delle caratteristiche.

Adottando questo metodo, si assumerà di poter estrarre da questa funzione una misura di polarità, positiva o negativa, che indicherà un tratto più o meno accentuato.

Il vantaggio effettivo di questo approccio consiste nella possibilità di valutare le performance in modo standard, visualizzando le matrici di confusione e misurandone l'accuratezza, argomenti trattati nella sezione 2.4.1.

4.5.1 Input Features

Vengono utilizzati come ingresso della rete due embedding di Mikolov realizzati negli esperimenti precedenti, ovvero l'"Embedding 2" e l'"Embedding 3".

4.5.2 Architettura della rete

Le reti che vengono realizzate sono molto simili a quelle precedentemente implementate; i dettagli delle architetture vengono presentati nella tabella 4.13.

Ad ogni livello della rete viene applicata la stessa procedura degli esperimenti precedenti.

Layer	Modello 11	Modello 12	Modello 13
conv1	3×3, 100	5×3, 100	7×5, 100
mpool1	4×4	4×4	4×4
conv2	3×63, 75	3×63, 75	5×63, 75
conv3	1×32, 50	1×32, 50	3×32, 50
conv4	—	1×16, 25	1×16, 25
fcout	1×1, 10	1×1, 10	1×1, 10

Tabella 4.13: Architetture implementate a partire dall'embedding 2

In ogni livello convoluzionale della rete si utilizzano uno stride di dimensione 2 e padding same, tranne che per l'ultimo livello convoluzionale con padding valid, per i dettagli si consiglia di visitare la sezione 2.2.2.

Layer	Modello 14	Modello 15
conv1	7×5, 100, stride 2, same pad	7×5, 100, stride 2, same pad
mpool1	4×4, stride 2, same pad	4×4, stride 2, same pad
conv2	5×63, 75, stride 2, same pad	5×63, 75, stride 2, same pad
conv3	3×32, 50, stride 2, same pad	3×32, 50, stride 2, same pad
conv4	1×16, 25, stride 2	3×16, 25, stride 2, same pad
conv5	—	1×8, 16, stride 2
fcout	1×1, 10	

Tabella 4.14: Architetture implementate a partire dall'embedding 3

In tutte le simulazioni l'ottimizzatore scelto è *Adagrad*, introdotto nella sezione 2.3.2, il valore di *learning rate* di ogni modello viene mostrato nella tabella 4.15.

Come funzione obiettivo viene utilizzata la *Softamax Cross Entropy*, presentata nella sezione 2.3.1.

Modelli		Learning Rate
Embedding 2	**Modello 11**	0,000 1
	Modello 12	0,000 5
	Modello 13	0,000 5
Embedding 3	**Modello 14**	0,000 5
	Modello 15	0,000 5

Tabella 4.15: Learning rate delle simulazioni effettuate nell'esperimento 2

4.5.3 Performance

Prendendo in considerazione le diverse architetture implementate, vengono presentati i risultati ottenuti in termini di loss.

Modelli		**Train loss**	**Test loss**	**Tempo di training**
Embedding 2	**Modello 11**	3,275	3,455	20 h
	Modello 12	3,297	3,459	30 h
	Modello 13	3,444	3,458	40 h
Embedding 2	**Modello 14**	3,350	3,459	20 h
	Modello 15	3,230	3,454	60 h

Tabella 4.16: Confronto delle *loss* ottenute nelle diverse reti con i due diversi embedding

Come già detto, l'utilizzo di questo approccio consente una valutazione non più solo in termini di loss ma se ne potrà analizzare anche l'accuratezza, introdotta nella sezione 2.4.1.

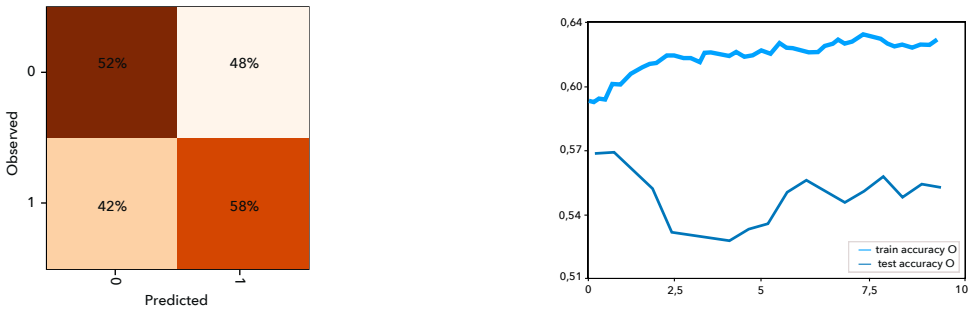
Modelli		**Train/Test Accuracy [%]**				
		O	C	E	A	N
Embedding 2	**Modello 11**	61/61	60/59	63/60	58/56	57/54
	Modello 12	62/59	61/50	64/45	61/56	61/56
	Modello 13	63/55	63/60	63/61	63/58	62/59
Embedding 3	**Modello 14**	61/61	61/57	62/52	60/57	60/56
	Modello 15	63/60	62/48	64/60	62/49	62/48

Tabella 4.17: Confronto delle accuracy sul test set delle diverse architetture

Il "Modello 13" sembrerebbe il migliore tra quelli proposti poiché presenta valori di accuracy più alti e la differenza di prestazioni tra train e test set è una tra le più basse, dimostrando minore propensione all'overfitting.

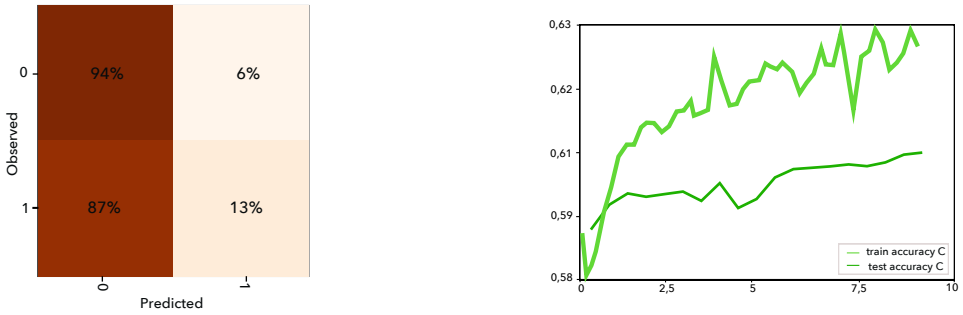
Viene presentata un analisi più dettagliata su questo modello, in particolare mostrando il grafico dell'accuratezza, su train e test set, e le matrici di confusione calcolate per ogni tratto di personalità.

Come si può vedere chiaramente dalle matrici di confusione, le caratteristiche *Conscientiousness* e *Extraversion* sono quelle che vengono predette nel modo peggiore.



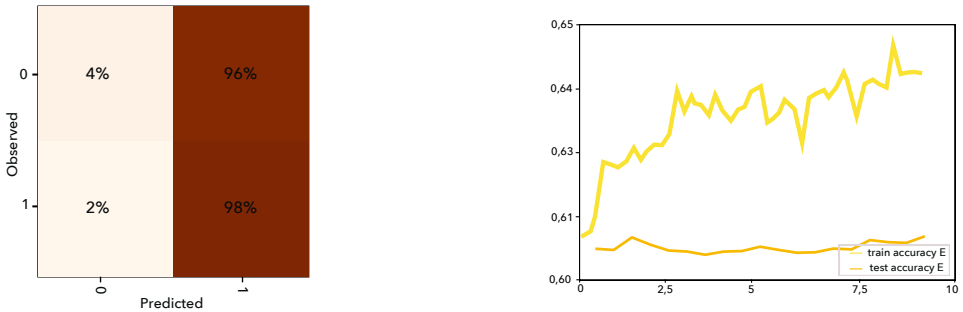
(a) *Visualizzazione matrice di confusione* (b) *Visualizzazione accuratezza su train e test*

Figura 4.7: Analisi della caratteristica Openness del “Modello 13”



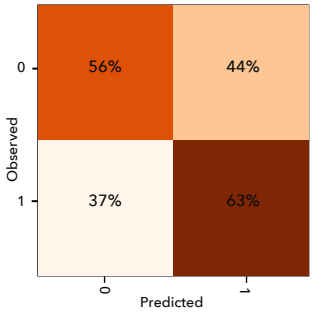
(a) *Visualizzazione matrice di confusione* (b) *Visualizzazione accuratezza su train e test*

Figura 4.8: Analisi della caratteristica Conscientiousness del “Modello 13”

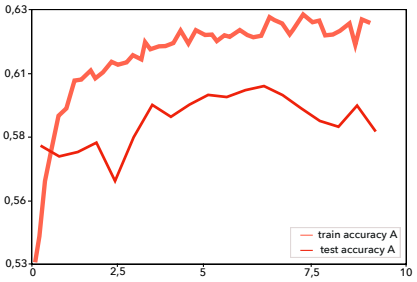


(a) *Visualizzazione matrice di confusione* (b) *Visualizzazione accuratezza su train e test*

Figura 4.9: Analisi della caratteristica Extraversion del “Modello 13”

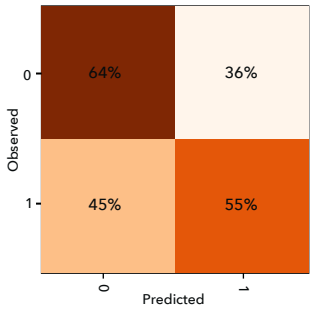


(a) *Visualizzazione matrice di confusione*

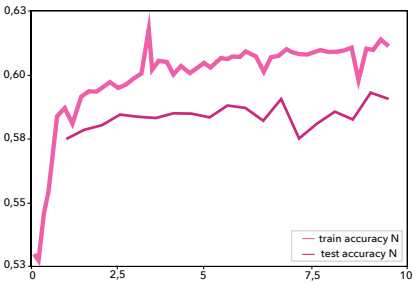


(b) *Visualizzazione accuratezza su train e test*

Figura 4.10: Analisi della caratteristica Agreeableness del “Modello 13”



(a) *Visualizzazione matrice di confusione*



(b) *Visualizzazione accuratezza su train e test*

Figura 4.11: Analisi della caratteristica Neuroticism del “Modello 13”

Capitolo 5

Conclusioni

La natura di questo progetto di tesi è altamente sperimentale ed è volta a presentare analisi dettagliate sull'argomento, in quanto allo stato attuale non esistono importanti indagini che affrontino il problema dell'apprendimento dei tratti di personalità a partire da testo in linguaggio naturale.

La domanda fondamentale che viene posta al fine di risolvere questo compito è quale sia la metodologia adatta alla rappresentazione del testo. Durante la sperimentazione, infatti, sono stati esaminati due principali approcci e ne è stata comparata l'efficacia.

Siamo partiti da una rappresentazione semplificata del testo ricorrendo al modello *bag-of-words*. I risultati ottenuti sono sub-ottimali dal punto di vista della complessità. Inoltre la codifica utilizzata non fornisce alcuna informazione utile al sistema riguardo le relazioni che possono sussistere tra le parole di una frase.

In seguito è stata sfruttata una classe di algoritmi distribuzionali per insegnare alla rete il significato e le relazioni sussistenti tra le parole. Sfruttando la versione *skip-gram* dell'algoritmo `word2vec` di Tomas Mikolov vengono rappresentate sotto forma di vettori le mappature tra parole e contesti nello spazio. Ricorrendo al secondo metodo i risultati ottenuti dimostrano come utilizzare un embedding sia la tecnica di estrazione di features più efficiente per filtrare le informazioni contenute in un testo.

È emerso che un'evoluzione di questo progetto potrebbe affacciarsi alla valutazione di rappresentazioni alternative del testo, annotazioni, part-of-speech [39] e altre tecniche di NLP, per approfondire e migliorare ulteriormente l'indagine.

Dal punto di vista delle architetture implementate, abbiamo iniziato dall'implementazione di reti neurali *fully-connected* come base per capire come modelli semplici di Deep Learning possano fornire informazioni sulle caratteristiche nascoste della personalità.

Infine, ci addentriamo nelle reti neurali *convoluzionali* molto più specializzate e efficiente delle precedenti nell'ambito del Text Mining.

Sviluppi futuri di questo lavoro potrebbero valutare una procedura di apprendimento alternativa, sfruttando altri modelli, quali le reti ricorrenti, per ottenere dei risultati più efficaci.

La prima valutazione effettuata è definita come una regressione che tenta di prevedere l'esatto valore reale per ogni tratto di personalità. Il problema presentato è estremamente complesso, e le performance ottenute sono ancora lontane da quelle desiderate. Per questo motivo viene eseguita una seconda valutazione che trasforma il nostro compito in un problema di classificazione binaria multi-label. I risultati raggiunti questa volta, evincono che ottimizzare la funzione di loss di un modello predittivo di regressione è molto più difficoltoso rispetto ad ottimizzare una funzione di costo stabile come la Softmax. Questo è evidente poiché il modello di regressione tenta di produrre un esatto valore per ogni input, e i valori anomali predetti possono introdurre gradienti enormi.

Bibliografia

[1] Arthur L Samuel. «Some studies in machine learning using the game of checkers». In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229 (cit. alle pp. viii, 5).

[2] Soumen Chakrabarti et al. «Data mining curriculum: A proposal (Version 1.0)». In: *Intensive Working Group of ACM SIGKDD Curriculum Committee* 140 (2006) (cit. a p. viii).

[3] James Franklin. «The elements of statistical learning: data mining, inference and prediction». In: *The Mathematical Intelligencer* 27.2 (2005), pp. 83–85 (cit. a p. viii).

[4] Ah-Hwee Tan et al. «Text mining: The state of the art and the challenges». In: *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*. Vol. 8. sn. 1999, pp. 65–70 (cit. a p. viii).

[5] Murray R Barrick e Michael K Mount. «The big five personality dimensions and job performance: a meta-analysis». In: *Personnel psychology* 44.1 (1991), pp. 1–26 (cit. a p. viii).

[6] Philip J Corr e Gerald Matthews. *The Cambridge handbook of personality psychology*. Cambridge University Press Cambridge, 2009 (cit. a p. 1).

[7] Benjamin J Sadock, Virginia A Sadock e Pedro Ruiz. *Comprehensive textbook of psychiatry*. lippincott Williams & wilkins Philadelphia, 2000 (cit. a p. 1).

[8] Lewis R Goldberg. «The structure of phenotypic personality traits.» In: *American psychologist* 48.1 (1993), p. 26 (cit. a p. 1).

[9] Paul T Costa e Robert R McCrae. «The revised neo personality inventory (neo-pi-r)». In: *The SAGE handbook of personality theory and assessment* 2.2 (2008), pp. 179–198 (cit. a p. 1).

[10] Harry C Triandis e Eunkook M Suh. «Cultural influences on personality». In: *Annual review of psychology* 53.1 (2002), pp. 133–160 (cit. a p. 3).

[11] Lisa M Saulsman e Andrew C Page. «The five-factor model and personality disorder empirical literature: A meta-analytic review». In: *Clinical psychology review* 23.8 (2004), pp. 1055–1085 (cit. a p. 3).

[12] Vinod Nair e Geoffrey E Hinton. «Rectified linear units improve restricted boltzmann machines». In: *Proceedings of the 27th International Conference on machine learning (ICML-10)*. 2010, pp. 807–814 (cit. a p. 6).

[13] Richard Hahnloser et al. «Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit». In: 405 (lug. 2000), pp. 947–51 (cit. a p. 6).

[14] R. H. R. Hahnloser, H. S. Seung e J. J. Slotine. «Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks». In: *Neural Computation* 15.3 (2003), pp. 621–638. ISSN: 0899-7667. DOI: 10.1162/089976603321192103 (cit. a p. 6).

[15] Xavier Glorot, Antoine Bordes e Yoshua Bengio. «Deep sparse rectifier neural networks». In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 315–323 (cit. a p. 6).

[16] Daniel Svozil, Vladimir Kvasnicka e Jiri Pospichal. «Introduction to multi-layer feed-forward neural networks». In: *Chemometrics and intelligent laboratory systems* 39.1 (1997), pp. 43–62 (cit. a p. 7).

[17] Tara N Sainath et al. «Convolutional, long short-term memory, fully connected deep neural networks». In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4580–4584 (cit. a p. 7).

[18] Yoon Kim. «Convolutional neural networks for sentence classification». In: *arXiv preprint arXiv:1408.5882* (2014) (cit. a p. 8).

[19] Christopher D Manning, Christopher D Manning e Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999 (cit. a p. 8).

[20] Tomas Mikolov et al. «Distributed representations of words and phrases and their compositionality». In: *Advances in neural information processing systems*. 2013, pp. 3111–3119 (cit. alle pp. 8, 11, 21).

[21] Hanna M Wallach. «Topic modeling: beyond bag-of-words». In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 977–984 (cit. alle pp. 8, 21).

[22] Andrej Karpathy. «Cs231n convolutional neural networks for visual recognition». In: *Neural networks* 1 (2016) (cit. a p. 9).

[23] Sergey Ioffe e Christian Szegedy. «Batch normalization: Accelerating deep network training by reducing internal covariate shift». In: *arXiv preprint arXiv:1502.03167* (2015) (cit. a p. 9).

[24] Zhou Wang e Alan C Bovik. «Mean squared error: Love it or leave it? A new look at signal fidelity measures». In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117 (cit. a p. 10).

[25] Yichuan Tang. «Deep learning using linear support vector machines». In: *arXiv preprint arXiv:1306.0239* (2013) (cit. a p. 10).

[26] Weiyang Liu et al. «Large-Margin Softmax Loss for Convolutional Neural Networks.» In: *ICML*. 2016, pp. 507–516 (cit. a p. 11).

[27] Chris Dyer. «Notes on noise contrastive estimation and negative sampling». In: *arXiv preprint arXiv:1410.8251* (2014) (cit. a p. 12).

[28] Tongliang Liu e Dacheng Tao. «Classification with noisy labels by importance reweighting». In: *IEEE Transactions on pattern analysis and machine intelligence* 38.3 (2016), pp. 447–461 (cit. alle pp. 12, 31).

[29] Chockalingam Viswesvaran e Deniz S Ones. «Measurement error in "Big Five Factors" personality assessment: Reliability generalization across studies and measures». In: *Educational and Psychological Measurement* 60.2 (2000), pp. 224–235 (cit. a p. 12).

[30] Sebastian Ruder. «An overview of gradient descent optimization algorithms». In: *arXiv preprint arXiv:1609.04747* (2016) (cit. alle pp. 12, 13).

[31] John Duchi, Elad Hazan e Yoram Singer. «Adaptive subgradient methods for online learning and stochastic optimization». In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159 (cit. a p. 13).

[32] S-I Horikawa, Takeshi Furuhashi e Yoshiki Uchikawa. «On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm». In: *IEEE transactions on Neural Networks* 3.5 (1992), pp. 801–806 (cit. a p. 13).

[33] Kenneth P Burnham e David R Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media, 2003 (cit. a p. 16).

[34] Tomas Mikolov, Wen-tau Yih e Geoffrey Zweig. «Linguistic regularities in continuous space word representations». In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2013, pp. 746–751 (cit. a p. 25).

[35] Katrin Erk e Sebastian Padó. «A structured vector space model for word meaning in context». In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2008, pp. 897–906 (cit. a p. 25).

[36] Marco Baroni, Georgiana Dinu e Germán Kruszewski. «Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors». In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2014, pp. 238–247 (cit. a p. 25).

[37] Tomas Mikolov et al. «Efficient estimation of word representations in vector space». In: *arXiv preprint arXiv:1301.3781* (2013) (cit. a p. 25).

[38] Laurens van der Maaten e Geoffrey Hinton. «Visualizing data using t-SNE». In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605 (cit. a p. 26).

[39] Roger W Brown. «Linguistic determinism and the part of speech.» In: *The Journal of Abnormal and Social Psychology* 55.1 (1957), p. 1 (cit. a p. 37).