UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA
DIPARTIMENTO DI ECONOMIA, METODI QUANTITATIVI E STRATEGIE DI IMPRESA
CORSO DI LAUREA MAGISTRALE IN SCIENZE STATISTICHE ED ECONOMICHE

Analisi delle relazioni tra le pubblicazioni del DEMS Bicocca mediante tecniche di word embedding

Relatore: Prof. Matteo Maria Pelagatti
Co-relatore: Prof. Fabio Mercorio

Tesi di Laurea Magistrale di:
LUCA BASSANESE
MATRICOLA 794564

Anno Accademico 2018 - 2019

Indice

1

Introduzione

1.1 Introduzione

L'analisi dei dati testuali negli ultimi anni sta prendendo sempre più piede negli ultimi anni. Le fonti di dati testuali sono sempre state molteplici (libri, giornali, lettere, bollettini, etc..) ma ora esse stanno diventando sempre più digitali e liberamente accessibili a chiunque. Questi milioni di dati che contengono la manifestazione esplicita del pensiero umano non possono più passare in secondo piano, bensì necessitano analisi specifiche per permettere un'estrazione sistematica dell'informazione contenuta al loro interno.

In questo ambito si è sviluppato questo lavoro di tesi, con l'idea di sfruttare le tecniche più recenti per applicarle a dati su cui non erano mai state usate prima. In particolar modo il dataset analizzato è il corpus delle pubblicazioni del Dipartimento di Economia, Metodi quantitativi e Strategie d'impresa dell'Università degli Studi di Milano-Bicocca, con il fine di estrarre da esso le relazioni che intercorrono sia tra diverse pubblicazioni sia tra differenti autori.

Per raggiungere questo obiettivo sono state impiegate particolari tecniche di word embedding che consentono di rappresentare con vettori numerici dati testuali. Dato che il dataset

risulta essere di piccole dimensioni e le tecniche sfruttate si basano su reti neurali artificiali che richiedono un'elevata numerosità campionaria per essere efficaci, la domanda a cui vuole rispondere questo studio è la seguente: riusciranno le tecniche di word embedding a fornire una rappresentazione significative dei dati testuali analizzati?

1.2 Motivazioni

La motivazione principale che ha spinto per lo sviluppo di questa tesi è stato il desiderio di applicare nuove tecniche di word embedding ad un dataset su cui non si era ancora provato questo nuovo approccio. Questo studio vuole essere un punto di partenza per ulteriori sviluppi che sicuramente saranno permessi dalla creazione di nuovi algoritmi in un prossimo futuro.

Inoltre, si è voluto andare a testare in linea pratica le tecniche di word embedding in un ambito empirico particolarmente avverso alle loro caratteristiche. Tastare con mano i limiti e i punti di forza di queste tecniche è stata una motivazione forte che ha spinto per questo lavoro di tesi.

Infine, in questo studio è stato possibile avere un confronto diretto dato un input testuale tra ciò che una persona percepisce come informazione e ciò invece che un algoritmo apprende da esso. Questo confronto è sin da ora di grande interesse e sarà nei prossimi anni sempre più al centro dell'attenzione, con il continuo ed inesorabile sviluppo delle reti neurali.

2

Metodologia

2.1 Premesse

2.1.1 Workflow

Per orientarsi all'interno di questo lavoro di tesi viene ora definito il workflow seguito nello sviluppo della stessa in Figura 2.1 . Esso riporta tutti i processi eseguiti in questi mesi e gli input e gli output ottenuti da quest'ultimi, ognuno dei quali viene abbinato alla specifica collocazione all'interno della tesi.

In principio si è utilizzato il corpus, cioè il dataset di tutte le pubblicazioni da parte dei professori appartenenti al Dipartimento di Economia, Metodi quantitativi e Strategie d'impresa (DEMS). Durante la fase di preprocessing sono state affrontate tutte le varie problematiche relative ai dati testuali degli abstract, dei titoli e delle parole chiave presenti nel dataset. Più nel dettaglio sono state rimosse le varie ridondanze, sono stati selezionati i titoli in lingua inglese e su questi dati testuali è stato eseguito un processo di data cleaning, per rendere gli stessi uniformi e adatti ad essere forniti come input per i modelli. Dal momento che i modelli considerati richiedono input leggermente diversi, nella fase del preprosessing si

sono tenute conto di queste esigenze implementando, dopo un preprocessing generale, un preprocessing specifico per rendere i dati conformi ad ogni modello.

La fase successiva è stata la fase di generazione dei modelli, pur avendo strutture diverse, hanno tutti in comune lo stesso obiettivo: rappresentare delle frasi in uno spazio vettoriale fisso. Pertanto essi sono definiti come Vector Space Models. In particolar modo i modelli generati sono facenti parte ad una delle seguenti architetture diverse:

1. Doc2vec
2. Sent2vec
3. Universal sentence encoder
4. SBERT

Per ognuna di queste sovra-categorie sono stati sviluppati modelli differenti in base alla scelta di diversi parametri propri di ciascun modello. Per questa ragione, questa fase ha richiesto periodi di tempo computazionali particolarmente significativi.

Dopo aver generato i modelli è stato necessario porli sottovalutazione, fase non facile dal momento che l'addestramento è stato di tipo non supervisionato. Per valutare la bontà degli embedding generati dai modelli sono state seguite le seguenti fasi:

1. Sono state selezionate 30 parole chiave ritenute rappresentative del campione
2. Per ciascuna di essa è stato generato l'embedding dai vari modelli
3. Per ogni coppia di parole presenti è stata calcolata la cosine similarity
4. È stato chiesto al professore Pelagatti di valutare le relazioni tra le coppie di parole e creare così il gold benchmark
5. È stato calcolata la correlazione tra le valutazioni del gold benchmark e la similarità ottenuta dai modelli
6. Per ciascuna tipologia di architettura è stato selezionato il modello con la maggior correlazione

Avendo a questo punto selezionato i modelli migliori è stato possibile calcolare la similarità tra gli autori mediante due metriche differenti:

1. la cosine similarity tra gli embedding dei titoli degli autori a confronto
2. l'indice di similarità di Jaccard semplice e pesato tra le parole chiave scelte degli autori

Nella fase finale del progetto di tesi è stata eleborata la visualizzazione in due dimensioni

degli embedding dei titoli e delle parole chiavi, previa riduzione della dimensionalità mediante l'algoritmo TSNE. Queste visualizzazioni, congiuntamente alle visualizzazioni quantitative delle misure di similarità ottenuto alla fase precedente, sono state poste sotto la valutazione del professore Pelagatti, il quale ha stabilito se l'apprendimento non supervisionato ha portato a risultati significativi.

Infine, sulla base della valutazioni ricevute, sono stati elaborati i risultati della tesi, che incorporano in sé tutte le conoscenze ottenute durante il lavoro di tesi, con una rielaborazione personale sull'efficacia dei modelli dopo averli empiricamente utilizzati ed infine vengono proposti possibili sviluppi ulteriori.

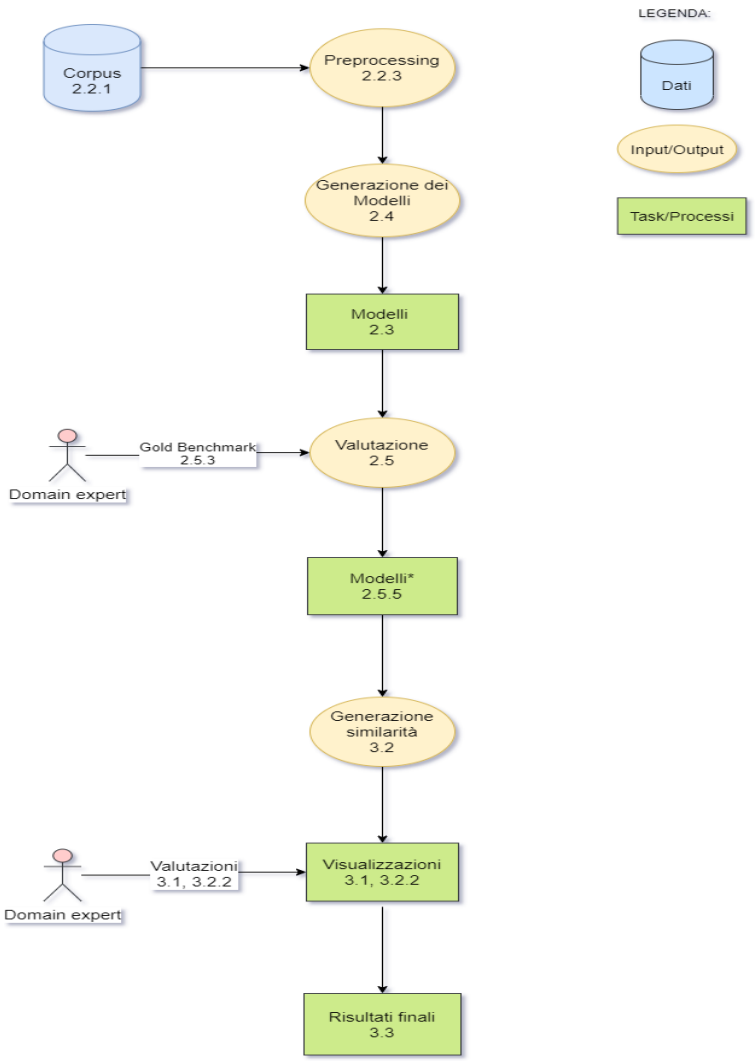

Figura 2.1 Workflow di tesi

2.1.2 Implementazione

In un primo momento l'analisi dei dati e lo sviluppo dei modelli sono stati svolti in R. Questo però sin da subito ha iniziato a mostrare le sue criticità, a causa delle limitazioni dei pacchetti disponibile e per il fatto che la maggior parte della letteratura riguardante il natural language processing fa riferimento al linguaggio di programmazione Python.

Per i motivi sopra elencati l'implementazione del lavoro di tesi è stata prevalentemente eseguita in Python. Python è particolarmente adatto a questo tipo di lavoro non soltanto perché in esso sono presenti svariati pacchetti per il natural language processing e pacchetti dei modelli utilizzati, ma anche per la sua nativa implementazione delle espressioni regolari, espressioni che consentono di lavorare in maniera efficace con le stringhe, cosa assai utile in presenza di dati testuali.

Per quanto riguarda il software utilizzato inizialmente si è lavorato in locale mediante l'utilizzo di Jupyter Notebook. Purtroppo il calcolatore sfruttato in prima battuta non risultava essere abbastanza performante, infatti i modelli utilizzati risultano essere particolarmente onerosi dal punto di vista computazionale. Per questo motivo si è passati all'utilizzo della piattaforma di Google denominata Google Colaboratory.

Google Colaboratory permette di eseguire un Jupyter notebook direttamente dal browser, con il grande vantaggio di avere a disposizione calcolatori performanti e non solo, infatti viene anche offerta la possibilità di utilizzare potenti unità di elaborazione grafica (GPU) e unità di elaborazione tensorica (TPU), risorse particolarmente utili nel momento in cui bisogna sviluppare dei modelli. Motivo ulteriore per l'utilizzo di Google Colaboratory è la sua integrazione diretta con Google Drive, che permette istantaneamente di salvare file su Drive, utile per salvare rapidamente le varie tipologie di modelli creati. Inoltre, da esso è possibile eseguire comandi direttamente dalla command line, cosa che è stata necessaria per l'utilizzo di sent2vec.

Per tenere conto degli sviluppi successivi del codice è stato utilizzato GitHub, di modo che in caso di errore fosse sempre possibile tornare ad una versione precedente del codice. Come ultimo strumento è stato usato Tableau, software per la visualizzazione interattiva dei dati, che è reso disponibile in maniera gratuita per gli studenti dall'Università degli Studi di

Milano-Bicocca. Tramite Tableau sono state create le visualizzazioni presenti nella tesi.

2.2 Dati

2.2.1 Corpus

In questo lavoro di tesi si è lavorato sul dataset degli articoli pubblicati del Dipartimento di Economia, Metodi Quantitativi e Strategie di impresa (DEMS) dell'Università degli studi di Milano-Bicocca. Il dataset è un sottoinsieme del database Scopus, il più grande database per abstract e citazioni.

Il dataset è costruito in modo tale che ad ogni riga corrisponde una pubblicazione. Esso comprende 2669 righe e 250 colonne. Le variabili presenti nel dataset sono:

- 166 variabili qualitative
- 67 variabili quantitative continue
- 12 variabili quantitative discrete
- 5 variabili che si riferiscono ad una data

Le pubblicazioni riportate in questo dataset sono comprese tra l'anno 1987, prima della fondazione della Bicocca (1998), e l'anno 2019. Gli autori diversi presenti nel dataset sono 127 ed il numero di pubblicazioni per autori varia tra un massimo di 180 pubblicazioni ad un minimo di una pubblicazione.

2.2.2 Analisi Esplorative

L'analisi del dataset si è concentrata esclusivamente sulle variabili che riportavano il titolo, gli abstracts e le parole chiave fornite personalmente dal professore (sono presenti anche le parole chiave formate in automatico da scopus). Questo poiché, essendo l'obiettivo precipuo di questo studio rappresentare in maniera significativa un articolo tramite tecniche di word embedding, i dati utilizzati sono dati unicamente testuali, contenenti informazioni specifiche della pubblicazione stessa.

Il primo problema riscontrato è risieduto nel fatto che sia i titoli, che gli abstract, che le parole chiave erano stati inseriti nelle medesime variabili in lingua differente, quali:

- Inglese
- Italiano

- Tedesco
- Francese
- Spagnolo

Dal momento che la maggior parte dei dati è in lingua inglese e che i modelli pre-istruiti che verranno successivamente utilizzati richiedono un input in inglese, la prima soluzione pensata fu quella di tradurre in automatico in lingua inglese tutti i dati di lingua differente. Questa soluzione venne però scartata, poiché avrebbe introdotto nel dataset una componente di distorsione dovuta alla traduzione algoritmica, che si sarebbe propagata con un effetto a cascata anche sui modelli, venendo a creare modelli distorti.

Accantonata l'idea della traduzione si è passati alla fase di selezione dei testi in lingua inglese. Questa è avvenuta tramite l'utilizzo di un classificatore linguistico, in particolare è stato usato l'identificatore per i linguaggi Langid in Python, il quale dato un input testuale è in grado di stabilirne la lingua di appartenenza. In ultima battuta sono stati eliminati tutti i duplicati presenti nei titoli e negli abstract.

In Tabella 2.1 è presente un confronto tra le parole utilizzate per ciascun titolo e quelle utilizzate per ciascun abstract. Come si può notare gli abstract in lingua inglese sono in numero inferiore rispetto ai titoli (916 contro 1542). Invece, ragionevolmente, la lunghezza dei titoli è di gran lunga inferiore a quella degli abstract, con rispettive lunghezze medie di 10 parole e 148 parole.

	count	mean	std	min	median	max
Titoli	1548.0	10.686047	4.099397	1.0	10.0	28.0
Abstract	916.0	147.945415	75.596754	26.0	132.0	681.0

Tabella 2.1 Tabella riassuntiva sul numero di parole presenti per Titoli e Abstract

La tabella 2.2 evidenzia come la maggiore lunghezza degli abstract si traduca giocoforza in una maggiore quantità di parole totali utilizzate, 16542 per i titoli, 135518 per gli abstract, tendenza che si riflette anche sulla grandezza del vocabolario, definito come numero di parole uniche utilizzate, dove il vocabolario degli abstract risulta essere all'incirca tre volte più grande di quello dei titoli, 3247 rispetto a 9758.

	N. parole tot.	Vocabolario
Titoli	16542	3247
Abstract	135518	9758

Tabella 2.2 Confronto grandezze tra Titoli ed Abstract

Per analizzare nel dettaglio quali parole siano stati usate e quante volte esse vengano ripetute all'interno degli abstract possiamo fare riferimento alla Figura 2.2. Come si può notare quasi il 40% delle parole che costituiscono il vocabolario sono state utilizzate una sola volta all'interno dell'intero corpus. Inoltre, soltanto meno del 30% delle parole compaiono più di 5 volte. Questo può essere dovuto alla specificità insita in ciascuna pubblicazione, che porta l'autore ad usare un lessico altamente specializzato nel proprio ambito di ricerca, che difficilmente si ripropone in studi differenti.

Per quanto riguarda l'utilizzo delle parole nella variabile titoli l'andamento di fondo è simile a quello negli abstract, ma con un'intensità più accentuata. Come mostrato in Figura 2.3 quasi il 50% delle parole nel vocabolario compaiono una sola volta all'interno del corpus. Di contro, solo il 15% delle parole del vocabolario compaiono nel corpus più di 5 volte. Il motivo di questo comportamento è lo stesso riportato per il caso degli abstract, con l'unica differenza che per quanto riguarda i titoli, essendo essi di lunghezza inferiori, portano ad una maggiore sparsità del lessico utilizzato.

Dopo aver attentamente valutato le differenze nelle variabili, si è deciso di costruire i modelli e gli embedding sulla base dei titoli e non degli abstract. Questa scelta è stata presa in base a molteplici motivi.

In primo luogo, la numerosità campionaria dei titoli risulta essere significativamente maggiore di quella degli abstract, valutata come troppo esigue per soddisfare le esigenze dei modelli utilizzati.

In secondo luogo, sebbene nei titoli molte parole compaiono unicamente, rendendo pertanto difficoltoso l'obiettivo di un embedding significativo, va sottolineato altresì che le parole utilizzate nei titoli risultano essere estremamente specifiche dell'argomento trattato, pertanto un embedding dei titoli in linea di principio dovrebbe essere maggiormente caratterizzante rispetto ad un embedding degli abstract.

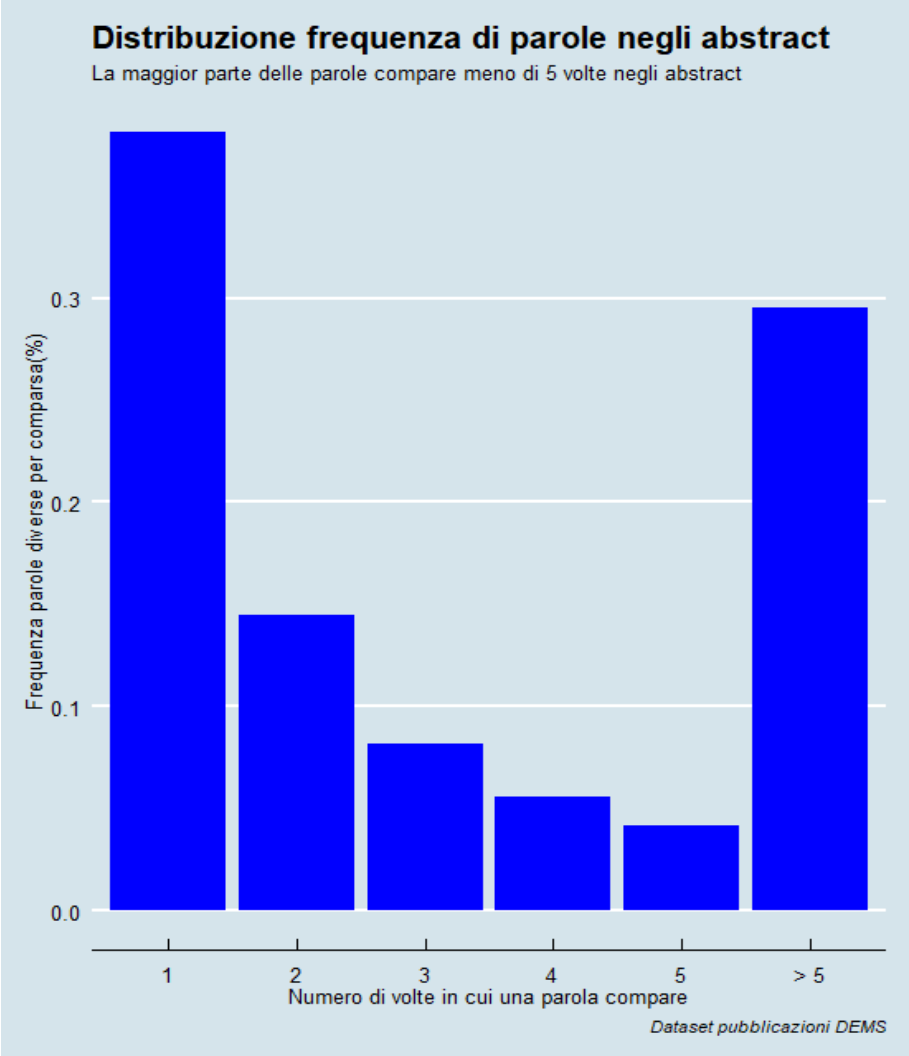

Figura 2.2 Istogramma rappresentante la distribuzione della frequenza di parole negli abstract

Infine, è stata presa in considerazione la diversa lunghezza tra i titoli e gli asbtract, infatti gli abstract risultano, oltre ad essere più prolissi, meno omogenei tra se stessi, si va da abstract lunghi una ventina di parole ad asbtract lunghi una centinaia. A questa motivazione si aggiunge il fatto che abstract particolarmente lunghi, creano un embedding meno denso, avendo essi la possibilità di svariare di più. Invece i titoli, essendo molto più omogenei, creano embedding della stessa densità tra loro, pertanto embedding più significativi e su cui è possibile sviluppare un confronto.

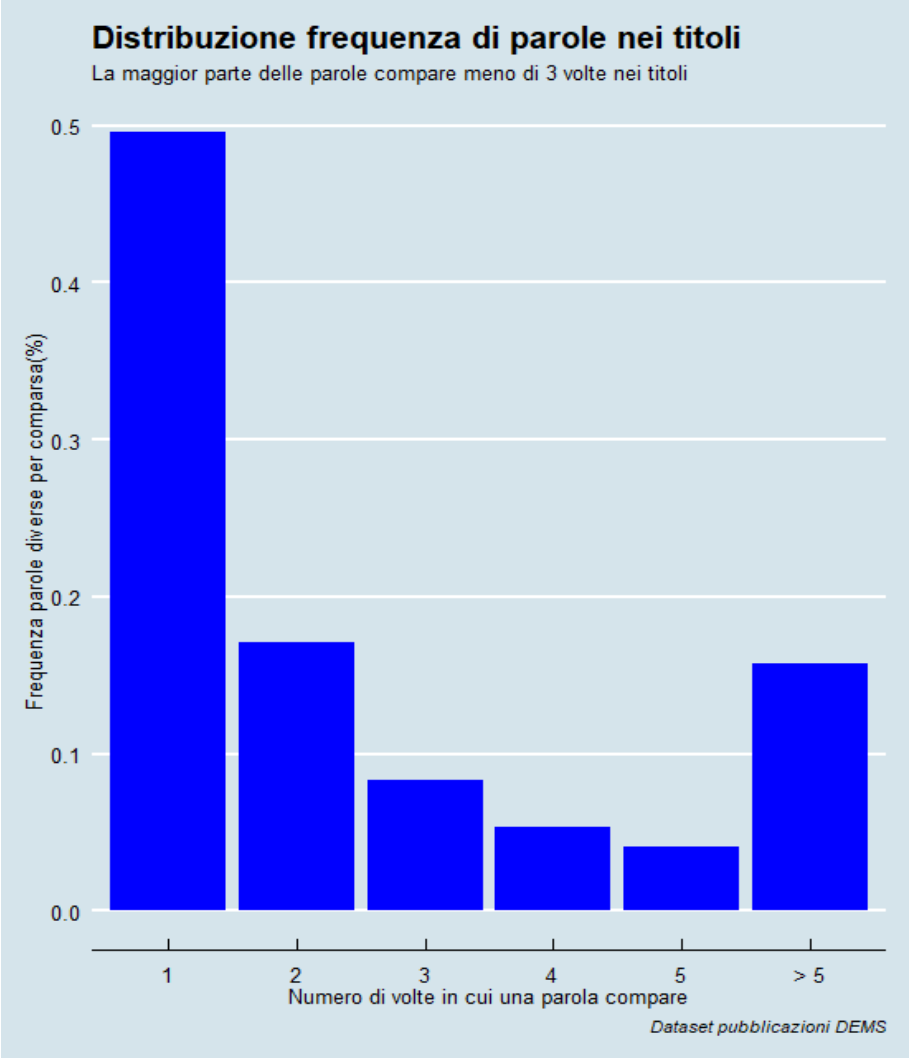

Figura 2.3 Istogramma rappresentante la distribuzione della frequenza di parole nei titoli

2.2.3 Preprocessing

Come affermato nel capitolo precedente si è deciso di usare i titoli come dato testuale da dare in pasto ai vari modelli. Le prime fasi del preprocessing erano già state implementate nelle analisi esplorative, infatti dei vari titoli sono stati selezionati solamente quelli in lingua inglese ed unici.

Nella fase seguente di pulizia dei dati si è fatto utilizzo delle espressioni regolari per eliminare dai titoli la punteggiatura, non portando essa particolare informazione. Si è poi proceduto alla riduzione del testo in minuscolo e ci si è assicurato che non ci fossero incongruenze, quali ad esempio spazi vuoti, all'inizio ed alla fine dei paragrafi.

Una fase particolarmente comune nel naturale language preprocessing è quella della rimozione delle stop words e della lemmatizzazione. La prima consiste nel rimuovere le parole più comuni, dette stop words, che formano la struttura di qualsiasi frase di senso compiuto. Esse sono ad esempio le preposizioni e gli articoli, elementi della sintassi particolarmente presenti ma che non portano informazioni di natura specifca, per cui è pratica comune eliminarle. La lemmatizzazione consiste invece nel sostituire una parola con il suo lemma, cioè sostituire tutte le parole in un corpus con la loro forma di citazione in un dizionario. Si è deciso di non usufruire di queste due tecniche poiché esse vanno esplicitamente ad alterare l'informazione portata da una frase, andando inoltre ad eliminare vocaboli che sono parte integrante della stessa, portando pertanto ad una perdita di informazione. Infine, i modelli utilizzati di word embedding, basati su reti neurali, richiedono esplicitamente il dato testuale nel suo intero, in quanto necessario per prevedere accuratamente le parole vicine tra loro.

Nella fase finale del preprocessing sono stati creati i bigrammi e i trigrammi dei titoli. Un n-gramma, in questo ambito, è una sottosequenza di n parole in una data frase. Costruire gli n-grammi ed aggiungerli all'input iniziale per darli congiuntamente in pasto ad un modello è una pratica vastamente utilizzata, poiché permette di considerare in maniera unica quei termini che non riescono ad essere definiti da una singola parola, per cui raggiungono un senso compiuto specifico grazie alla contemporanea presenzadi più parole. Un termine di questo tipo è, ad esempio, machine learning. Una volta creati i bigrammi e i trigrammi di ogni titolo sono stati costruiti tre diversi input testuali:

1. L'input testuale dei titoli originale
2. Un input costituito dalla concatenazione tra gli unigrammi ed i bigrammi per ogni titolo
3. Un input costituito dalla concatenazione di unigrammi, bigrammi e trigrammi per ogni titolo

Questi diversi input sono stati utilizzati come parametri per generare modelli diversi.

2.3 Vector Space Models

2.3.1 Introduzione

I vector space model sono modelli che permettono di rappresentare dati testuali in uno spazio vettoriale. Questa loro capacità è di grandissimo interesse sia in ambito pratico che in ambito teorico, infatti riuscire a rappresentare qualsiasi dato testuale in vettori numerici rende possibile l'utilizzo di tecniche di machine learning su qualsiasi dato testuale.

Lo sviluppo di questi modelli è più che mai attuale, infatti allo sviluppo di reti neurali artificiali ne è seguito un loro utilizzo per la creazione di sempre nuovi e più performanti vector space model. Dato questo continuo e contemporaneo sviluppo non è ancora possibile definire il modello uniformemente migliore, motivo per cui è necessario testare diversi modelli sul proprio caso empirico poiché da caso a caso il modello migliore potrebbe variare.

Nell'ambito dei vector space model bisogna innanzitutto fare una divisione tra le tecniche di word embedding, che permettono la rappresentazione di singole parole in vettori numerici e le tecniche di document o sentence embedding che si basano sulle tecniche precedenti per creare una rappresentazione numerica non di singole parole, ma bensì di qualsiasi sequenza delle stesse, per cui sia di intere frasi che, volendo, di interi libri o documenti. In questo lavoro di tesi verranno presentate le seguenti tecniche di word embedding:

- Cbow
- Skip-gram
- BERT

Sulle quali si basano le seguenti tecniche di sentence embedding:

- Doc2vec
- Sent2vec
- Universal Sentence Encoder
- SBERT

Dei vari modelli esistenti in letteratura sono stati scelti questi specifici modelli di sentence embedding poiché si basano su approcci diversi.

Doc2vec è un modello sviluppato dai ricercatori Google che sfrutta la relativamente

semplice architettura CBOW e Skip-Gram per costruire un modello di document embedding. Il modello risultante è un modello che si adatta ai dati fornitigli.

Sent2vec è un modello sviluppato sull'implementazione fastText, sviluppata da Facebook, dell'architettura CBOW. Questa implementazione basata su un nucelo in C++ permette a Sent2vec di essere particolarmente efficiente.

L'universal Sentence Encoder è sempre un algoritmo sviluppato da Google, però a differenza di Doc2vec esso si basa su un'architettura simile a CBOW ma più profonda. Inoltre, questo modello è un modello preistruito, questo vuol dire che non è necessario il suo addestramento sullo specifico dataset, poiché è già stato addestrato su milioni di dati provenienti da più fonti. Questo pre-addestramento può essere un'arma a doppio taglio: se è vero che il modello ha potuto accedere ad una quantità di informazioni e dati imponente, non è detto che questa conoscenza generale riesca a tradursi in una maggior sensibilità in un ambito così specifico come quello in studio in questa tesi.

SBERT è un modello di rete neurale siamese che si basa su un'architettura particolarmente complessa, quella di BERT, che nel 2018 ha raggiunto risultati che hanno rappresentato lo stato dell'arte in ambiti svariati che vanno dalla sentiment analysis alla classificazione del testo. Il modello di per sé è pre-istruito, però offre anche la possibilità di eseguire un fine-tuning su nuovi dati, cioè il modello, inizializzato con i pesi del precedentemente addestramento, permette di ricalibrare gli stessi per adattarsi al nuovo caso specifico, lasciando invariata la struttura di base e specializzando gli ultimi layer per adattarsi al nuovo problema.

2.3.2 Continuos Bag of Words (CBOW)

In CBOW (Mikolov et al. [5]) le parole in una data finestra di ampiezza predefinita vengono utilizzate come input per prevedere la parola centrale, per cui vengono utilizzate sia le parole precedenti che successive alla parola da prevedere. Mediante la media dei vettori di input viene ricavato il vettore di proiezione. Questo metodo viene per l'appunto chiamato Bag of words in quanto l'ordine delle parole non ha alcuna influenza ed è rappresentato in Figura 2.4.

Per prevedere la parola d'interesse viene utilizzato un classificatore log-lineare, per ciò la complessità del modello è proporzionale a:

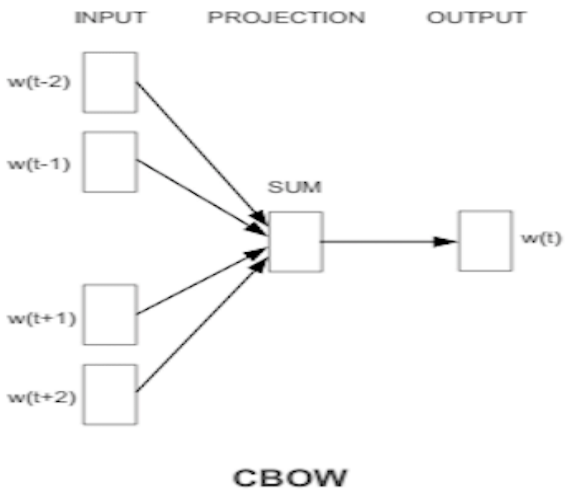

Figura 2.4 L'architettura CBOW predice la parola corrente basata sul contesto

$$Q = N \times D + D \times \log_2 V, \tag{2.1}$$

2.1 Grado di complessità CBOW

dove N è l'ampiezza della finestra, D è la dimensione dell'embedding e V è la grandezza del vocabolario.

L'embedding finale delle parole è dato dal vettore dei pesi del vettore di proiezione ottimizzati in fase di previsione.

2.3.3 Continuos Skip-gram

L'architettura della rete del continuos skip-gram (Mikolov et al. [5]), riportata in Figura 2.5, può essere vista come l'inverso di quella del CBOW. In questo caso l'input è la singola parola e si cerca di massimizzare la classificazione di parole che la precedono e che la seguono. Più precisamente viene usata la parola corrente come input di un classificatore log-lineare con layer continui di proiezione e si predicono le parole in un certo raggio prima e dopo della parola corrente.

All'aumento dell'ampiezza corrisponde un aumento nella qualità dei vettori ottenuti, al costo però di un aumento della complessità computazionale. Nel modello viene dato meno

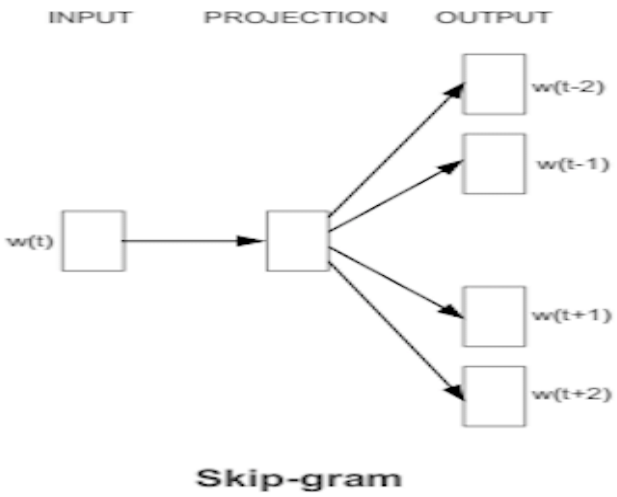

Figura 2.5 L'architettura Skipgram predice le parole circostanti data la parola corrente

peso alle parole più lontane dalla parola corrente. La complessità di addestramento di questa architettura è proporzionale a:

$$Q = C \times (D + D \times \log_2 V), \tag{2.2}$$

2.2 Grado di complessità Skip-gram

dove C è la massima distanza delle parole.

L'embedding di una data parola, similarmente a quanto accade con CBOW, viene ricavato dai pesi ottenuti dalla classificazione.

2.3.4 Doc2vec

Introduzione

I modelli di paragraph vectors (Le and Mikolov [4]) si basano su una struttura fissa per cui ciascun paragrafo è mappato da un unico vettore, rappresentato da una colonna nella matrice D e ciascuna parola è mappata da un vettore unico, rappresentata da una colonna nella matrice W. Su queste basi sono stati costruiti due differenti tipi di modelli: paragraph vectors distributed memory (PV DM) e i paragraph vectors distributed bag of words (PV DBOW).

PV DM

Nei PV DM l'ampiezza degli insiemi di parole selezionate è costante e gli insiemi sono estratti da una finestra che si muove all'interno del paragrafo. Il vettore dei paragrafi è condiviso tra tutti gli insiemi generati dallo stesso paragrafo ma non fra i paragrafi. La matrice delle parole W, invece, è condivisa tra tutti i paragrafi.

La struttura della rete è rappresentata dalla Fig. 2.6, come si nota essa è una derivazione della struttura di CBOW, con l'aggiunta della matrice D dei paragrafi.

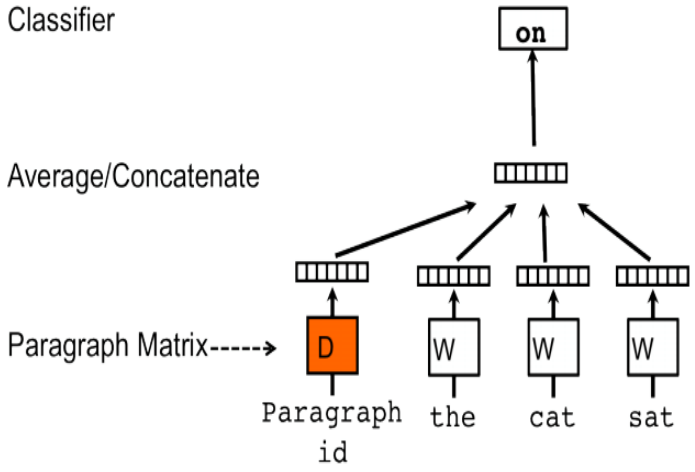

Figura 2.6 L'architettura del PV DM è estremamente simile a quella presentata in Fig. 2.4 con l'unica aggiunta del token del paragrafo che è mappato da un vettore della matrice D.

I vettori dei paragrafi ed i vettori delle parole sono istruiti tramite discesa del gradiente stocastica ed il gradiente è ottenuto tramite backpropagation. A ciascuno step della discesa del gradiente stocastica viene estratto un insieme di parole di lunghezza prefissata da un paragrafo casuale, viene calcolato il gradiente dell'errore della rete in Figura 2.6 e viene usato quest'ultimo al fine di aggiornare i parametri del modello.

In fase di previsione il vettore di paragrafi, essendo esso mancante, viene calcolato tramite uno step di inferenza. Anche questo passaggio viene ottenuto tramite discesa del gradiente stocastica. In questo step, i parametri del resto del modello, cioè il vettore delle parole W ed i pesi softmax, sono fissi.

Dati N paragrafi, M parole nel vocabolario, si vuole apprendere il vettore di paragrafi tale per cui ciascun paragrafo è rappresentato da p dimensioni e ciascuna parola è rappresentata de q dimensioni, allora il modello ha un totale di $N \times P + M \times Q$ parametri (escludendo i parametri di softmax). Gli aggiornamenti sono sparsi e per tanto efficienti.

Brevemente l'algoritmo in sé ha due fasi principali:

1. L'addestramento per ottenere il vettore delle parole W, i pesi softmax U, b ed i vettori dei paragrafi D sui paragrafi già noti
2. La fase di inferenza per ottenere i vettori dei paragrafi D per nuovi paragrafi, aggiungendo più colonne in D ed usando la discesa del gradiente su D mantenendo fissi W, U, b

PV DBOW

Il PV DBOW può essere considerato come una variazione del modello Skip-Gram, come si vede in Figura 2.7, in cui l'input della rete è ora non più una parola, bensì la matrice dei paragrafi.

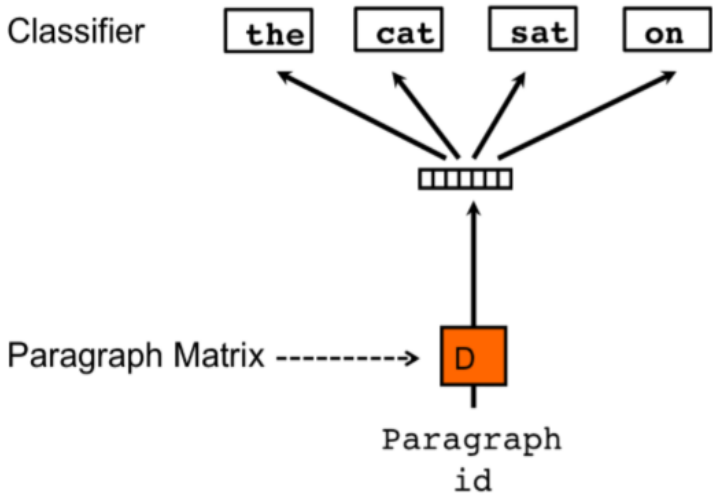

Figura 2.7 In questa versione il vettore dei paragrafi è addestrato per prevedere le parole in una piccola finestra

In particolare, a ciascuna iterazione della discesa stocastica del gradiente, viene estratta una finestra di testo e da essa viene estratta in maniera casuale una parola, che diventa l'obbiettivo di classificazione a partire dal vettore del paragrafo.

Confronto PV DM e PV DBOW

Le differenze tra i due modelli di paragraph vectors sono le seguenti:

1. Il modello PV DM richiede l'utilizzo e l'immagazzinamento di più parametri, cioè i pesi softmax ed il vettore delle parole W, mentre il modello PV-DBOW richiede solo i pesi softmax
2. Il modello PV DBOW non considera l'ordine delle parole, a differenza del modello PV DM che tiene in considerazione l'ordine delle parole almeno per quanto riguarda il contesto
3. Per i motivi elencati di cui sopra il modello PV DM risulta essere maggiormente oneroso dal punto di vista computazionale rispetto il modello PV DBOW

2.3.5 Sent2vec

Il modello Sent2vec (Pagliardini et al. [6]) può essere visto come la naturale estensione dal contesto per parola del modello CBOW ad un contesto globale per ogni frase, essendo le parole scelte dal modello specificamente ottimizzate attraverso combinazioni additive nella frase, per mezzo di una funzione obbiettivo non supervisionata.

Formalmente si apprendono gli embedding della fonte v_w e del target u_w, per ciascuna parola w nel vocabolario V, con dimensione dell'embedding rispettivamente h e $k = |V|$. L'embedding della frase è calcolato come la media degli embedding delle parole fonte costituenti la frase. Questo modello è aumentato non solo considerando gli unigrammi, ma per ogni frase vengono considerati gli n-grammi presenti e gli embedding di quest'ultimi vengono utilizzati insieme a quelli delle parole per calcolare il sentence embedding. Il sentence embedding vs per la frase S, pertanto, è definito dalla seguente formula:

$$v_S := \frac{1}{|R(S)|} V \iota_{R(S)} = \frac{1}{|R(S)|} \sum_{w \in R(S)} v_w \tag{2.3}$$

2.3 Sentence embedding

dove $R(S)$ è la lista degli n-grammi (compresi gli unigrammi) presenti nella frase S ed il vettore indicatore $\iota_S \in \{0,1\}^{|V|}$ è un vettore binario che codifica S.

Al fine di predirre la parola mancante dal contesto è possibile utilizzare tre diverse funzioni di attivazione:

1. Negative sampling
2. Softmax
3. Hierarchical softmax

Per selezionare i possibili unigrammi target viene utilizzato il sottocampionamento, dove ogni parola w viene scartata con probabilità 1–$q_p(w)$ dove:

$$q_p(w) := \min\{1, \sqrt{\frac{t}{f_w}} + \frac{t}{f_w}\} \tag{2.4}$$

2.4 Probabilità di selezione

dove f_w è la frequenza normalizzata di w nel corpus e t è l'iperparametro di campionamento. Il sottocampionamento previene che parole troppo frequenti abbiano troppa influenza nell'apprendimento.

Il modello Sent2vec risulta essere particolarmente efficiente dal punto di vista computazionale. Infatti, data una frase ed un modello istruito, calcolare la rappresentazione della frase v_s richiede solamente $R(S|H|)$ operazioni a virgola mobile.

2.3.6 Universal Sentence Encoder

L'unversal sentence encoder (Cer et al. [1]) è un encoder di sentence embedding che si basa sull'architettura DAN (Deep avarage network) (Iyyer et al. [3]). La struttura del modello è una struttura atta alla classificazione del testo.

Come mostrato dalla Figura 2.8 data una frase singola, cioè una sequenza di tokens X, l'obbiettivo della rete è quello di ottenere come output una delle k possibili classificazioni della frase.

In primo luogo viene applicata una funzione di composizione g alla sequenza degli embedding delle parole v_w per $w \in X$ e dei bigrammi. Questa operazione è data dalla seguente formula:

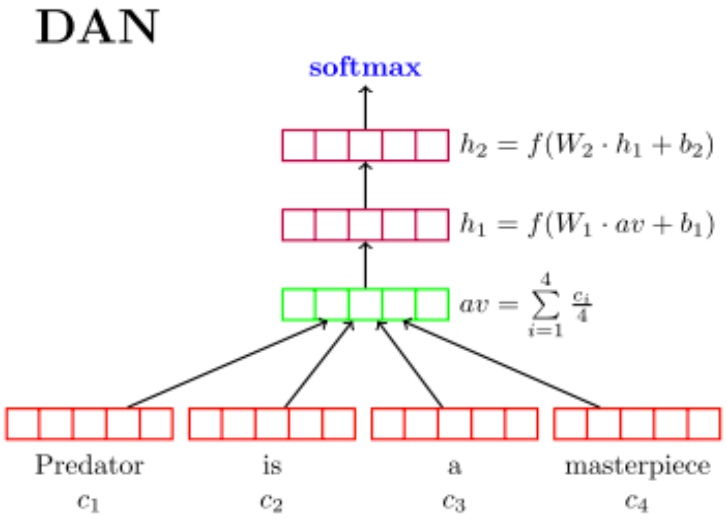

Figura 2.8 Raffigurazione dell'architettura DAN

$$z = g(w \in X) = \frac{1}{|X|} \sum_{w \in X} v_w \tag{2.5}$$

2.5 Operazione di composizione

Successivamente al posto di passare questa rappresentazione direttamente ad un output layer, viene ulteriormente trasformata z aggiungendo diversi layer prima di applicare la funzione softmax. Supponendo che abbiamo n layer, per ciascun layer viene calcolato:

$$z_i = g(z_{i-1}) = f(W_i \cdot z_{i-1} + b_i) \tag{2.6}$$

2.6 Operazione generale di composizione

dove W_i è una matrice $k \times d$ e b_i è un termine di bias.

Infine l'ultima rappresentazione dell'n-esimo layer z_n viene usata come input del layer softmax di previsione, dato da:

$$\hat{y} = softmax(W_n \cdot z_n + b_n) \tag{2.7}$$

2.7 Previsione utlimo layer

Questo modello, utilizzando a valle una funzione di composizione g, risulta non tenere conto dell'ordine delle parole. La profondità di questo modello permette ugualmente di catturare lievi variazioni negli input in maniera migliore di un classico modello CBOW.

Per quanto riguarda la complessità calcolare ogni layer richiede solo una moltiplicazione matriciale, pertanto la complessità aumenta in maniera proporzionale all'aumentare del numero di layer utilizzati.

Il modello è un modello preistruito ed è stato addestrato su problemi di classificazione diversi, pertanto un singolo DAN è usato per ottenere sentence embedding per differenti compiti. In particolare l'Universal Sentence Embedding è stato addestrato sia tramite dati per l'addestramento non supervisionato estratti da internet da fonti come Wikipedia, web news, pagine web di domande e risposte e forum, sia da dataset per l'addestramento supervisionato quali:

1. Stanford Natural language inference(SNLI): una collezione di 570000 coppie di frasi annotate con le classi contradiction, entailment e neutral
2. SST: dataset che contiene 215154 frasi etichettate per la sentiment analysis
3. STS benchmark comprende 8628 coppie di frasi etichettate

Il modello preistruito prende come input stringhe in minuscolo tokenizzate con PTB e restituisce come output un sentence embedding di 512 dimensioni.

2.3.7 BERT

BERT (Devlin et al. [2]), che sta per Bidirectional Encoder Rappresentation for Transformer, è un modello di rappresentazione del linguaggio il cui scopo è quello di preistruire profonde rappresentazioni bidirezionali da testi non etichettati, condizionando congiuntamente il contesto sia da destra che da sinistra in tutti i layer.

BERT si sviluppa in una struttura articolata in due fasi:

1. La fase di pre-training: il modello è istruito su dati non etichettati per assolvere diversi compiti
2. La fase di fine-tuning: il modello è inizializzato con i parametri ottenuti dal pre-training, successivamente tutti i parametri sono ricalibrati per differenti scopi usando

dati etichettati. A scopi diversi corrispondo diversi modelli ricalibrati, sebbene i parametri inizializzati siano gli stessi.

Architettura del modello

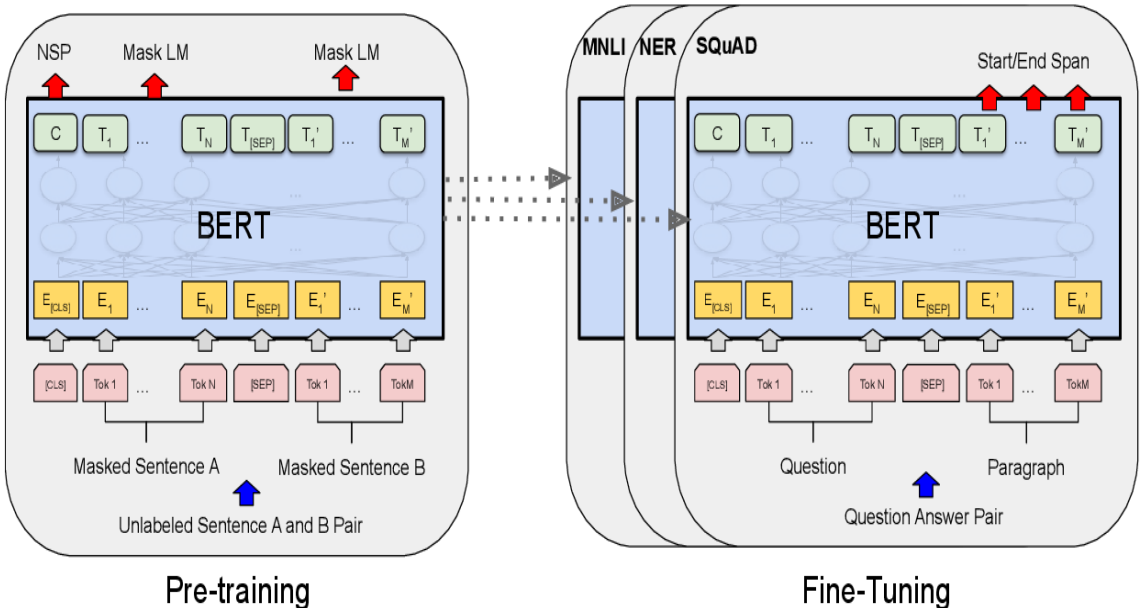

Figura 2.9 Struttura complessiva delle procedure di pre-training e fine tuning per BERT. Eccezion fatta per gli output layer, la stessa architettura è usata sia per il pre-training che per il fine-tuning. Gli stessi pre-addestrasti parametri del modello sono utilizzati per inizializzare modelli con scopi diversi. Durante il fine tuning tutti i parametri vengono ricalibrati. [CLS] è un simbolo speciale aggiunto davanti ad ogni input, mentre [SEP] è un token speciale di separazione

L'architettura del modello è rappresentata in Figura 2.9 e si tratta, come dice il nome stesso, di un struttura multi layer bidirezionale di tipo transformer encoder. In particolare il modello di base ha 12 layer, 768 è la grandezza dei pesi nascosti e 12 è il numero delle teste di auto-attenzione.

Come si può notare la struttura rimane pressoché invariata per le differenti procedure di pre-training e fine-tuning, questo perché la differenza risulta essere insita non nella tipologia di architettura, bensì nella diversa tipologia di dati in input (non etichettati vs etichettati) e il diverso scopo che essa deve raggiungere.

Gestione degli input

Per rendere BERT il più generale possibile, la rappresentazione dell'input testuale è in grado di rappresentare in maniera non ambigua sia una singola frase che una coppia di frasi in una singola sequenza di token. Questo risultato è stato raggiunto mediante una particolare rappresentazione dell'input che viene esemplificata in Figura 2.10.

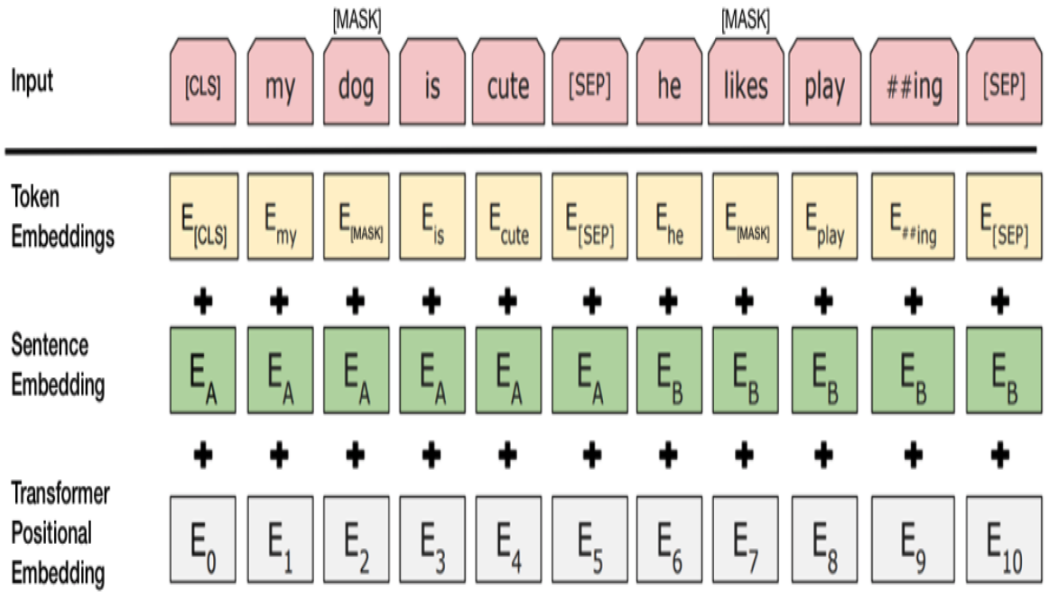

Figura 2.10 Rappresentazione degli input di BERT. Gli input embeddings sono la somma dei token embeddings, degli embedding di segmentazione e degli embeddings di posizione

In primo luogo viene applicata all'input testuale la tokenizzazione WordPiece con un vocabolario di 30000 token. Questo tipo di tokenizzazione divide in automatico le parole in sotto-parole al fine di mantenere fisso il numero di vocaboli nel vocabolario. Questo metodo permette di evitare di incorrere nel problema di aver parole al di fuori del vocabolario, quindi parole non gestibili da un semplice modello, infatti quest'ultime possono essere rappresentate come una successione di sotto-parole presenti tra i 30000 token scelti.

Ai token divisi in questo modo viene aggiunto in prima posizione un token speciale di classificazione ([CLS]). Le coppie di frasi sono raggruppate insieme in una singola sequenza. Esse vengono differenziate in due modi:

1. Vengono separate da un token speciale([SEP])
2. Viene aggiunto un token di segmentazione per stabilire se i token appartengono alla frase A o alla frase B

Infine viene aggiunto un token di posizionamento che porta informazione riguardante alla posizione del token nella sequenza completa di token.

Pertanto dato un input testuale, la sua rappresentazione per il modello è definita dalla somma dell'embedding del corrispondente token, dell'embedding di segmentazione e dell'embedding di posizione, come mostrato in Figura 2.10.

Pre-training BERT

Durante la fase di pre-training i parametri del modello vengono aggiornati per rispondere congiuntamente a due specifici scopi:

1. Previsione dei token mascherati: per addestrare il modello bidirezionale vengono mascherati il 15% di tutti i WordPiece token, estratti casualmente, essi vengono sostituiti con un token speciale [MASK]. L'obbiettivo della rete è quello di predire accuratamente questi token
2. Next sentence prediction(NSP): dato un corpus monolingua al modello vengono offerti come input una coppia di frasi, A e B, dove nel 50% dei casi la frase B segue la frase A, nel rimanente 50% dei casi la frase B è una frase presa casualmente dal corpus. Lo scopo del modello è quello di prevedere accuratamente se le due frasi sono realmente consecutive o se, invece, esse non sono successive.

Questa fase di pre-training è stata svolta su due immensi corpus:

1. BooksCorpus: corpus da 800M di parole in cui sono presenti diverse varietà di libri in lingua inglese
2. Wikipedia inglese: corpus da 2500M di parole estratte da Wikipedia ignorando le liste, le tabelle e le intestazioni.

Fine-tuning

Nella fase di fine-tuning vengono semplicemente inseriti nel modello valori di input ed output specifici per l'obbiettivo di interesse e vengono aggiornati i parametri. In questa fase il modello viene addestrato con dati etichettati per scopi che vanno dalla classificazione per sentiment analysis, al riconoscimento di domande e risposte, al riconoscimento delle relazioni di causa e di effetto.

2.3.8 SBERT

Sentence BERT (SBERT) (Reimers and Gurevych [7]) è una modificazione della rete BERT usando una rete siamese, cioè una rete neurale che utilizza gli stessi pesi mentre lavora contemporaneamente su due vettori in input per calcolare come output dei vettori che siano comparabili. Questo tipo di struttura permette a SBERT di derivare un sentence embedding di ampiezza fissa semanticamente significativo.

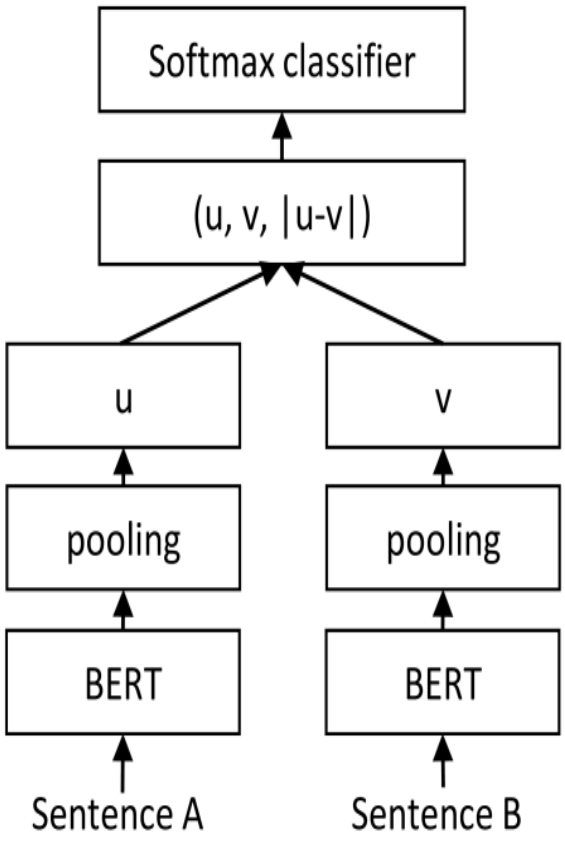

Figura 2.11 Architettura di SBERT con un funzione di classificazione come funzione obbiettivo. Le due rete BERT hanno i pesi legati (struttura di una rete neurale siamese)

La struttura del modello è presentata nella Figura 2.11. Dato un output del modello BERT pre-istruito SBERT esegue un'operazione di aggregazione dello stesso per derivarne un vettore di dimensioni fisse. La funzione di aggregazione utilizzata è il calcolo della media tra tutti i vettori di output di BERT.

Successivamente vengono concatenati i sentence embedding delle due frasi di input u e v con la loro differenza element wise $|u-v|$ ed il risultato della concatenazione viene moltiplicato con i pesi di addestramento $W_t \in \mathbb{R}^{3n \times k}$. Il risultato finale viene utilizzato per calcolare la funzione obbiettivo:

$$o = softmax(W_t(u, v, |u-v|)) \tag{2.8}$$

2.8 Funzione obbiettivo SBERT

Dove n è la dimensione dei sentence embedding e k è il numero delle classi. Viene successivamente ottimizzata la perdita cross-entropica.

Il modello SBERT è stato pre-istruito su una combinazione dei dataset SNLI e Multi Genre NLI. Il SNLI (Stanford natural language inference corpus) è una collezione di 570000 coppie di frasi annotate con le classi contradiction, entailment e neutral. Il MultiNLI contiene 430000 coppie di frasi e ricopre una vasta varietà di generi del linguaggio scritto e parlato. SBERT è stato addestrato con un softmax a tre vie per un'epoca, con batch size di 16, l'ottimizzatore Adam con un tasso di apprendimento di 2e-5 ed un warm-up period sul 10% dei dati di training.

2.3.9 Conclusione sui modelli

I modelli presentati coprono un vario spettro di architetture differenti. Essi inoltre si differenziano sia per grado di complessità sia per tipologia di addestramento (non istruiti, pre-istruiti, fine-tuned di modelli pre-istruiti). Per queste loro differenti caratteristiche sono stati scelti in questo lavoro di tesi, con la speranza che almeno uno di essi porti a risultati significativi.

2.4 Generazione Modelli

2.4.1 Introduzione

Per assicurarsi che i modelli descritti nella sezione 2.3 coprano una gamma più ampia possibile di diverse casistiche, sono stati generati molteplici modelli Doc2vec e Sent2vec, variando tra di essi i parametri del modello specifico.

Per quanto riguarda il modello Universal Sentence Encoder è stato semplicemente applicato così com'è, trattandosi esso di un modello pre-istruito.

Infine, per il modello SBERT verrà descritta la procedura attuata al fine di renderlo adatto al contesto di tesi, pertanto verrà descritto come attraverso il fine-tuning si è passato da un modello pre-istruito ad un modello specifico per i dati analizzati.

2.4.2 Doc2vec

Per il modello doc2vec sono stati mantenuti i seguenti parametri fissi:

- Numero di epoche: 100
- Tasso di apprendimento: 0.025
- Numero minimo di parole considerate: 1
- Ampiezza finestra di contesto: 3

Dati questi parametri prefissati sono stati costruiti 18 modelli in base alle combinazioni dei diversi fattori:

- Tipo di architettura: PV DM o PV DBOW
- Grandezza del vettore di embedding: 300, 400 o 500
- Dati in input: unigrammi, unigrammi e bigrammi o unigrammi, bigrammi e trigrammi

2.4.3 Sent2vec

Per il modello Sent2vec, la cui implementazione è possibile solo tramite la linea di comando del terminale, sono stati mantenuti fissi i seguenti parametri:

- Numero di epoche: 20
- Tasso di apprendimento: 0.2

- Numero minimo di parole considerate: 1

Dati questi parametri prefissati cono stati costruiti 27 modelli in base alle combinazioni dei seguenti fattori:

- Tipo di funzione di perdita: softmax, negative sampling o hierarchical softmax
- Grandezza del vettore di embedding: 300, 400 o 500
- Dati in input: unigrammi, unigrammi e bigrammi o unigrammi, bigrammi e trigrammi

2.4.4 Universal Sentence Encoder

L'Universal Sentence Encoder non necessita di alcuna generazione, in quanto è un modello pre-istruito. Per il suo utilizzo è solo necessario il suo download dall'Hub di TensorFlow di Google. Esso permette di creare un embedding di 512 dimensioni dell'input testuale fornitogli.

2.4.5 SBERT

SBERT si base su un modello pre-istruito su cui si è sviluppato un fine-tuning per fornire al modello una conoscenza di dominio sui dati trattati. Dal momento che il metodo di addestramento di SBERT è di tipo supervisionato, si è cercato di sviluppare un addestramento supervisionato che permettesse all'algoritmo di approfondire la sua conoscenza nell'ambito delle pubblicazioni scientifiche di tipo statistico-economico.

Per ottenere questo tipo di addestramento si è sfruttato il seguente espediente basato sui dati testuali degli abstract. Presi gli abstract sono stati preprocessati, togliendo la punteggiatura non necessaria ed eliminando tramite espressioni regolari il publisher spesso presente come frase finale degli abstract.

Successivamente ogni abstract è stato diviso nelle frasi che lo compongono. Ogni singola frase è poi stata accoppiata due volte: una volta con la frase seguente ed una volta con una frase random presa dal corpus. In totale sono state create 9822 coppie di frasi di cui il 50% sono state etichettate come frasi seguenti ed il restante 50% sono state etichettate come frasi indipendenti.

Infine, queste coppie di frasi sono state date come input alla rete siamese di SBERT con il compito di classificarle correttamente. Il modello risultante è stato poi utilizzato come modello di sentence embedding per SBERT.

2.5 Valutazione dei modelli

2.5.1 Introduzione

Avendo finito la fase di generazione dei modelli si passa ora alla loro valutazione per scegliere quali modelli risultano essere i più performanti. Questa fase risulta particolarmente complessa in questo caso di studio, in quanto gli algoritmi utilizzati sono modelli non supervisionati la cui valutazione da parte di un umano risulta essere sempre problematica.

In prima battuta si è pensato di valutare direttamente l'embedding sui titoli, calcolando una misura di similarità tra i vettori che rappresentano i titoli. Questa misura similarità sarebbe dovuta essere confrontata con la similarità 'vera' stabilita dagli esperti di dominio. Il problema principale di questo approccio risiede nello stabilire questa similarità 'vera', infatti il concetto di similarità è estremamente vasto e pur sottoponendo ad un esperto di dominio due titoli la valutazione empirica risulta assai complicata. Inoltre, questa valutazione potrebbe essere estremamente variabile in base alla diversa sensibilità degli esperti, che potrebbero dare un peso estremamente differente a fattori che a loro parere sono più o meno determinanti affinché due testi siano simili.

Per le ragioni sopra riportate si è deciso di procedere con un diverso approccio per la selezione dei modelli. Invece che i titoli, sono stati valutati gli embedding creati dai modelli delle parole chiave. Sono state selezionate 30 parole chiave rappresentative del campione, si sono generate tutte le coppie possibili e di queste e stata richiesta una valutazione qualitativa di dipendenza agli esperti di dominio. Il vantaggio di questa procedura risiede nel fatto che due parole sono più facilmente confrontabili da parte di una persona, possedendo esse meno sfaccettature rispetto a due coppie di titoli.

Infine, viene calcolata la correlazione tra le similarità ottenute tramite gli embedding e le similarità vere ottenute tramite il parere degli esperti di dominio. Il modello selezionato sarà il modello che ha ottenuto il grado di correlazione più alto tra quelli significativi.

2.5.2 Selezione parole chiave

Le parole chiave sono le parole decise dall'autore che riassumono in se stesse gli argomenti ed i campi trattati dalla pubblicazione. Pertanto, risultano essere particolarmente adatte a rappresentare in maniera sintetica l'ambito inerente al corrispondente articolo. In base a questo principio, è coerente ritenere che un modello sia quanto più performante quanto più siano umanamente sensati gli embedding delle parole chiave da esso generati.

La selezione delle parole chiave è avvenuta nel seguente modo. In primo luogo, sono stati eseguite sulle parole chiave quelle fasi di preprocessing necessarie a renderle compatibili con i modelli sviluppati.

In secondo luogo, sono state eliminate le parole chiave non presenti nel vocabolario dei titoli, questo poiché esse non potevano essere rappresentate da modelli Doc2vec e Sent2vec che, essendo stati addestrati unicamente su questo corpus, non sono in grado di rappresentare parole estranee allo stesso.

Successivamente si è creata una distribuzione di parole in base alla loro numerosità. In questo modo sono state contate quante volte le parole chiave vengono scelte dagli autori. Da questa distribuzione di 'popolarità' delle parole chiave sono stati scelti tre intervalli da cui estrarre casualmente 10 parole in ciascuno, di cui successivamente si valuteranno gli embedding. Gli intervalli di frequenza scelti sono i seguenti: le parole chiave con una frequenza superiore al 95%, le parole chiave che sono comprese nell'intervalli di frequenza tra il 75% e l'80% e le parole chiave che sono comprese nell'intervallo di frequenza tra il 50% e il 55%.

Questi tipi di intervalli sono stati scelti per rappresentare un campione il più rappresentativo possibile tra le parole chiave presenti. Si può notare che non è presente un intervallo per selezionare le parole chiave più rare, questo perché a parole chiave meno frequenti corrispondono parole meno frequenti all'interno dei titoli, pertanto parole in cui i modelli fanno più fatica a costruire un embedding significativo.

In conclusione, le parole selezionate sono state le seguenti:

- Top 95%: economics and econometrics, innovation, complex dynamics, bifurcation, conjoint analysis, performance, statistics and probability, market driven management,

global markets, multistability

- 75-80%: experiment, big data, gini index, global market, efficiency, communication, satisfaction, econometric models, ownership, inter rater agreement
- 50-55%: asymmetries, corporate responsibility, creative industries, banks, multi-way factor analysis, proportional election, heterogeneous players, open innovation, competitive advantage, endogenous preferences

2.5.3 Gold Benchmark

Stabilite a questo punto le parole chiave da utilizzare si è passato alla creazione del gold benchmark. Ciò consiste nell'etichettatura del confronto tra parole chiave eseguita da un esperto di dominio. Dalle 30 parole chiave selezionate sono state create le coppie di combinazioni semplici, pertanto sono stati creati 435 confronti tra coppie di parole. Per ciascuna di queste coppie è stato chiesto al professore Pelagatti di valutarne il grado di relazione mediante la scelta di 5 possibili intensità di correlazione: Non Correlate, Debolmente Correlate, Correlate, Fortemente Correlate, Sinonimi. La costruzione di questo Gold Benchmark è stata fatto mediante un Google Spredsheet mostrato in Tabella 2.3.

Keywords	**Relazione tra keywords**
economics and econometrics - innovation	Debolmente correlate
economics and econometrics - complex dynamics	Fortemente correlate
economics and econometrics - bifurcation	Correlate
economics and econometrics - conjoint analysis	Correlate
economics and econometrics - performance	Debolmente correlate
economics and econometrics - statistics and probability	Correlate
economics and econometrics - market driven management	Non correlate
economics and econometrics - global markets	Non correlate
economics and econometrics - multistability	Debolmente correlate

Tabella 2.3 Prime righe del confronto tra parole operato dal professore Pelagatti

2.5.4 Metriche di valutazione

Per scegliere il miglior modello è stato prima necessario per ogni singolo modello generare gli embedding delle 30 parole chiavi e da questi embedding calcolare una misura di similarità tra le 435 coppie.

Come misura di similarità si è scelto di utilizzare il coseno di similitudine, definito dalla seguente formula:

$$similarity = \cos(\theta) = \frac{A \cdot B}{||A||\,||B||} \tag{2.9}$$

2.9 Coseno di similitudine

dove A e B sono i due vettori numerici che si vogliono confrontare. Il valore della similitudine così costruito assume un valore compreso tra -1 e +1, dove -1 indica una corrispondenza esatta ma opposta e +1 indica che i due vettori sono uguali.

Già da una prima analisi dei risultati di similarità si è notato che alcuni modelli non erano stati in grado di ottenere risultati significativi, infatti i modelli Sent2vec creati con funzione di perdita softmax e negative sampling, riportavano per ciascuna coppia il valore pari ad 1, pertanto sono stati esclusi da ulteriori valutazioni.

Per quanto riguarda le classi definite durante la creazione del Gold Benchmark, esse sono state rimappate su scale numerica con i seguenti valori: alla classe 'Non Correlate' è stato assegnato valore 0, a 'Debolmente Correlate' 0.25, a 'Correlate' 0.5, a 'Fortemente Correlate' 0.75, a 'Sinonimi' 1. Questa trasposizione su scala numerica è stata necessaria per garantire un confronto diretto con le similarità calcolate con il coseno di similitudine.

Il confronto per la scelta dei modelli è avvenuto mediante l'utilizzo di due coefficienti di correlazione. Il primo, l'indice di correlazione lineare di Pearson, dati X e Y due vettori numerici, è calcolato come segue:

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} \tag{2.10}$$

2.10 Coefficiente di correlazione lineare di Pearson

dove ρ_{XY} è la covarianza tra X e Y e σ_X e σ_Y sono le due deviazioni standard. L'indice è compreso tra -1 e 1, dove assume valore di -1 in presenza di perfetta correlazione lineare negativa e valore +1 in presenza di perfetta correlazione lineare positiva.

Il secondo indice utilizzato, l'indice di correlazione per ranghi di Spearman è semplicemente dato dall'indice di correlazione lineare di Pearson applicato non direttamente sui

vettori X e Y, ma bensì sui rispettivi ranghi rg_X e rg_Y. Pertanto esso è calcolato dalla seguente formula:

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X} \sigma_{rg_Y}} \tag{2.11}$$

2.11 Coefficiente di correlazione per ranghi di Spearman

L'indice è compreso tra -1 e 1, dove assume questi valori quando ognuna delle variabili è una perfetta funzione monotona (crescente o descrescente) dell'altra.

La differenza tra i due indici di correlazione risiede nel fatto che l'indice di Spearman è una misura statistica non parametrica che misura la relazione tra due variabili, mentre il coefficiente di correlazione di Pearson misura la relazione lineare tra due variabili.

2.5.5 Scelta modello

I risultati delle metriche di correlazione presentate sono esposti nella Tabella 2.4. La tabella risulta essere ordinata in maniera decrescente in base alla correlazione di Spearmen. Come si può notare il modello che risulta essere più performante è il modello Sent2vec costituito con i trigrammi, lo hierarcical softmax e 300 come dimensione di embedding.

Nonostante sia il modello migliore esso non raggiunge un livello di correlazione elevata, solo il 21% per Spearman ed il 24% per Pearson, entrambe le misure sono significative. Questo risultato non particolarmente confortante è un chiaro segnale di come sia difficile con una quantità di dati scarsi generare degli embedding di buona qualità.

Per quanto riguarda il confronto con gli altri modelli di Sent2vec si può notare una tendenza di incremento della correlazione all'aumentare del numero di n-grammi utilizzati. Inoltre, c'è una lieve predilezione per spazi vettoriali più contenuti.

Degli altri modelli solo SBERT risulta avere una correlazione ritenuta significativa fissato un livello di significatività del 10% per quanto riguarda entrambe le correlazioni, ma comunque con un'intensità di correlazione lieve intorno al 10%. Invece, la maggior parte dei modelli di Doc2vec risultano essere non correlati con il gold benchmar, ad esclusione dei modelli generati con gli unigrammi e l'architettura PV DBOW che risultano addirittura correlati negativamente con il gold benchmark.

Modelli	Cor. Spear.	pvalue Spear.	Cor. Pear.	pvalue Pear.
sent2vec_tri_hs_300	0.208	0.000	0.244	0.000
sent2vec_tri_hs_500	0.196	0.000	0.229	0.000
sent2vec_tri_hs_400	0.192	0.000	0.216	0.000
sent2vec_bi_hs_500	0.110	0.021	0.134	0.005
sent2vec_bi_hs_300	0.110	0.022	0.133	0.006
sent2vec_bi_hs_400	0.104	0.031	0.131	0.006
sbert	0.093	0.053	0.107	0.025
Universal_sent_enc	0.080	0.097	0.066	0.168
sent2vec_uni_hs_300	0.050	0.298	0.079	0.100
sent2vec_uni_hs_500	0.026	0.589	0.055	0.254
sent2vec_uni_hs_400	0.022	0.640	0.051	0.288
doc2vec_bi_dbow_500	0.009	0.855	0.020	0.672
doc2vec_bi_dm_300	0.002	0.969	-0.002	0.964
doc2vec_tri_dbow_300	-0.008	0.867	-0.016	0.741
doc2vec_tri_dm_400	-0.013	0.780	0.030	0.532
doc2vec_bi_dm_400	-0.013	0.779	-0.001	0.986
doc2vec_tri_dbow_500	-0.014	0.775	-0.009	0.844
doc2vec_uni_dm_300	-0.025	0.604	0.027	0.577
doc2vec_uni_dm_400	-0.030	0.534	0.018	0.710
doc2vec_bi_dbow_500	-0.031	0.524	-0.020	0.681
doc2vec_tri_dbow_400	-0.032	0.505	-0.022	0.645
doc2vec_tri_dm_300	-0.036	0.453	-0.009	0.854
doc2vec_tri_dm_500	-0.047	0.331	-0.030	0.527
doc2vec_uni_dm_500	-0.047	0.325	0.001	0.988
doc2vec_uni_dbow_300	-0.063	0.193	-0.085	0.078
doc2vec_bi_dbow_400	-0.070	0.143	-0.055	0.254
doc2vec_bi_dbow_300	-0.086	0.073	-0.069	0.153
doc2vec_uni_dbow_400	-0.109	0.024	-0.093	0.052
doc2vec_uni_dbow_500	-0.115	0.017	-0.120	0.012

Tabella 2.4 Tabella con correlazioni tra modelli e Gold Benchmark

3

Risultati

3.1 Visualizzazioni

3.1.1 Riduzione dimensionalità

Avendo ottenuto i modelli migliori per ciascun tipo di architettura, si è ora in possesso di una rappresentazione in uno spazio vettoriale multidimensionale di ciascun titolo. Il problema di questa rappresentazione risiede nel fatto che il numero di dimensioni varia tra i diversi modelli e soprattutto è difficile per un essere umano valutare intuitivamente le relazioni che intercorrono tra vettori numerici di elevata dimensione.

Per risolvere questo problema e per fornire una visualizzazione bidimensionale facilmente interpretabile, è stata utilizzata una tecnica di riduzione della dimensionalità. La tecnica in questione è t-distributed stochastic neighbor embedding (t-SNE), un algoritmo di riduzione della dimensionalità non lineare che genera uno spazio a dimensione ridotta in cui sono vicini i dati simili tra loro e lontani i dati dissimili, pertanto risulta essere particolarmente adatto al nostro contesto.

L'algoritmo funziona come segue, dato un insieme di N oggetti $x_1, \ldots, x_N$ in uno spazio

ad alta dimensionalità, t-SNE costruisce una distribuzione delle probabilità p_{ij}, simmetriche nelle due variabili e proporzionale alla similarità tra i punti x_i e x_j, definita come:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} \tag{3.1}$$

3.1 Probabilità di similarità nello spazio iniziale

dove:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i)}{\sum_{k \neq i} \exp(-||x_i - x_j||^2/2\sigma_i)} \tag{3.2}$$

3.2 Distribuzione condizionale della probabilità di similarità

L'ampiezza del parametro σ_i è scelta in modo tale per cui la perplessità della distribuzione condizionale uguagli un valore di perplessità fornito come iperparametro dell'algoritmo.

t-SNE cerca di costruire una mappa d-dimensionale $y_1, \ldots, y_N$ (con $y_i \in \mathbb{R}^d$) in cui i punti riflettano al meglio la similarità p_{ij} nello spazio di partenza. A tale scopo, la similarità q_{ij} tra due punti y_i ed y_j nello spazio a dimensionalità ridotta viene definita come:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i} (1 + ||y_i - y_j||^2)^{-1}} \tag{3.3}$$

3.3 Similarità tra i e j nello spazio a dimensionalità ridotta

In questo caso è stata usata una distribuzione t-Student con un grado di libertà, le cui code pesanti consentono di modellare meglio la dissimilarità tra oggetti distanti. La posizione y_i dei punti nello spazio a dimensione ridotta è quindi calcolata minimizzando tramite discesa del gradiente la divergenza di Kullback-Leibler della distribuzione Q rispetto a P:

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3.4}$$

3.4 Divergrenza di Kullback-Leibler

L'uso della divergenza di Kullback-Leibler come funzione obiettivo consente di avere penalità elevate se punti vicini nello spazio originale (p_{ij} elevato) vengono considerati lontani nello spazio a dimensionalità ridotta (q_{ij} piccolo), mentre al viceversa corrisponde una penalità ridotta, tendendo quindi a preservare la struttura locale della distribuzione dei punti. Il risultato è una mappa a bassa dimensionalità che riflette le similarità tra i punti nello spazio ad alta dimensionalità.

L'algoritmo così definito è stato utilizzato per ridurre la dimensionalità degli embedding sia dei titoli che delle parole chiave, pertanto i vettori numerici rappresentanti i dati testuali sono stati mappati in due dimensioni che mantenessero le relazioni di similarità. La scelta di usare due sole dimensioni risulta essere la più logica al fine di creare una visualizzazione chiara e interpretabile degli embedding.

3.1.2 Visualizzazione titoli

In Figura 3.1 sono rappresentati gli embedding dei titoli secondo il modello SBERT. Essendo una visualizzazione di Tableau essa è interattiva e al seguente link è possibile esplorarla nel suo completo. Per ogni specifico modello è possibile scegliere uno o più autori e confrontare la rappresentazione bi-dimensionale delle relazioni tra i titoli rilevate dai vari modelli. Quanto più un titolo sarà vicino ad un altro quanto più i loro embedding saranno simili, pertanto titoli vicini tra loro risultano essere in maggiore relazione rispetto a titoli lontani.

In particolare andando ad analizzare questo caso specifico in cui si confrontano le pubblicazioni del professore Pelagatti con quelle del professore Manera si possono evidenziare diversi trend. In primo luogo, si nota come il modello abbia catturato un grosso cluster di paper nella regione in basso a destra tutti relativi a studi sul petrolio, sul suo prezzo e sul prezzo in generale dell'energia. Nella zona in alto a destra, invece, il modello rappresenta come simili diversi paper che analizzano casi di studio italiani, mentre leggermente più in zona centrale del primo quadrante risiedono paper trattanti argomenti ambientali. Il terzo quadrante sembra essere connotato da un approccio maggiormente teorico, con la presenza in basso a sinistra di un cluster di pubblicazioni facenti riferimento test statistici non-nested.

Oltre alle visualizzazzioni per singoli modelli È stata creata anche una visualizzazione congiunta di tutti i modelli migliori, che può essere esplorata al seguente link. In essa è

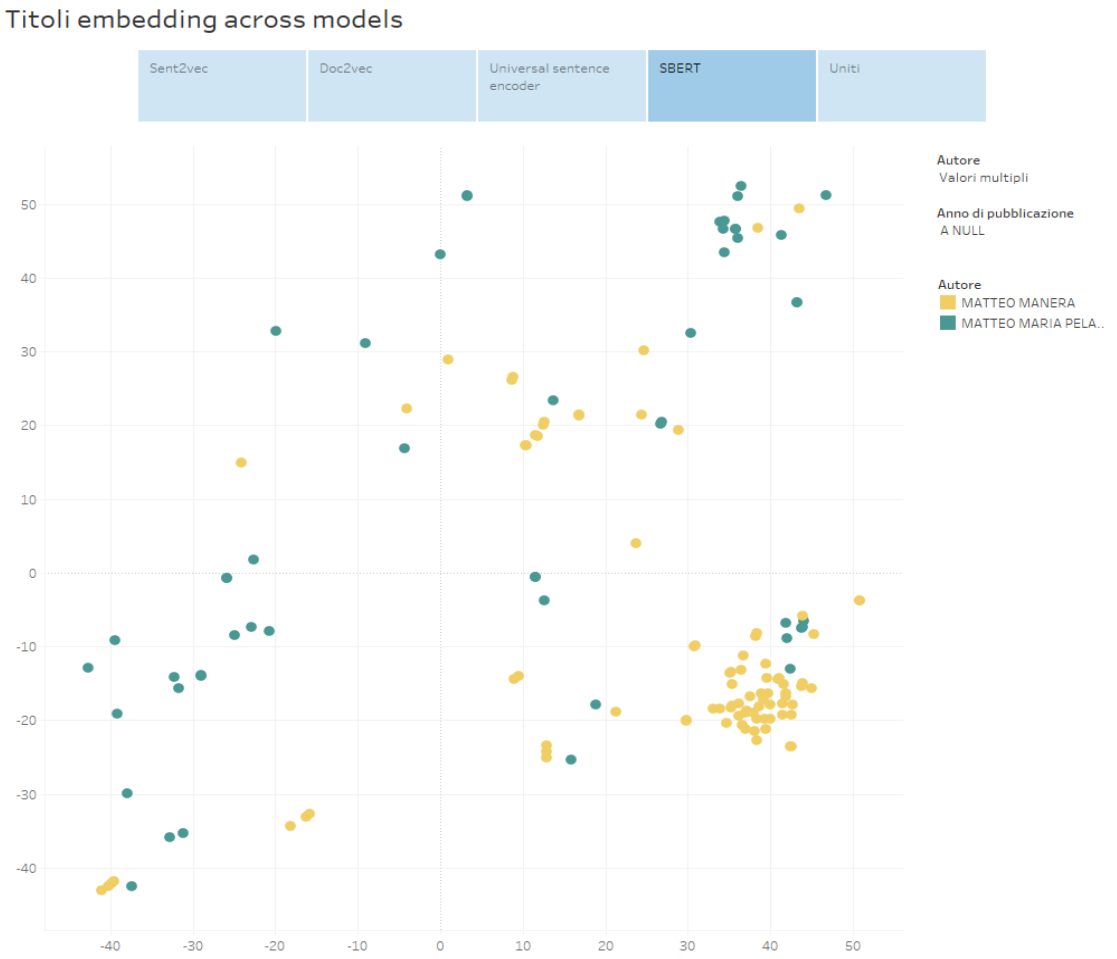

Figura 3.1 Rappresentazione dell'embedding dei titoli ottenuto con SBERT

possibile selezionare se usare uno o più modelli, quali anni tenere in considerazione e quali professori confrontare. Tale visualizzazione permette un ulteriore confronto diretto tra i modelli, questa volta però sugli embedding dei titoli anziché su quello delle parole.

Le sopracitate visualizzazioni sono state sottoposte alla valutazione del professore Pelagatti. Il giudizio sugli embedding dei vari modelli è stato negativo, infatti il professore Pelagatti non è riuscito a riscontrare nelle relazioni catturate dai modelli, le relazioni reali che intercorrono tra i vari titoli. Questo fallimento negli embedding dei titoli può prevalentemente essere imputato alla ristrettezza del dataset. Infatti tutti i modelli presentati si basano sulle reti neurali artificiali, architetture che necessitano di un'elevata quantità di dati per raggiungere risultati di qualità. Inoltre, può avere avuto un ruolo determinante anche l'estrema specificità dei dati analizzati, infatti essi appartengono allo specifico gruppo di articoli scientifici ed

all'ancor più ristretto ambito di articoli scientifici economici e statistici.

3.1.3 Visualizzazione Parole Chiave

Per la visualizzazione delle parole chiave, riportata in Figura 3.2, si è usato l'embedding Sent2vec risultato migliore proprio per quanto riguarda le parole chiave. Nell'immagine viene anche riportata la frequenza con cui sono state ripetute le stesse parole chiave. Si può notare come il modello ponga vicine parole chiave in cui è presente un termine in comune, però fallisca completamente nel ricreare un ambito globale in cui identificare settori diversi. Come detto in precedenza sembrerebbe che la ridotta numerosità dei dati e l'estrema specificità dell'ambito in questione non abbia consentito di creare embedding di qualità sufficiente.

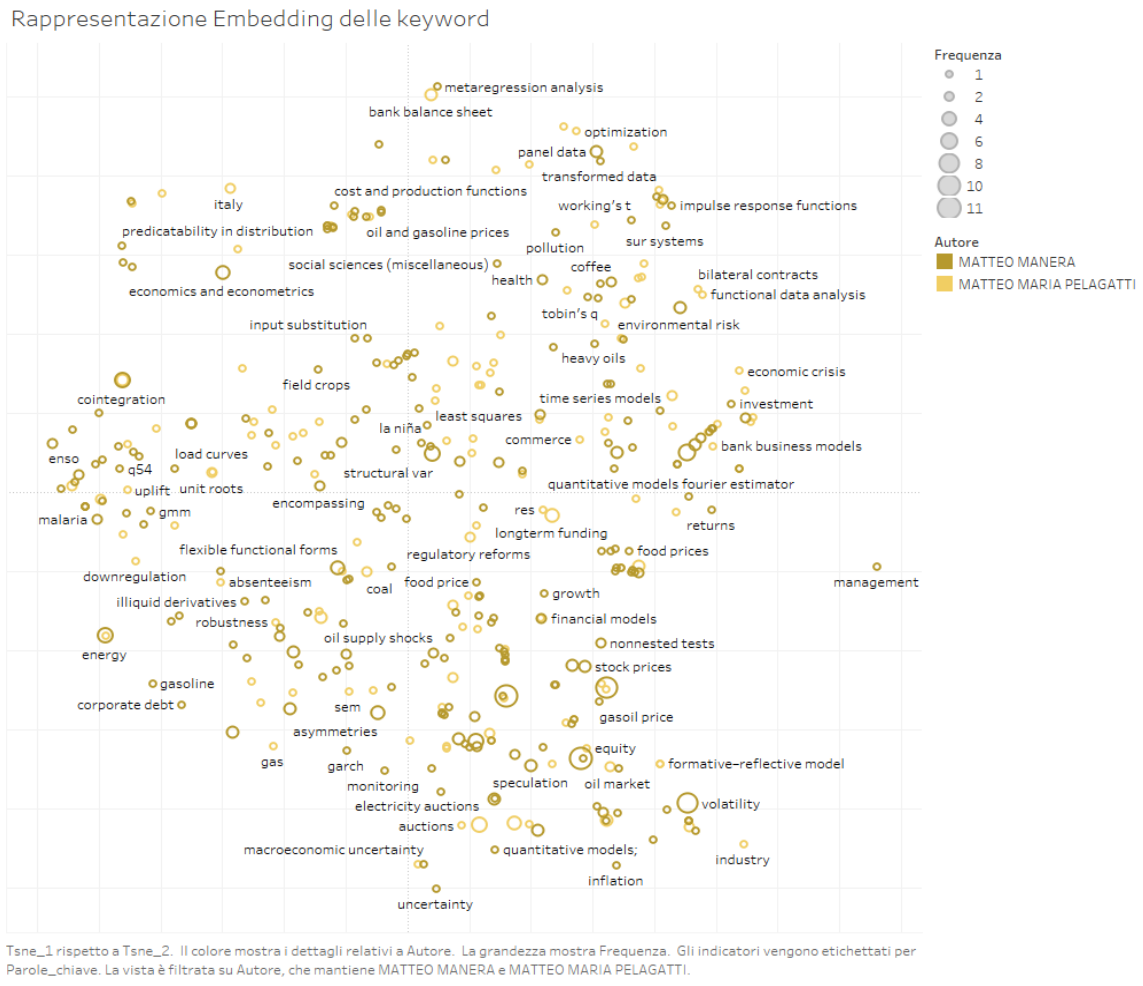

Figura 3.2 Rappresentazione dell'embedding delle parole chiave ottenuto con Sent2vec

3.2 Valutazione quantitativa

3.2.1 Generazione similarità

Fino a questo momento si sono andati a valutare i singoli embedding generati da titoli o parole chiave, però nelle realtà risulta essere più conveniente avere una misura aggregativa di comparazione tra i diversi autori. Lo scopo di quest'ultima fase è di ottenere una metrica che permetta di valutare la similarità tra due diversi autori. Per questa ragione sono state implementate tre metriche di similarità.

Le prime due sono indipendenti dagli embedding generati e sono concepite per valutare la similarità tra autore in base alle parole chiave, dato che come stabilito nella sezione 3.1 gli embedding per esse non sono risultati qualitativamente accettabili. Esse sono la similarità di Jaccard e la similarità di Jaccard pesata.

La similarità di Jaccard, dati due insiemi campionari A e B, è definita dalla seguente formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.5}$$

3.5 Similarità di Jaccard

La similarità di Jaccard pesata, come da nome, pesa anche in base alla frequenza degli elementi presenti negli insiemi campionari. In primo luogo, viene creata l'unione tra A e B. Da essa vengono generati due vettori numerici $x = (x_1, \ldots, x_n)$ e $y = (y_1, \ldots, y_n)$ che contano per ogni elemento presente nell'unione il numero di volte in cui esso compare rispettivamente nell'insieme A e nell'insieme B. Posto ciò, la similarità di Jaccard pesata è definita come segue:

$$J_W(x,y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)} \tag{3.6}$$

3.6 Similarità di Jaccard pesata

I due indici di similarità variano tra 0 e 1, assumendo valore 0 quando i due insiemi sono completamente dissimili ed 1 quando essi sono completamente simili. Nel nostro caso gli insiemi campionari sono le parole chiave utilizzate dai due autori a confronto.

Il terzo indice per il confronto della similitudine tra autori è dato dalla media dei coseni di similitudine calcolati per ciascuna coppia di titoli possibili. In pratica dati due insiemi campionari A e B, dati dai titoli dei due autori a confronto, viene calcolato il prodotto cartesiano tra A e B e per ogni combinazione risultante viene calcolata la similarità del coseno tra i due embedding a confronto. Infine, di tutte le similarità ottenute viene calcolata la media.

Vale la pena notare che l'indice in questo modo calcolato è una derivazione di un indice di similarità, ma non è esso stesso un indice di similarità in quanto se calcolato per confrontare l'autore con se stesso non dà come risultato 1, bensì una misura che indica la coerenza tra i vari lavori svolti dall'autore. In linea teorica se un autore tratta ambiti vari l'indice assume valori piccoli, se invece l'autore predilige pubblicare all'interno di un ambito particolarmente specifico l'indice assume valori elevati.

3.2.2 Visualizzazioni

Dalle metriche esposte in precedenza sono stati sviluppate due visualizzazioni interattive (link). La prima, in Figura 3.3, si concentra sulle similarità per parole chiave e rappresenta per ogni coppia di autori selezionata il valore assunto dagli indici di Jaccard e i top 10 autori simili agli autori selezionati, avendo così un confronto diretto tra autori, insieme al contesto generale.

Figura 3.3 Confronto similarità per parole chiave tra due autori

La seconda, mostrata in Figura 3.5, è simile al confronto precedente, solo che stavolta l'interesse è rivolta ai titoli, pertanto l'indice di similarità utilizzato è la media dei coseni di similitudine descritta nella sezione precedente.

Questi indici sono stati messi valutati dal professore Pelagatti, il quale ha ritenuto maggiormente efficace l'indici di similitudine di Jaccard. Questo risultato, per il quale una valutazione non basata sugli embedding risulta essere superiore rispetto ad una basata sugli stessi, conferma quanto stabilito tramite le visualizzazioni precedenti, cioè che gli embedding costruiti non sono riusciti a catturare l'informazione testuale necessaria per essere giudicati rappresentativi.

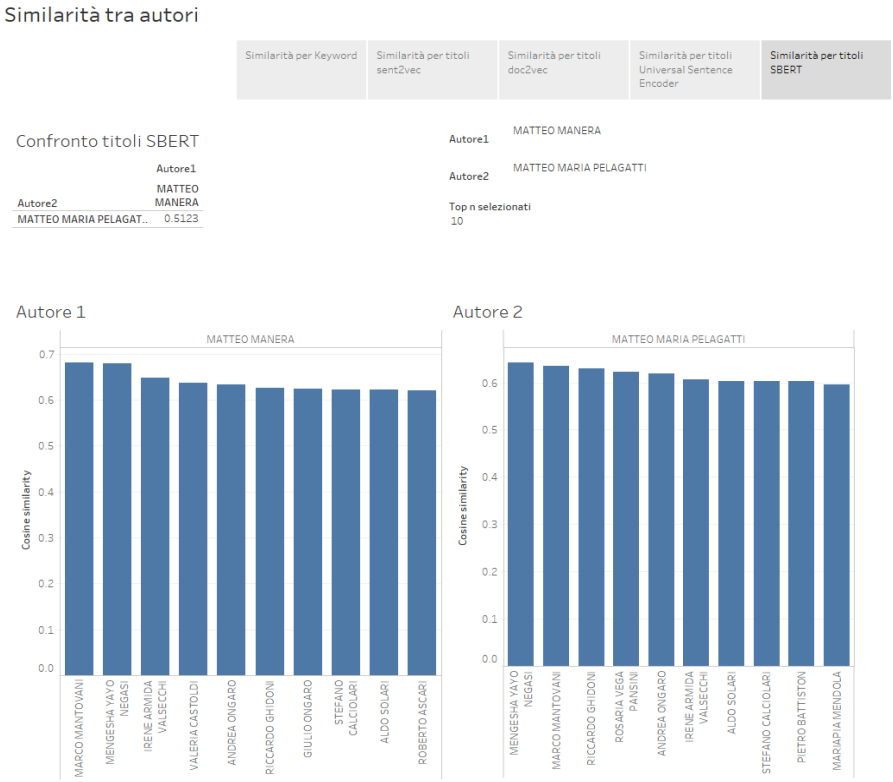

Figura 3.4 Confronto similarità per titoli tra due autori usando l'embedding di SBERT

3.3 Confronto finale per autore

3.3.1 Scaling Multidimensionale

Stabilito dunque che gli indici di similitudine di Jaccard riescono ad essere più precisi nel rappresentare le similitudini tra autori, si è deciso di utilizzarli per creare una visualizzazione finale per confrontare direttamente gli autori. Per fare ciò è stata utilizzata la tecnica dello scaling multidimensionale, che permette, a partire da una matrice di dissimilarità, di fornire una configurazione spaziale delle entità presenti nella matrice.

Più formalmente, data una matrice di dissimilarità D tra n entità, l'obbiettivo dello scaling multidimensionale è quello di trovare una configurazione di punti X tale per cui la matrice $\hat{\Delta}$ renda minima la seguente funzione di perdita:

$$L = \left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} (\hat{\Delta}_{ij} - f(D_{ij}))^2 \right\}^{\frac{1}{2}} \tag{3.7}$$

3.7 Funzione di perdita Scaling Multidimensionale

dove:

- $\hat{\Delta}_{ij} = [(x_i - x_j)^T (x_i - x_j)]^{\frac{1}{2}}$
- $f(\cdot)$ è una funzione monotona che trasforma le dissimilarità in distanze
- w_{ij} sono opportuni pesi per ogni coppia di entità

Nel nostro caso per ottenere la matrice D di dissimilarità partendo dalla matrice di similarità S si è utilizzata la seguente relazione:

$$D = 1 - S \tag{3.8}$$

3.8 Relazione tra similarità e dissimilarità

3.3.2 Visualizzazione

Il risultato finale del confronto tra autori è mostrato in Figura 3.5. Essendo derivato dalle similarità di Jaccard pesate nel grafico risultano essere vicini autori che usano spesso le stesse

parole chiave e lontani autori che usano parole chiave diverse. La rappresentazione riesce a catturare bene le relazioni sulle distanze tra autori, ma fa più fatica a rilevare cluster specifici. Autori vicini sono sì in relazione tra loro, ma le relazioni con autori più lontani non sono ben definite.

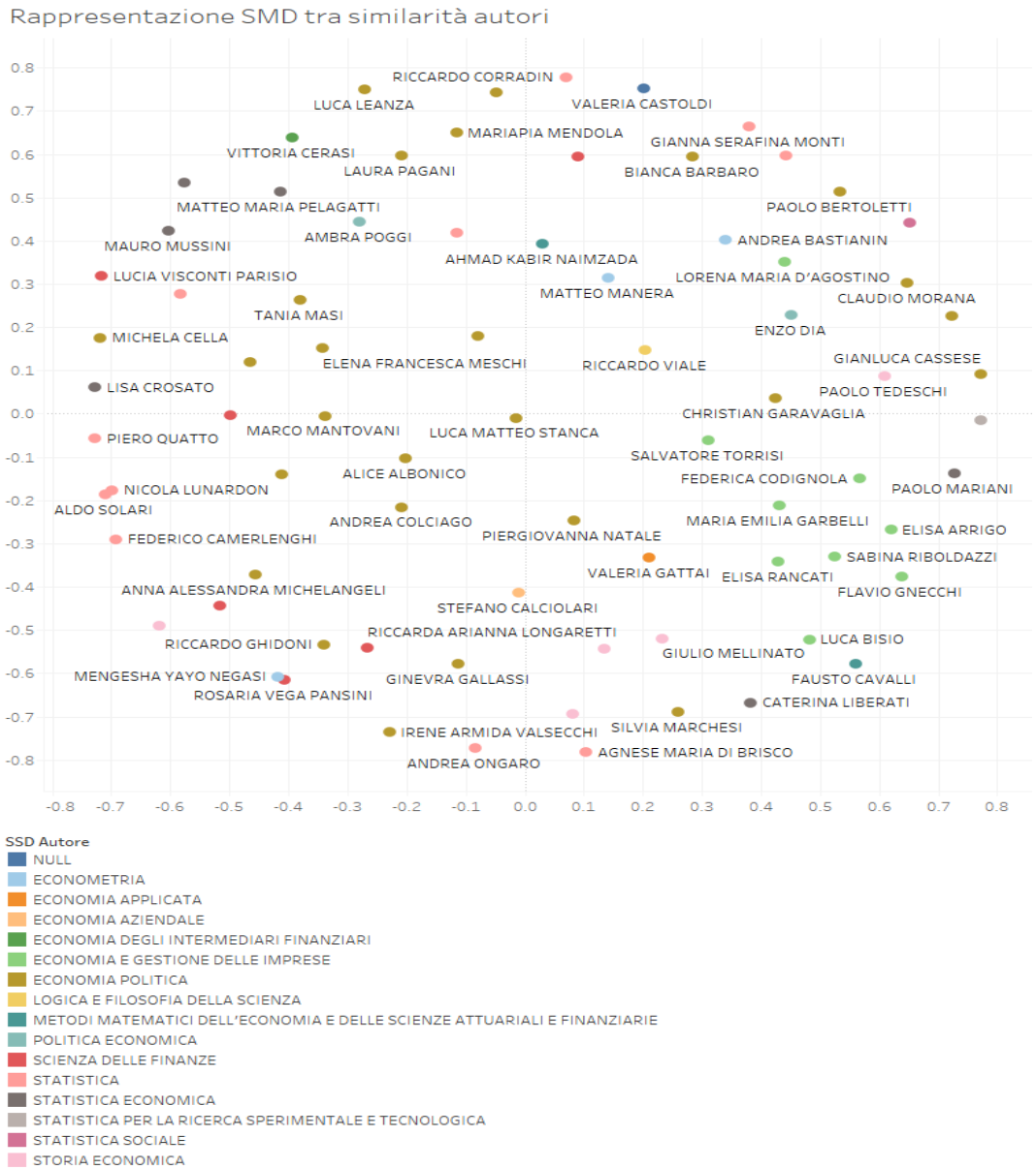

Figura 3.5 Scaling Multidimensionale dell'indice di similarità di Jaccard pesata tra autori

Infatti, lo scaling mutidimensionale ha riprodotto una distribuzione sferica di densità quasi omogenea degli autori. Questo è dovuto al fatto che l'indice di Jaccard pesato assume

un valore positivo per autori che usano le stesse parole chiave e un valore pari a 0 tra tutti gli autori che non hanno mai utilizzato parole chiave in comune. Da ciò deriva il fatto che per ogni autore la maggior parte degli indici di similarità calcolati risulta essere uguale a 0, pertanto il modello, in un contesto in cui tutti gli autori risultano essere simili con pochi e dissimili con molti, rappresenta questo dato con una forma circolare, riuscendo a modellare le distanze tra autori vicini e tenendo il più costante possibili le distanze tra tutti gli autori dissimili.

Per dare un'idea dai possibili cluster che si sarebbero dovuti creare nella figura tramite diversi colori sono evidenziati diversi settori scientifici disciplinari. Va comunque fatta una premessa, non è assicurato che un autore appartenente ad uno specifico settore pubblichi solo all'interno dello stesso. Detto ciò si può notare come si sia formato un cluster ben definito nel quarto quadrante per quanto riguarda il settore "Economia e Gestione delle Imprese". Altri gruppi vengono creati da un piccolo cluster di "Statistica Economica" nel secondo quadrante ed un cluster di storia economica in basso nel quarto quadrante, entrambi con qualche outlier in altre zone del grafico. In generale è preponderante la presenza di autori appartenenti ad "Economia Politica", che variano uniformemente in tutto lo spazio eccezion fatta per il quarto quadrante in cui sono presenti solo in due unità ai bordi dello stesso. Simile ragionamento si può fare per quanto riguarda il settore "Statistica", che oltre ad un cluster di 4 autori nel terzo quadrante in alto a destra, risulta avere autori sparsi nel grafico tranne nel quarto quadrante, che è monopolizzato da "Economia e Gestione delle Imprese".

In conclusione, lo scaling multidimensionale sulle similarità di Jaccard pesate è stato uno strumento mediamente efficace nel rappresentare le relazione tra gli autori. Ad una discreta capacità nel rilevare le relazioni singolari fa da contro altare una mediocre sensibilità nella rivelazione di relazione globali.

4

Conclusioni

4.1 Conclusioni

Alla domanda posta inizialmente se fosse possibile rappresentare in maniera significativa i dati testuali delle pubblicazioni del Dipartimento di Economia, Metodi quantitativi e Strategie d'impresa si è giunti ad una risposta negativa. Gli sforzi protratti durante tutto il lavoro di tesi sono risultati essere vani di fronte ai limiti ed alle criticità che affliggono i modelli basati sul word embedding, debolezze intrinseche della natura delle reti neurali che accomunano la struttura di questi modelli.

Più nel dettaglio, alla ridotta numerosità dei campioni nel dataset ed alla estrema specificità e specializzazione dell'ambito trattato dai dati testuali si possono attribuire le cause della mancata generazione di un embedding di qualità.

Al fallimento di tecniche basate sulle reti neurali artificiale fa da contraltare il relativo successo di algoritmi particolarmente semplici. Le relazioni tra pubblicazioni non individuate dai modelli di word embedding sono state rilevate dalle semplici metriche di similarità di Jaccard. È stato anche possibile a partire da quest'ultime sviluppare una rappresentazione parzialmente valida delle relazioni che intercorrono tra gli autori.

In conclusione, ad un maggiore complessità non necessariamente corrisponde un migliore risultato, pertanto non bisogna bistrattare i risultati ottenuti da modelli semplici, in quanto a volte sono gli unici risultati raggiungibili.

4.2 Possibili sviluppi

I possibili sviluppi riguardano principalmente il superamento delle difficoltà incontrate, superando le limitazioni dovute alla dimensione del dataset originario.

Da un lato è possibile scaricare da scopus i dati mondiali delle pubblicazioni relative all'ambito statistico ed economico, utilizzare questa grande mole di dati per ottenere embedding significativi. Purtroppo, per far ciò è necessario avere a disposizione hardware superiori a quelli disponibili gratuitamente online.

Da un altro lato è possibile concentrare l'attenzione non solo sul Dipartimento di Economia, Metodi quantitativi e Strategie d'impresa ma su tutti i dipartimenti dell'Università deli Studi di Milano-Bicocca. Ciò non solo consentirebbe di avere un maggiore quantitativo di dati, ma anche di avere accesso ad una maggiore diversificazione fra gli stessi. Sviluppare un modello di questo tipo permetterebbe di analizzare le relazioni che sussistono tra i vari dipartimenti, con il chiaro obbiettivo di accrescere rapporti interdisciplinari tra i dipartimenti, in modo tale da creare un'università più coesa.

Infine, un obiettivo da raggiungere nei prossimi anni, potrebbe essere quello di sviluppare uno spazio liberamente accessibile a ciascun professore, in cui sia possibile un confronto automatico ed efficace tra pubblicazioni, in grado anche di suggerire in base ad un'idea iniziale di titolo di ricerca, chi ha affrontato temi simili in passato, venendo così a fornire un indizio per una probabile collaborazione di ricerca.

Elenco delle figure

Bibliografia

[1] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, 2015.

[4] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

[7] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.