Università degli studi di Modena e Reggio Emilia

Dipartimento di Ingegneria "Enzo Ferrari"

Corso di Laurea Magistrale in Ingegneria Informatica

Utilizzo di tecniche di Topic Modelling applicate a comunicati stampa

Relatore:

Prof. Francesco Guerra

Candidato:

Andrea Spinazzola

Correlatore:

Ing. Francesco Del Buono

Ing. Matteo Paganelli

Anno Accademico 2019/2020

Indice

1 Introduzione **2**

2 Topic Modelling **5**

2.1 Definizione di Topic Modelling 5

2.2 Algoritmi classici di Topic Modelling 7

2.2.1 TF-IDF 8

2.2.2 Latent Semantic Analysis 9

2.2.3 Non-Negative Matrix Factorization 11

2.2.4 Probabilistic Latent Semantic Analysis 12

2.2.5 Latent Dirichlet Allocation 13

2.3 Metodi di Embedding 18

2.3.1 Word2vec 19

2.3.2 Doc2vec 23

2.3.3 Transformer 24

2.3.4 BERT 26

3 Definizione del problema e degli strumenti utilizzati **29**

3.1 Definizione del problema 29

3.2 Strumenti e librerie 30

3.2.1 Scrapy 31

3.2.2 Pandas 31

3.2.3 NumPy 32

3.2.4 HuggingFace . . . 32
3.2.5 SentenceTransformer . . . 33
3.3 Dataset . . . 34
3.4 Valutazione . . . 40
3.4.1 Adjusted Rand index . . . 40
3.4.2 Normalized Mutual Information . . . 41
3.4.3 Coerenza . . . 42

4 Prima soluzione **44**
4.1 Top2vec . . . 44
4.2 Implementazione . . . 47
4.2.1 Embedding . . . 48
4.2.2 Clustering . . . 52
4.2.3 Estrazione dei topic . . . 56
4.2.4 Fine-tuning . . . 58
4.3 Risultati . . . 60

5 Seconda soluzione **65**
5.1 Implementazione . . . 65
5.1.1 Preprocessing ed embedding . . . 66
5.1.2 Riduzione della dimensione e clustering . . . 71
5.1.3 Estrazione delle parole . . . 73
5.2 Risultati . . . 75

6 Conclusioni **80**

Capitolo 1

Introduzione

Ogni giorno, enormi moli di dati testuali vengono generati e processati in qualsiasi settore dell'informatica. Tweet, query di ricerca, articoli di giornale e post sui social network sono solamente alcuni degli esempi di informazioni testuali che leggiamo od utilizziamo quotidianamente. Una delle sfide che la ricerca tecnologica si è posta, a partire già dagli anni '50, è quella di riuscire ad estrarre conoscenza da questi dati in maniera automatizzata. E' nato quindi il settore del Natural Language Processing (NLP), un campo di studi che unisce informatica, intelligenza artificiale e linguistica, con l'obiettivo di sviluppare tecniche e strumenti per analizzare e comprendere dati prodotti in linguaggio naturale. La ricerca ha orientato molto i suoi sforzi su queste tematiche, in quanto i dati testuali sono ormai ovunque nella nostra vita di tutti i giorni.

Alcune delle tecniche utilizzate dal NLP sono definite supervisionate, dato che richiedono di avere a priori un dataset di testi già annotati manualmente per imparare a classificare dei nuovi documenti. Altre tecniche sono invece definite non supervisionate perché, al contrario, non necessitano di avere un dataset di allenamento ma riescono ad estrarre informazioni significative solamente osservando la struttura dei dati. Uno dei task non supervisionati più importanti e popolari per l'analisi di dati testuali è sicuramente quello del topic modelling, che consiste nel capire ed estrarre gli argomen-

ti tematici di cui una raccolta di documenti tratta. I campi di applicazione del topic modelling sono molti e diversificati. Esso viene utilizzato, per esempio, quando si ha una quantità molto grossa di dati testuali che non potrebbero essere letti ed ordinati da una persona, per filtrare solamente dei documenti in base a determinati interessi oppure per raggruppare testi simili.

Il topic modelling è un campo molto vasto, in cui esistono numerose tecniche differenti tra di loro per risolvere il problema dell'individuazione degli argomenti trattati in un corpus. Questo lavoro di tesi si inserisce all'interno del settore del topic modelling con due obiettivi principali. Il primo è quello di analizzare lo stato dell'arte attuale del topic modelling, partendo dagli algoritmi più classici fino ad arrivare ad analizzare le tecniche più moderne ed innovative. Il secondo è quello di risolvere il problema del topic modelling applicato ad un contesto ben preciso, ovvero quello dell'estrazione degli argomenti trattati da un dataset composto da comunicati stampa redatti dal governo inglese.

Il primo capitolo tratterà il problema del topic modelling nel dettaglio, fornendo una panoramica sullo stato dell'arte e delle tecniche più importanti. Verrà posta particolare attenzione alla descrizione degli algoritmi e delle tecnologie utilizzate durante l'implementazione della soluzione analizzata nei capitoli successivi.

Nel secondo capitolo verrà presentato il problema su cui si è basato il lavoro di tesi. Verrà fornita una descrizione dei dataset e delle metriche di valutazione utilizzate per i test.

Nel terzo capitolo verrà proposta una prima soluzione al problema dell'estrazione dei topic in un dataset costituito da comunicati stampa. Verranno descritte nel dettaglio l'architettura e le tecniche utilizzate. Verranno poi mostrati e discussi i risultati ottenuti.

Nel quarto capitolo verrà presentata una seconda soluzione al problema con un approccio diverso. Anche in questo caso verrà approfondita nel dettaglio l'architettura

utilizzata e saranno mostrati i risultati ottenuti.

Nelle conclusioni si farà un confronto tra i due approcci utilizzati, osservando i loro risultati, e si ragionerà su possibili sviluppi futuri che possano integrare e migliorare la soluzione.

Capitolo 2

Topic Modelling

Questo capitolo si concentra sul problema del topic modelling in generale. Verrà fornita la sua definizione e verrà fatta una panoramica sullo suo stato dell'arte, mostrando una classificazione delle tecniche disponibili e descrivendo nel dettaglio gli algoritmi utilizzati successivamente per implementare la soluzione al problema dell'estrazione dei topic in un dataset di comunicati stampa.

2.1 Definizione di Topic Modelling

Il Topic Modelling viene definito da David M. Blei nell'articolo *Probabilistic Topic Models* [15] come un insieme di algoritmi che permettono di scoprire ed annotare la struttura tematica di una collezione di documenti. Sono metodi statistici che analizzano le sequenze di parole dei testi e scoprono i temi che ricorrono in essi, come sono connessi l'uno con l'altro e come cambiano nel tempo. Un metodo di topic modelling è quindi una tecnica di machine learning non supervisionato che permette di analizzare un insieme di dati testuali non etichettati, di individuare dei pattern all'interno di essi e di raggruppare automaticamente delle parole che meglio descrivono gli argomenti trattati nel corpus. Alla fine di un processo di topic modelling si ottengono quindi:

1. Una lista di topic di cui i documenti trattano

2. Dei set di documenti raggruppati in base al topic di cui trattano

Un topic od un argomento, infatti, non è altro che un insieme di parole che spesso vengono menzionate insieme all'interno di un set di documenti. Ad esempio, se in un testo osservo parole come "insegnante, studenti, educazione", so che probabilmente verrà menzionata anche la parola "scuola". In particolare, dato il contesto, la probabilità di osservare la parola "scuola" sarà molto più alta di quella associata ad una qualsiasi altra parola, come ad esempio "sport". Un buon algoritmo di topic modelling, quindi, ha l'obiettivo di formare dei gruppi di parole coerenti tra di loro, ognuno dei quali rappresenta una tematica specifica. Inoltre, dopo aver imparato gli argomenti latenti a partire da una raccolta di testi, ogni volta che abbiamo un documento nuovo possiamo stimare la probabilità che questo sia associato ad ognuno di essi.

E' importante evidenziare che le tecniche di topic modelling sono tutte non supervisionate, differenziandosi da quelle di topic classification che richiedono invece a priori un dataset di allenamento. Il topic modelling e il topic classification sono quindi due approcci completamente diversi e si sceglie di utilizzare uno o l'altro metodo a seconda di diversi fattori.

Gli algoritmi di topic modelling vengono utilizzati quando non si ha a disposizione dei dati già classificati o quando non si hanno le risorse per generarli manualmente. Tuttavia essi necessitano di dati con una qualità molto alta, ottenuta anche tramite tecniche di pre-processing che trasformano i testi in modo da renderli successivamente meglio interpretabili dagli algoritmi di NLP. Alla fine del processo di topic modelling si ottiene una collezione di documenti che sono stati raggruppati insieme in base a degli argomenti non conosciuti a priori e degli insiemi di parole ed espressioni che sono stati utilizzati per ottenere queste relazioni.

Gli algoritmi di topic classification, invece, necessitano di un dataset contenente testi già suddivisi per argomenti e lo utilizzano per imparare a classificare dei nuovi

documenti sulla base dei topic conosciuti a priori. Questi metodi richiedono quindi più dati di partenza ma permettono di ottenere modelli che classificano testi sconosciuti molto più accuratamente.

Vengono di seguito definite la notazione e la terminologia utilizzata successivamente per descrivere i metodi di topic modelling analizzati:

- Una *parola* è l'unità di base dei dati discreti, definita all'interno di un dizionario con indici $\{1, ..., V\}$. Spesso le parole sono rappresentate nel seguente modo: la parola w è un vettore di lunghezza pari alla dimensione del dizionario e tale che $w_v = 1$ e $w_u = 0$, con v indice della parola nel dizionario e $u \neq v$
- Un *documento* è una sequenza di N parole definita come $w = (w_1, w_2, ..., w_N)$ dove w_n è la n-esima parola della sequenza
- Un *corpus* è una collezione di M documenti definito come $D = \{d_1, d_2, ..., d_M\}$

2.2 Algoritmi classici di Topic Modelling

Durante il lavoro di tesi è stato cercato di analizzare quale è stata l'evoluzione delle tecnologie per il topic modelling e come esse sono cambiate nel tempo. Si è partiti dall'approfondire le tecniche più classiche e fondamentali di topic modelling, come LDA e LSA, e si è arrivati a studiare approcci che utilizzano invece tecnologie più moderne ed in particolare il deep learning. Per quanto riguarda i metodi i più classici si è assistito ad una costante evoluzione nel corso degli anni che ha portato ad ottenere algoritmi sempre più performanti. Si è partiti da modelli più semplici di rappresentazione delle informazioni testuali, come TF-IDF, e si è arrivati a tecniche di machine learning più evolute come con LDA. Nell'articolo *Topic Modeling: A Comprehensive Review* [27] viene indicata una netta classificazione di tutte queste tecniche, come mostrato nella figura 2.1. Si possono distinguere algoritmi non probabilistici, come LSA e NNMF,

basati sul concetto di rappresentazione del corpus tramite una matrice delle occorrenze e metodi probabilistici, come LDA e pLSA, che invece utilizzano modelli generativi.

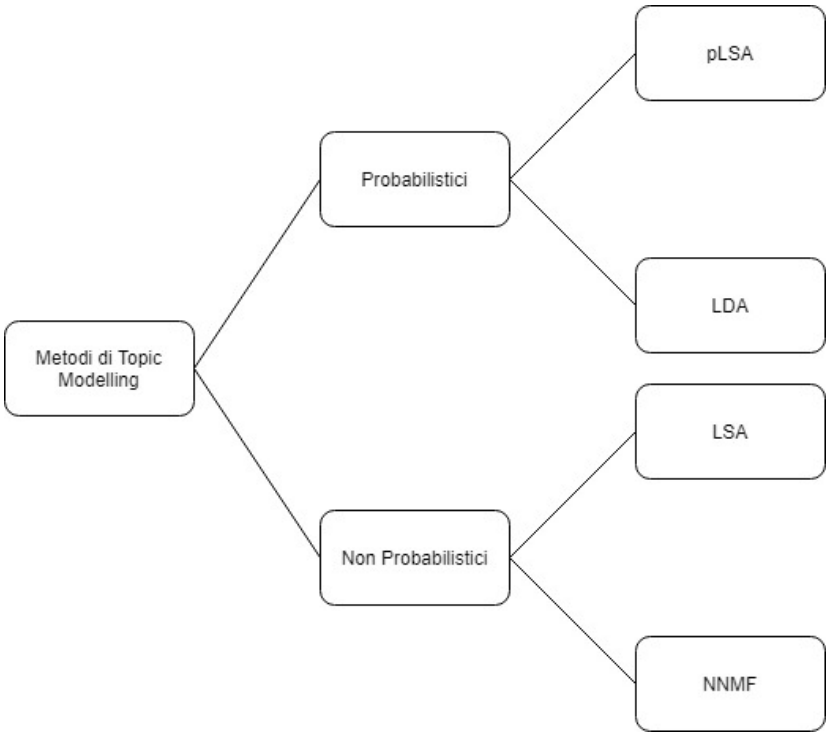

Figura 2.1: Classificazione dei metodi classici di topic modelling estratta da *Topic Modeling: A Comprehensive Review* [27]

2.2.1 TF-IDF

La maggior parte delle tecniche utilizzate per il topic modelling, ed anche per molti altri campi dell'NLP, derivano dal settore dell'Information Retrieval. Essa è una disciplina che si occupa delle tecniche atte a soddisfare i bisogni informativi delle persone, attraverso metodi di memorizzazione, rappresentazione e organizzazione dei documenti contenenti le informazioni.

Già negli anni '80 il campo dell'information retrieval ha prodotto uno schema di rappresentazione del testo chiamato *term frequency-inverse document frequency* (TF-IDF) [24] per applicare una statistica numerica a una parola che sarebbe rappresentativa della sua importanza all'interno di un documento. Tale funzione aumenta proporzionalmen-

te al numero di volte che il termine è contenuto nel documento, ma cresce in maniera inversamente proporzionale con la frequenza del termine nella collezione. Essa può essere scomposta in due fattori:

1. Il primo è il numero di termini presenti nel documento. Questo numero può essere diviso per la lunghezza del documento stesso per evitare che siano privilegiati i documenti più lunghi.
$$tf_{ij} = \frac{n_{ij}}{|d_j|}$$
dove n_{ij} è il numero di occorrenze del termine i nel documento j, mentre il denominatore $|d_j|$ è il numero di termini presenti nel documento j

2. Il secondo fattore della funzione indica l'importanza generale del termine i nella collezione:
$$idf_i = \log_{10} \frac{|D|}{|\{d : i \in d\}|}$$
dove $|D|$ è il numero di documenti nella collezione, mentre il denominatore è il numero di documenti che contengono il termine i.

L'equazione tf-idf per il termine i contenuto nel documento j può essere quindi scritta come:

$$(tf\text{-}idf)_{i,j} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$$

La funzione TF-IDF riduce dunque documenti di lunghezza arbitraria a vettori numerici a dimensione fissa ma, tuttavia, non fornisce informazioni sulla struttura statistica inter o intra-documento.

2.2.2 Latent Semantic Analysis

Per risolvere questa mancanza sono state quindi introdotte altre tecniche ed in particolare la *Latent Semantic analysis* (LSA), presentata da Deerwester et al. nel 1990 [19]. Essa utilizza la decomposizione ai valori singolari (single value decomposition o

SVD) per ottenere la riduzione della dimensionalità della matrice termine-documento ed identificare un sottospazio lineare che cattura la maggior parte della varianza in una raccolta di documenti, consentendo a LSA di essere un processo generativo piuttosto che discriminatorio. L'algoritmo inizia generando una matrice delle occorrenze delle parole nei documenti. Nella forma più semplice essa può essere creata identificando per ogni parola il numero di volte che essa appare in ogni documento. Tuttavia questo metodo tiene poco conto del significato dei termini. Infatti parole più specifiche o che appaiono solamente in pochi documenti sono più cariche di significato. Per questo motivo si predilige l'utilizzo di una matrice generata tramite TF-IDF, così parole che sono spesso presenti in alcuni documenti ma poche volte nell'intero corpus avranno un peso alto. La matrice delle occorrenze A presenta però alcune limitazioni:

1. E' una matrice di dimensioni molto elevate, che potrebbe dunque essere costosa da utilizzare dal punto di vista computazionale

2. E' una matrice con molto rumore visto che può essere ricca di termini che appaiono poche volte

3. E' una matrice presumibilmente sparsa, ovvero con molti elementi uguali a zero

Per questi motivi è necessario ridurre la dimensione della matrice A per essere in grado di ottenere dei topic che catturino bene le relazioni tra le parole e i documenti. La riduzione viene eseguita utilizzando la decomposizione ai valori singolari (SVD), una tecnica di algebra lineare che permette di scomporre A nelle seguenti tre matrici:

$$A \approx USV^T$$

dove U è la matrice ortogonale documento-topic, S è la matrice diagonale contenente i valori singolari di A e V è la trasposizione dell'ortogonale che rappresenta la matrice termine-topic. Questo approccio permette di raggiungere compressioni decisamente migliori rispetto a TF-IDF ed inoltre permette di individuare alcuni aspetti linguistici di base, come la sinonimia e la polisemia.

2.2.3 Non-Negative Matrix Factorization

Non-negative matrix factorization (NNMF) è un metodo di fattorizzazione di una matrice che può essere applicato a qualsiasi tipo di dato in forma matriciale. Consideriamo di avere un corpus formato da M documenti contenenti N possibili parole. NNMF assume che la matrice termine-documento A, costruita partendo dal corpus per esempio utilizzando TF-IDF, può essere approssimata dal prodotto di due matrici W e H.

$$A \approx WH$$

Dove W è una matrice di dimensione $M \times T$ e H è una matrice di dimensione $T \times N$. Tutti i valori di W e H sono non negativi mentre la dimensione T equivale al numero di topic desiderati. Il valore W_{ij} può essere interpretato come un peso che indica la presenta del topic j nel documento i, mentre il valore H_{jk} può essere interpretato come un peso che indica la presenza della parola k nel topic j.

Questa tecnica di fattorizzazione può essere trasformata in un problema di ottimizzazione che permette di trovare le due matrici non negative W e H minimizzando la seguente funzione:

$$\parallel A - WH \parallel$$

dove $\parallel X \parallel$ è la norma euclidea di X. Lee e Seung [21] hanno proposto nel 1999 un algoritmo iterativo per risolvere questo problema di ottimizzazione, che è tuttora usato frequentemente. Partendo da due matrici non negativi casuali W e H, l'algoritmo le aggiorna iterativamente utilizzando le seguenti formule:

$$W = W\frac{A \cdot H^T}{W \cdot H \cdot H^T} \qquad\qquad H = H\frac{W^T \cdot A}{W^T \cdot W \cdot A}$$

Sebbene questo algoritmo sia semplice da implementare, esso è abbastanza lento a convergere. Questo problema lo si ha sopratutto quando bisogna trovare i valori in matrici che sono molto grosse. Pertanto è necessario utilizzare un metodo più efficace se si ha il problema di scoprire i topic presenti in un dataset di documenti molto ampio.

2.2.4 Probabilistic Latent Semantic Analysis

Un ulteriore miglioramento lo si è avuto con l'introduzione del metodo *probabilistic Latent Semantic Analysis* (pLSA), ideato da Hofmann nel 1999 [25]. Tramite questo approccio ogni parola in un documento viene campionata da un modello di miscela, in cui i componenti della miscela sono variabili casuali multinomiali che possono essere viste come rappresentative dei topic. Ogni parola viene generata da un singolo topic e diverse parole all'interno di un documento possono essere generate da diversi topic. Si vuole quindi trovare un modello probabilistico tale per cui per ogni documento d_i e per ogni parola w_j la probabilità $P(d_i, w_j)$ corrisponda all'elemento nella matrice documento-termine all'indice (i, j). Inoltre, dato un documento d, la probabilità che il topic z sia presente in esso è definita come P(z|d) mentre una parola w è scelta dal topic z con probabilità $P(w|z)$. La probabilità congiunta di avere dunque una parola in un documento può essere definita come:

$$P(D, W) = P(D) \sum_{Z} P(Z|D)P(W|Z)$$

P(D), P(Z|D) e P(W|Z) sono quindi i parametri del modello. P(D) può essere determinata direttamente dal corpus. P(Z|D) e P(W|Z) sono invece generate da una distribuzione multinomiale e vengono addestrate da un algoritmo di massimizzazione delle aspettative (algoritmo EM) per inferenza di stime di parametri che dipendono da variabili latenti non osservate. È importante sottolineare che pLSI non fornisce un modello probabilistico al livello dei documenti, in quanto P(D) non è modellata da nessun parametro, e questo comporta due problemi:

- Non è possibile assegnare una probabilità ad un nuovo documento non presente nel training set
- Il numero di parametri di pLSA aumenta linearmente con il numero di documenti che si hanno e questo può portare a problemi di overfitting

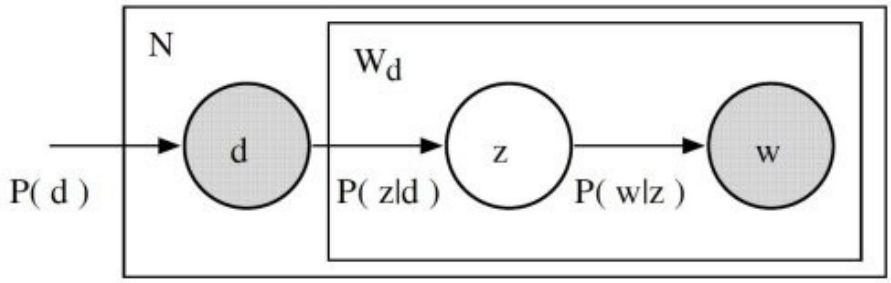

Figura 2.2: Rappresentazione grafica di pLSA estratta dall'articolo *Probabilistic latent semantic indexing* [25]. d e w rappresentano gli elementi di cui siamo a conoscenza, ovvero i documenti e le parole, mentre z rappresenta la variabile latente dei topic

2.2.5 Latent Dirichlet Allocation

Latent Dirichlet Allocation, abbreviato come LDA, è un modello generativo probabilistico ideato da J. K. Pritchard, M. Stephens e P. Donnelly nel 2000. Ha radici nella biologia evolutiva, essendo stato sviluppato per lo studio della genetica delle popolazioni. E' stato poi successivamente introdotto nel campo del machine learning nell'articolo *Latent Dirichlet Allocation* di David Blei, Andrew Ng e Michael I. Jordan pubblicato nel 2003 [16].

L'idea alla base di LDA è che ogni documento può essere descritto da una distribuzione di topic e che ogni topic sia caratterizzato da una distribuzione di parole. Per capire questo concetto consideriamo, per esempio, di avere un corpus di mille documenti e le mille parole più presenti in essi. Assumiamo quindi che ogni documento contenga in media cinquecento di queste parole. Un metodo per capire a quale categoria appartenga ogni documento è quello di connettere ciascuno di essi ad ogni parola che contiene, come mostrato nella figura 2.3. In questo modo i documenti che sono connessi agli stessi set di parole tratteranno gli stessi argomenti. Questo algoritmo è tuttavia molto oneroso dal punto di vista computazionale, specialmente quando si ha un corpus molto esteso.

Si può risolvere questo problema introducendo un ulteriore livello latente. Consi-

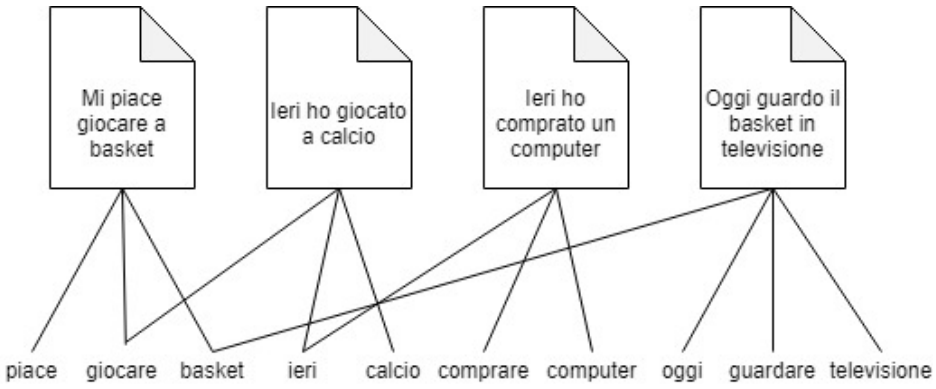

Figura 2.3

deriamo di sapere che ricorrono dieci topic allinterno di tutti i documenti, dei quali però non siamo a conoscenza. Possiamo quindi utilizzare le informazioni che abbiamo a priori, ovvero i testi e le parole presenti in essi, per individuare questi topic. Lo si riesce a fare collegando le singole parole ai topic che meglio le rappresentano e collegando ciascun topic ai documenti che trattano di essi. Si ottiene quindi che ogni documento è rappresentato da un insieme di topic e che ogni topic è definito da un insieme di parole, come mostrato nella figura 2.4.

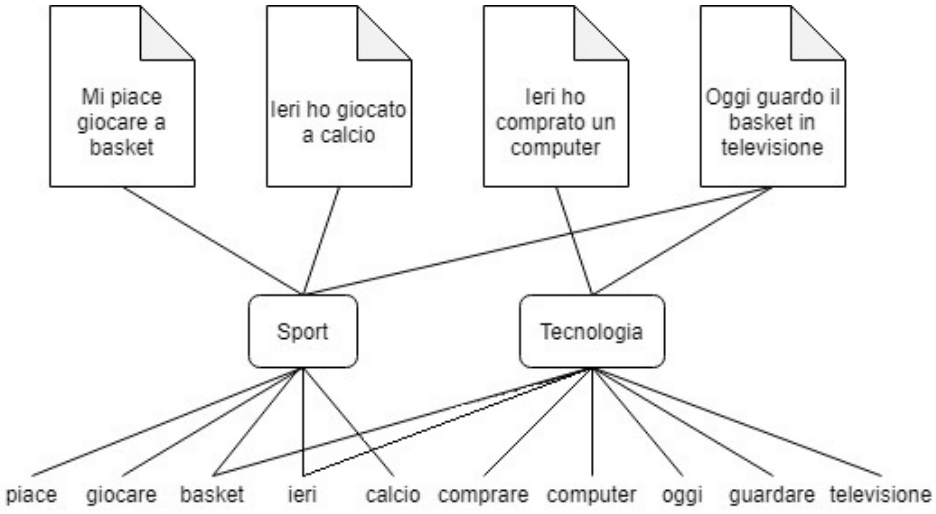

Figura 2.4

L'obiettivo dell'LDA è proprio quello di trovare questi collegamenti e quindi di capire le distribuzioni dei topic nei documenti e delle parole nei topic.

Come spiegato nell'articolo *Latent Dirichlet Allocation* [16], l'innovazione di LDA

consiste nell'usare le distribuzioni di Dirichlet per rappresentare quanto ogni argomento è presente in ogni documento e quanto ogni termine è presente in ogni topic, consentendo così l'inferenza bayesiana su un modello gerarchico a tre livelli. LDA assume che ogni documento d venga generato nel seguente modo:

1. Si sceglie il numero di parole N che il documento deve avere in base alla distribuzione di Poisson:

$$N \sim Poisson(\xi)$$

2. Si sceglie una mistura di topic dall'insieme θ, in base alla distribuzione di Dirichlet su un numero fissato k di topic:

$$\theta \sim Dir(\alpha)$$

3. Per ognuna delle N parole w_n del documento:

   (a) Si sceglie un topic da una distribuzione multinomiale:

$$z_n \sim Multinomiale(\theta)$$

   (b) Si sceglie una parola w_n dalla distribuzione multinomiale del topic:

$$P(w_n|z_n, \beta)$$

E' importante sottolineare che il numero di topic k è fisso e conosciuto a priori. Le probabilità delle parole di appartenere ad uno dei k topic sono quindi stimate tramite l'algoritmo e sono contenute nella matrice β di dimensione $k \times V$ dove $\beta_{i,j} = P(w_j = 1|z_i = 1)$

Il modello dell'LDA è rappresentato tramite diagramma nella figura 2.5:

- K è il numero di topic
- N è il numero di parole nel documento

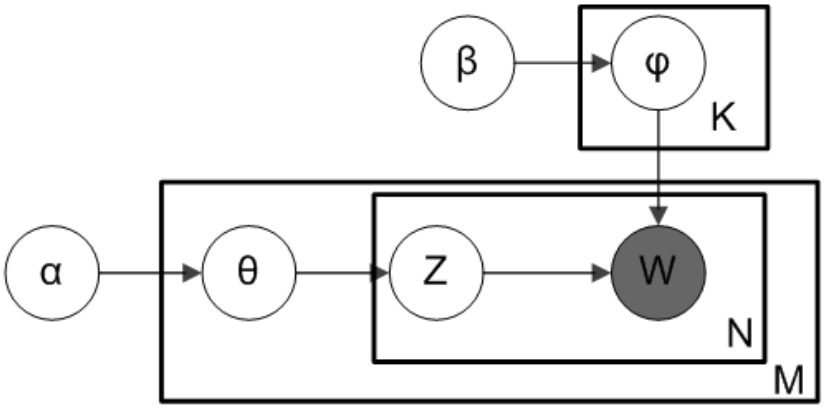

Figura 2.5: Rappresentazione grafica di LDA estratta dall'articolo *Latent Dirichlet Allocation* [16]. I rettangoli esterni rappresentano i documenti ed i topic mentre quello interno le parole. Il cerchio scuro rappresenta gli elementi conosciuti, ovvero le parole, mentre i cerchi bianchi rappresentano le variabili latenti

- M è il numero di documenti
- α è il parametro che definisce la distribuzione di Dirichlet documento-topic
- β è il parametro che definisce la distribuzione di Dirichlet topic-termine
- φ_k è la distribuzione delle parole del topic k
- θ_m è la distribuzione dei topic per il documento m
- z_{mn} è il topic assegnato alla parola n nel documento m
- w_{mn} è la parola osservata nel documento m in posizione n

Come viene mostrato, LDA contiene tre livelli di rappresentazione, a differenza di pLSA. I parametri α e β sono al livello del corpus e vengono campionati una volta nel processo di generazione di esso. Il parametro θ_m è una variabile a livello dei documenti che viene campionata una volta per ognuno di essi. Le variabili z_{mn} e w_{mn} sono a livello delle parole e sono campionati una volta per ogni termine in ogni documento. Infine, φ_k rappresenta la distribuzione di probabilità di parole per un argomento k.

Analizziamo meglio nel dettaglio il funzionamento dell'algoritmo. Come è stato spiegato, LDA si basa sul concetto che ogni parola di ogni documento deriva da un topic e che quel topic è scelto da una distribuzione del documento su tutti i topic. Dobbiamo quindi considerare due matrici:

- α è una matrice dove ogni riga rappresenta un documento ed ogni colonna un topic. L'elemento in posizione (i, j) indica quindi quanto il topic j è presente nel documento i. Pertanto $\theta_{mk} = P(k \mid m)$ è la probabilità di avere il topic k nel documento m

- β è una matrice dove ogni riga rappresenta un topic e d ogni coloona una parola. L'elemento in posizione (i, j) indica quindi quanto la parola j è presente nel topic i. Pertanto $\varphi_{kn} = P(n \mid k)$ è la probabilità di avere la parola n nel topic k

Si può quindi definire che la probabilità di avere una parola dato un documento equivale a:

$$P(n \mid d) = \sum_{k \in K} P(n \mid k) P(k \mid d)$$

dove K è il numero totale di topic. Essa equivale dunque al prodotto scalare tra α e β per ogni topic k. Possiamo quindi considerare LDA simile a quanto visto con la decomposizione ai valori singolari in LSA. La matrice delle parole nei documenti è infatti decomposta in due matrici contenenti le distribuzioni dei topic nei documenti e delle parole nei topic.

Osserviamo ora come LDA impara i parametri di queste due matrici. Il primo passo è quello di assegnare casualmente ogni parola di ogni documento ad uno dei K topic. Successivamente si cerca di massimizzare una funzione di verosimiglianza tra i documenti e le due matrici. Per far ciò si utilizza il campionamento di Gibbs, un algoritmo Monte Carlo basato su Catena di Markov utilizzato per ottenere una sequenza di campioni casuali da una distribuzione di probabilità multivariata [23]. Esso viene utilizzato per migliorare ad ogni passo le matrici α e β nel seguente modo:

1. Per ogni documento d e per ogni topic k

   (a) Si calcola $P_k = P(k \mid d)$, cioè la proporzione di parole nel documento d assegnate al topic k

   (b) Si calcola $P_n = P(n \mid k)$, cioè la proporzione di assegnamenti al topic k su tutti i documenti che arrivano dalla parola n

   (c) Alla parola n viene assegnato un nuovo topic che abbia probabilità pari a $P_k * P_n$, assumendo quindi che tutti gli assegnamenti dei topic siano corretti eccetto quello per la parola in questione. Si aggiornano quindi le matrici α e β in base al nuovo assegnamento

2. Si ripete il passo precedente fino ad ottenere dei buoni risultati

LDA funziona generalmente molto meglio di pLSA. Inoltre, come si è visto precedentemente, pLSA non è in grado di gestire documenti nuovi non presenti nel dataset di allenamento. LDA, invece, utilizza il dataset di partenza per modellare le distribuzioni e quindi si può utilizzare per osservare nuovi documenti campionando da esse. Proprio per la sua natura statistica, LDA necessita di un set di dati ragionevolmente ampio per poter fornire dei buoni risultati. Durante il lavoro di tesi si è osservato che, allenando LDA tramite il dataset di comunicati stampa descritto successivamente, si sono ottenuti dei buoni risultati in termini di coerenza tra i topic identificati solamente utilizzando un campione di grandezza superiore ai 1000 documenti.

2.3 Metodi di Embedding

Grazie allo sviluppo che il deep learning ha avuto negli ultimi anni, si è assistito ad un utilizzo sempre maggiore delle reti neurali anche in molti campi del NLP ed alla nascita del termine neural language model, per indicare appunto un modello che utilizza una rappresentazione vettoriale delle parole per effettuare delle predizioni, ottenuta tramite una rete neurale.

Il primo modello linguistico neurale viene proposto da Bengio et al. nel 2001 [14] tramite una semplice rete neurale feed-forward seguita da un livello di pooling che permette di ottenere dei vettori rappresentanti le parole prese in input. Oggi questi vettori sono conosciuti con il nome di word embedding. Recentemente le reti neurali feed-forward sono state sostituite, per la modellazione linguistica, dalle reti neurali ricorrenti (RNNs) e dalle Long Short-Term Memory (LSTM). Negli ultimi anni sono stati però proposti molti nuovi modelli linguistici che estendono il modello classico di LSTM.

Verranno analizzati adesso quelle che sono state le innovazioni tecnologiche più importanti dell'ultimo decennio nel campo del NLP. Si porrà particolare attenzione ai modelli Transformer e a BERT, in quanto verranno utilizzati successivamente per la risoluzione del problema su cui si è basato il lavoro di tesi.

2.3.1 Word2vec

Word2vec è stato ideato e presentato tramite due articoli nel 2013 da un team di ricercatori di Google guidato da Tomas Mikolov [33][34] . Esso utilizza una rete neurale per imparare le associazioni tra le parole in un corpus di documenti. Una volta allenato, tale modello riesce ad identificare se delle parole sono simili tra di loro oppure può suggerire un termine mancante in una frase. Come indica il nome, word2vec impara a rappresentare ogni singola parola tramite un vettore. Questa rappresentazione può essere usata, per esempio, per misurare la somiglianza semantica tra due parole calcolando la cosine similarity tra i loro vettori. Word2vec permette quindi di ottenere una rappresentazione delle parole che può essere sfruttata in molti campi del NLP, tra cui anche il topic modelling.

Analizziamo nel dettaglio il funzionamento e l'architettura di word2vec. Essa è una semplice rete neurale a due livelli che viene allenata per ricostruire il contesto linguistico delle parole. Prende in input un insieme molto grande di documenti testuali

e produce uno spazio vettoriale dove ogni parola presente nel corpus è rappresentata tramite un vettore della dimensione desiderata E. Tali vettori sono posizionati nello spazio in modo tale che parole che hanno lo stesso contesto nel corpus siano vicine una all'altra. Esistono due modelli differenti di word2vec: Continuous Bag-of-Words (CBOW) e Skip-Gram.

Il modello CBOW è allenato a predire una parola partendo da un numero fisso di parole che la precedono e che la susseguono all'interno del documento. Considera quindi l'intero contesto come una unica osservazione e per questo funziona generalmente meglio sui dataset più piccoli come affermato da Mikolov.

Il modello Skip-gram è allenato, invece, partendo da una singola parola a predire le parole che la precedono e la susseguono in un documento. Considera quindi ogni coppia formata dalla parola di partenza e da una parola appartenente al contesto come una nuova osservazione e per questo tende a fornire migliori risultati quando si hanno dei dataset più grandi.

L'architettura di word2vec è simile a quella di un autoencoder. E' costituita, infatti, da due livelli nei quali prima i vettori delle parole vengono compressi in una dimensione minore e poi, invece che decomprimerli come avviene negli autoencoder, si calcolano le probabilità di avere ogni singola parola del dizionario all'interno del contesto della frase. Come si mostra nella figura 2.7, Word2vec è quindi formata da tre elementi fondamentali:

- Un livello di input nel quale ogni parola viene rappresentata tramite codifica one-hot. Ogni parola viene indicata con un vettore di lunghezza pari alla dimensione del dizionario e con ogni elemento uguale a 0 tranne quello all'indice che rappresenta la parola, al quale è assegnato il valore 1.

- Un livello nascosto che contiene una matrice di V righe ed E colonne rappresentante gli embedding delle parole. La riga della matrice all'indice i contiene infatti il vettore della i-esima parola del dizionario

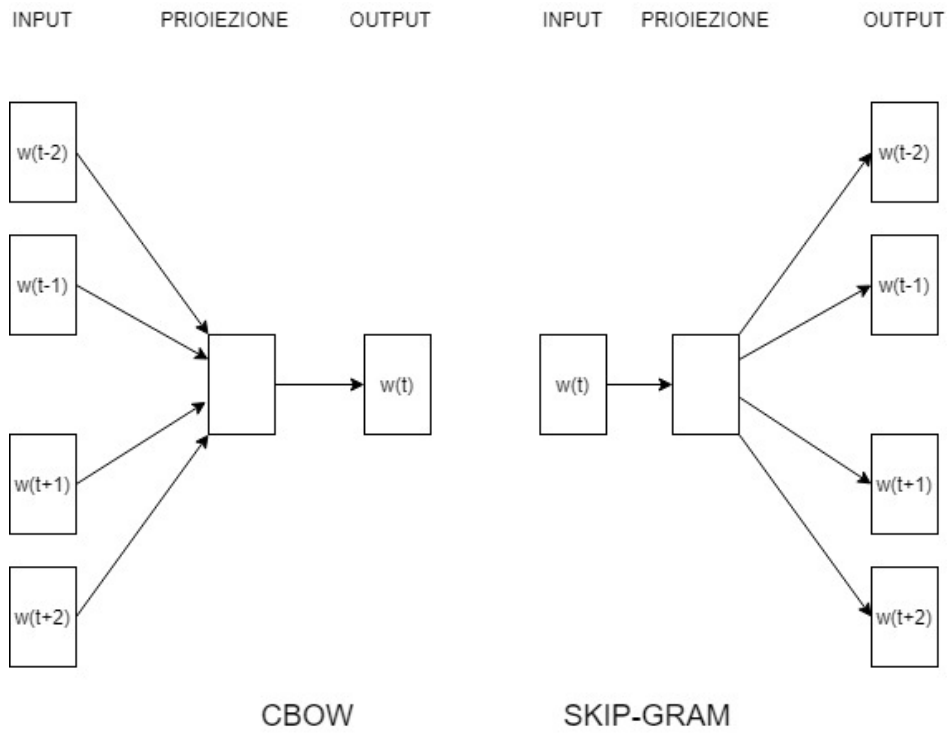

Figura 2.6: Rappresentazione dei modelli CBOW e Skip-gram di word2vec estratta dall'articolo *Efficient Estimation of Word Representations in Vector Space* [33]

- Un livello di output che contiene una matrice di E righe e V colonne nel quale la i-esima colonna rappresenta l'embedding della i-esima parola del contesto. E' presente inoltre una funzione di attivazione che viene applicata ad ogni riga della matrice e ritorna la distribuzione di probabilità di avere le parole contenute nel dizionario all'interno della frase. Si utilizza la seguente funzione di softmax:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

  dove z_j rappresenta la j-esima riga all'interno della matrice

Word2vec è allenata usando il metodo della massima verosimiglianza e cercando quindi di predire la parola che abbia la massima probabilità di essere inserita all'interno della frase. Tuttavia questo processo è molto oneroso dal punto di vista computazionale dato che è necessario calcolare la probabilità per tutte le parole presenti all'interno del dizionario ad ogni training step. Si utilizzano quindi delle tecniche per ridurre il numero di campioni di parole e per aumentare l'accuratezza del modello:

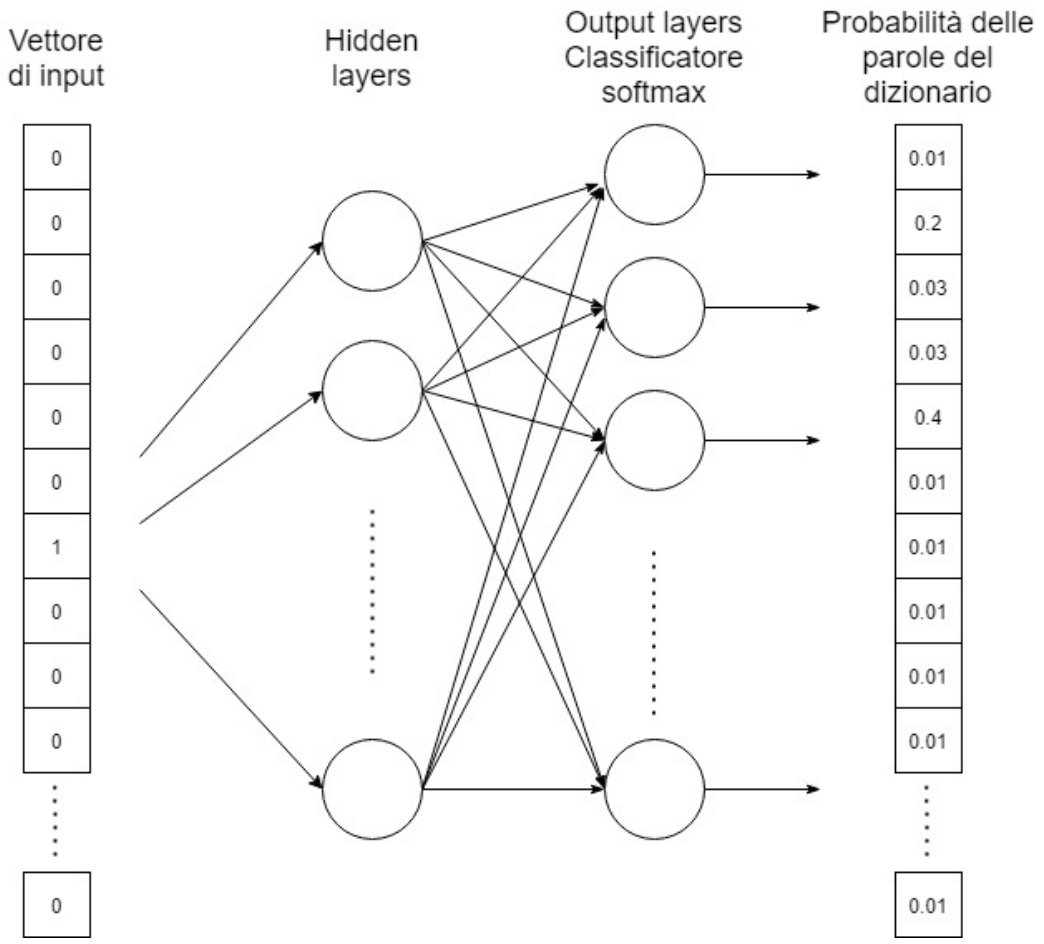

Figura 2.7: Architettura della rete neurale che compone word2vec

- Si utilizza una tecnica di negative sampling per selezionare solamente un numero N di parole di cui verranno calcolate le probabilità. Queste parole sono campionate all'interno del dizionario usando come probabilità la loro frequenza all'intero del corpus. Parole più frequenti avranno quindi probabilità maggiore di essere campionate

- Gruppi di parole che appaiono frequentemente vicine vengono considerate come una unica parola

- La qualità dell'embedding cresce con la dimensione E che viene utilizzata per i vettori. Solitamente si sceglie un valore compreso tra 100 e 1000

2.3.2 Doc2vec

Partendo dallo studio effettuato tramite word2vec, Mikolov ha proposto un ulteriore approccio innovativo che permette di ottenere vettori di embedding che rappresentano non più le singole parole, ma gli interi documenti. Questo metodo è chiamato doc2vec ed è stato presentato nel 2014 nell'articolo *Distributed Representations of Sentences and Documents* [30] .

L'architettura di doc2vec è molto simile a quella di word2vec. Viene però introdotto un ulteriore parametro, chiamato Paragraph ID, che è unico per ogni documento e che fornisce una rappresentazione vettoriale del suo contesto all'interno di uno spazio. Come per word2vec, esistono due architetture di doc2vec mostrate nella figura 2.8:

- L'architettura Distributed Memory Version of Paragraph Vector (PV-DM) riprende il modello CBOW di word2vec, con l'aggiunta però del Paragraph ID. I vettori delle parole rappresentano i termini presenti in tutto il corpus, mentre i Paragraph ID rappresentano ogni singolo documento e sono calcolati anche essi durante la fase di allenamento della rete. PV-DM impara a predire le parole mancanti in una frase capendo il contesto della frase.

- L'architettura Distributed Bag of Words Version of Paragraph Vector (PV-DBOW) riprende il modello Skip-gram di word2vec. Essa però utilizza il Paragraph ID per imparare a classificare tutte le parole della frase. Durante l'allenamento una lista di parole è campionata dal dizionario e successivamente il classificatore impara a capire se quelle parole appartengono alla frase rappresentata dal Paragraph ID.

Doc2vec ha permesso quindi di ottenere una rappresentazione dei documenti interamente basata sulla semantica delle loro parole. Come si vedrà con il metodo top2vec, questo ha portato ad un grosso passo avanti anche nel campo del topic modelling, permettendo di ottenere nuove tecniche non più basate sulla statistica e la probabilità ma sul concetto di embedding e clustering dei vettori.

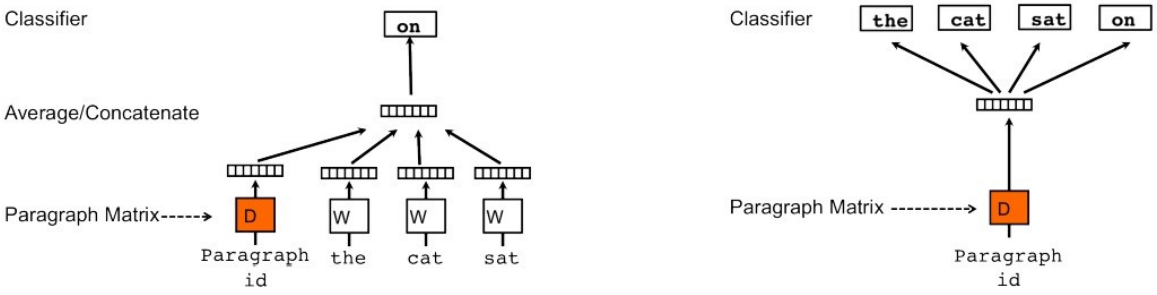

Figura 2.8: Architetture PV-DM e PV-DBOW di doc2vec estratte dall'articolo D*istributed Representations of Sentences and Documents* [30]

2.3.3 Transformer

Nel 2017 viene presentato tramite l'articolo di Vaswani et al. *Attention is all you need* una nuova e rivoluzionaria architettura di rete neurale chiamata Transformer, la quale si basa sul concetto di attenzione [39]. I Transformer utilizzano un modello sequence-to-sequence (seq2seq) cioè una rete neurale che prende in input una sequenza di elementi, come delle parole di una frase, e produce in output un'altra sequenza. I modelli seq2seq permettono di ottenere ottimi risultati per esempio nella traduzione, dove una sequenza di parole in una lingua è trasformata in una sequenza di parole in un'altra lingua, ma sono comunque utilizzati anche in altri campi dell'intelligenza artificiale dove è necessario utilizzare dati sequenziali.

I modelli seq2seq sono costituiti da un encoder e da un decoder. L'encoder prende in input la sequenza e la trasforma in un vettore in uno spazio n-dimensionale. Questo vettore è successivamente trasformato nuovamente tramite il decoder nella sequenza di output, che può essere una copia dell'input oppure per esempio la sua traduzione in un'altra lingua. Tramite il processo di allenamento l'encoder impara quindi ad ottenere una rappresentazione vettoriale coerente della sequenza in input, mentre il decoder impara a produrre correttamente la sequenza in output tramite il task desiderato.

L'elemento però che rende i Transformer così efficienti ed innovativi è quello del concetto di attenzione. L'attenzione è un meccanismo che permette di capire in una

sequenza di elementi quali sono quelli più importanti e su cui bisogna focalizzarsi maggiormente. L'encoder prende quindi in input una sequenza di parole e, utilizzando il meccanismo di attenzione, attribuisce loro un peso maggiore per i termini che sono più importanti dal punto di vista semantico all'interno della frase. Questi pesi saranno poi sfruttati dal decoder per migliorare il suo task. Il concetto di attenzione è spiegato nel dettaglio nell'articolo *Effective Approaches to Attention-based Neural Machine Translation* [31].

Analizziamo adesso l'architettura di un Transformer mostrata nella figura 2.9.

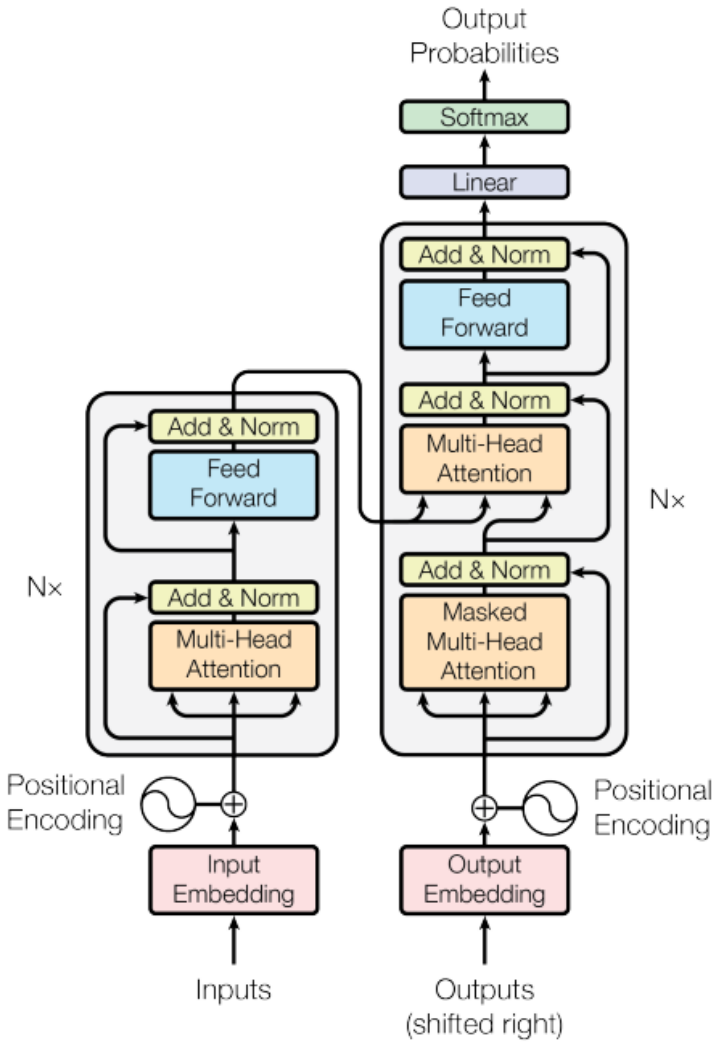

Figura 2.9: Architettura del modello Transformer estratta dall'articolo *Attention is all you need [39]*

Come si nota dall'immagine, l'encoder è posizionato sulla sinitra mentre il decoder sulla destra. Entrambi possono essere formati da un numero Nx di moduli posizionati uno sopra l'altro e che operano in parallelo. Sia gli input che gli output sono trasformati in un vettore n-dimensionale tramite embedding. Il modulo Multi-Head Attention è utilizzato per implementare il meccanismo di attenzione illustrato precedentemente. I pesi attribuiti alle parole sono calcolati tramite la seguente equazione:

$$Attention(Q, K) = softmax(\frac{QK^T}{\sqrt{d_k}})$$

dove Q è il vettore di una singola parola della sequenza, K è una matrice contenente i vettori di tutte le parole della sequenza e d_k è la dimensione di Q. La funzione di softmax è applicata per ottenere valori compresi tra 0 e 1, i quali vengono poi moltiplicati con i vettori delle corrispondenti parole per attribuire loro più o meno importanza. I valori dell'encoder sono passati ad una rete feed-forward e successivamente al decoder, il quale li sfrutta tramite il proprio meccanismo di attenzione per imaparare ad eseguire il task desiderato.

2.3.4 BERT

Uno dei modelli di Transformer più utilizzati nel settore del NLP è sicuramente BERT (Bidirectional Encoder Representations from Transformers). Esso è stato presentato nel 2018 da parte dei ricercatori di Google AI Language [20] e rappresenta lo stato dell'arte in un ampio numero di task del NLP. L'innovazione di BERT è stata quella di applicare il concetto di attenzione e di allenamento bidirezionale dei Tranformer ai modelli linguistici. Nell'articolo di presentazione di BERT viene infatti mostrato come esso permetta di ottenere modelli che interpretano molto meglio il contesto dei documenti rispetto ai modelli allenati per esempio tramite LSTM. Vengono inoltre introdotte due nuove tecniche di allenamento bidirezionale dei modelli, chiamate Masked LM (MLM) e Next Sentence Prediction (NSP).

Al contrario dei modelli unidirezionali, che leggono in modo sequenziale da sinistra a destra i testi, i Transformer analizzano solamente una parola alla volta e sono quindi definiti modelli bidirezionali. Questo permette di imparare il contesto delle parole analizzando tutte quelle vicine e non solamente quelle osservate precedentemente. BERT riceve come input una sequenza di token, i quali vengono poi trasformati in vettori tramite embedding e processati tramite la rete neurale. L'output è quindi sempre una sequenza di vettori, ognuno dei quali corrisponde ad uno dei token presi in ingresso. BERT ha dunque lo scopo di creare un modello linguistico e per questo, rispetto ai normali Tranformer, necessita solamente della parte di encoding della rete.

Quando si genera un modello linguistico è necessario definire un task di predizione su cui esso viene allenato. Molti modelli unidirezionali si basano per esempio sul predire la parola successiva ad una sequenza di termini. BERT, essendo invece un modello bidirezionale, utilizza due strategie:

1. Tramite la tecnica Masked LM, il 15% dei token utilizzati da BERT vengono mascherati sostituendoli con il token [MASK]. Il modello è allenato quindi a predire i valori originali delle parole mascherate basandosi sul contesto fornito dagli altri termini della sequenza. I token che costituiscono la frase, compresi le maschere, sono passate alla rete di encoding del Transformer, la quale genera i corrispondenti vettori. Essi vengono poi moltiplicati per la matrice di embedding delle parole del dizionario. Tramite una funzione di softmax vengono poi calcolate le probabilità di ogni parola del dizionario di essere inserite nel contesto.

2. Tramite la tecnica Next Sentence Prediction (NSP) il modello viene allenato ricevendo coppie di frasi ed imparando a predire se esse sono presenti una successiva all'altra all'interno del corpus. Durante l'allenamento il 50% degli input sono coppie in cui la seconda frase segue la prima in un documento, mentre il restante 50% è formato da due frasi prese casualmente all'interno del corpus. Per allenare BERT tramite questo task si inserisce un token [CLS] token all'inizio di ogni frase

ed un token [SEP] al termine. I token sono quindi processati tramite la rete di encoding e successivamente viene calcolata la probabilità che le due frasi prese in input siano consecutive tramite una funzione di softmax.

Una volta che il modello è stato allenato correttamente, esso può essere utilizzato anche su nuovi documenti. Come è stato però specificato BERT permette di ottenere gli embedding dei token delle parole che costituiscono le frasi, e non dei singoli documenti. Questi ultimi possono essere ottenuti facilmente applicando un livello ulteriore di pooling successivo alla rete di encoding di BERT, il quale ritorna l'embedding dei singoli documenti calcolando per esempio la media o la somma dei vettori dei token. Questo tipo di rappresentazione vettoriale dei documenti che sfrutta BERT è stata proposta nell'articolo *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks* [36] ed è facilmente utilizzabile sfruttando la corrispondente libreria python Sentence-Transformer [28] .

I modelli Transformer, ed in particolare BERT, permettono quindi di ottenere delle rappresentazione dei documenti in uno spazio basandosi interamente sui loro significati semantici. Documenti che trattano degli stessi argomenti saranno quindi probabilmente posizionati vicini all'interno di questo spazio e questo può essere sfruttato anche nel campo del topic modeling. Come verrà analizzato, è possibile raggruppare questi vettori tramite tecniche di clustering ed ottenere quindi una classificazione basata sui loro topic di appartenenza.

Capitolo 3

Definizione del problema e degli strumenti utilizzati

In questo capitolo verrà descritto il problema preso in esame durante il lavoro di tesi. Verranno inoltre presentate le librerie usate per la progettazione della soluzione ed infine i dataset e le metriche di valutazione utilizzate per ottenere i risultati quantitativi.

3.1 Definizione del problema

Ora che è stato spiegato il significato di topic modelling e che ne sono state descritte le principali tecnologie, è possibile introdurre il problema che è stato affrontato in questo lavoro di tesi. L'obiettivo principale dell'attività svolta è stato quello di utilizzare e confrontare tra di loro varie tecniche di topic modelling applicate a dei comunicati stampa.

Il comunicato stampa è uno degli strumenti utilizzati, per esempio da un'azienda o da un ente pubblico, per comunicare con i media e per fornire loro i dati e le informazioni che si desidera divulgare. Tuttavia, affinché questo tipo di relazione funzioni nel modo appropriato, i comunicati stampa devono avere delle particolari caratteristiche:

- Il contenuto di un comunicato stampa deve essere elaborato in funzione del messaggio che si vuole trasmettere tramite la notizia, evitando accuratamente tutte le informazioni inutili o superficiali. Deve contenere tutti i dati fondamentali per la notizia: chi, cosa, quando, dove e perché. Deve essere una sinossi che esprimi chiaramente quelli che sono stati i fatti, senza inserire commenti o giudizi personali

- La struttura di un comunicato stampa deve avere uno stile giornalistico costituito da costruzioni sintattiche semplici, periodi brevi e facilmente interpretabili dai destinatari. Bisogna evitare di scrivere frasi non più lunghe di due o tre righe oppure che contengano ripetizioni.

Queste due caratteristiche sono utili da analizzare anche per poter progettare un modello di NLP che riesca ad estrarre i topic dai comunicati stampa nel modo migliore. Utilizzare infatti tecniche di topic modelling probabilistiche, come LDA o pLSA, risulta poco efficiente. Come si è descritto, i comunicati stampa sono testi generalmente molto corti e per questo l'uso di algoritmi basati esclusivamente sul numero di occorrenze delle parole nei documenti o sulla loro frequenza non permette di ottenere topic ben contraddistinti. Per questo motivo si è deciso di affrontare il problema sfruttando tecnologie più recenti, ed in particolare i Transformer, in modo da riuscire ad ottenere una rappresentazione dei documenti basata sul loro contesto semantico che permetta successivamente di classificarli correttamente in base ai loro topic.

3.2 Strumenti e librerie

Per tutto il codice implementato per realizzare la soluzione è stato utilizzato Python. Esso è uno dei linguaggi di programmazione ad alto livello più utilizzati nell'ambito della data science perché permette di sfruttare numerose librerie messe a disposizione in maniera open source. Vengono di seguito descritte le librerie utilizzate per la soluzione.

3.2.1 Scrapy

Scrapy [9] è un framework di alto livello per il web crawling ed il web scraping che viene usato per navigare tra le pagine in rete ed estrarre dati da esse. Può essere utilizzato per molteplici scopi, dall'estrazione di dati a fini statistici alla creazione di archivi e basi di dati. Uno dei vantaggi di Scrapy è di permettere di inviare molteplici richieste in parallelo che vengono elaborate in modo asincrono. Questo significa che se alcune richieste falliscono, le altre possono essere completate correttamente. Scrapy permette inoltre di salvare le informazioni estratte nella struttura dati e nel formato desiderato, come per esempio in un documento JSON.

Ogni singolo crawler di Scrapy è generato tramite *Spider*, una classe che permette di dichiarare come estrarre le informazioni e come reindirizzarsi attraverso i link contenuti nelle pagine web. Ogni spider è infatti definito da un attributo *start_url* che contiene gli indirizzi da cui si partirà ad estrarre i dati e un metodo *parse()* in cui si specifica quali elementi salvare tramite comandi XPath e CSS. I dati possono essere memorizzati temporaneamente, per esempio in un dizionario, e alla fine dell'esecuzione del crawler li si potrà esportare in un documento di testo nel formato desiderato.

3.2.2 Pandas

Pandas (Python Data Analysis Library) [7] è una libreria Python open source utilizzata per il trattamento e l'analisi di dati. E' uno dei framework più usati nel campo della data science perché mette a disposizione una serie di strutture dati e funzioni utili per manipolare qualsiasi tipo di dato. Pandas è stato principalmente sfruttato nel corso della progettazione per utilizzare la struttura dati *DataFrames*, in particolare per riuscire a manipolare facilmente i comunicati stampa estratti tramite Scrapy.

3.2.3 NumPy

Numpy [6] è una libreria realizzata per Python che aggiunge il supporto per matrici ed array multidimensionali, oltre a fornire una grande collezione di funzioni matematiche per svolgere operazioni su queste strutture di dati. L'elemento principale di NumPy è *ndarray*, che permette di rappresentare strutture dati n-dimensionali con elementi tutti dello stesso tipo.

NumPy è stato utilizzato infatti all'interno del lavoro di tesi per svolgere operazioni complesse tra dati in forma vettoriale. E' stato per esempio sfruttato per calcolare medie, somme o moltiplicazioni di array di dimensioni elevate ottenuti tramite l'embedding dei documenti analizzati.

3.2.4 HuggingFace

HuggingFace [3] è una libreria open source realizzata in Python e compatibile sia con TensorFlow 2.0 che con PyTorch. Essa fornisce una vasta varietà di modelli di Transformer pre-allenati, messi a disposizione gratuitamente e facilmente utilizzabili tramite poche linee di codice. Allenare un modello Transformer può essere un compito molto complesso, oltre a richiedere risorse computazionali elevate, e per questo avere una libreria che permette di condividere modelli già allenati è uno strumento davvero utile.

In particolare il pacchetto Transformers di HuggingFace fornisce più di 30 modelli di Transformer allenati su oltre 100 lingue, come per esempio BERT, GPT, T5 e XLM. Inoltre è presente una vasta comunità nella quale ricercatori e programmatori mettono a disposizione i loro modelli allenati su dataset e task diversificati.

Per poter utilizzare un modello Tranformer su un documento è necessario sempre applicare la seguente pipeline:

1. Si generano i token a partire dal testo preso in input. Un token è un blocco di testo categorizzato, normalmente costituito da caratteri indivisibili chiamati lessemi.

La tokenizzazione è un processo reversibile, in quanto è possibile ricostruire il testo originale a partire dai token

2. Si effettua l'encoding di ogni singolo token per ottenere delle rappresentazioni vettoriali che abbiano significato

3. Si effttua il decoding dei vettori generati dal Transformer per ottenere o per calcolare il risultato del task desiderato

3.2.5 SentenceTransformer

SentenceTransformer [28] è un framework per il linguaggio Python sviluppato da Ubiquitous Knowledge Processing Lab e descritto nell'articolo *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks* [36]. Esso fornisce una serie di modelli preallenati utilizzabili per ottenere rappresentazioni vettoriali per frasi, paragrafi od interi documenti. Questi vettori sono rappresentativi dei significati dei testi e perciò documenti che trattano argomenti simili saranno posizionati vicini nello spazio generato. SentenceTransformer fornisce numerosi modelli, in più di 100 lingue, che rappresentano lo stato dell'arte in vari task. Essi sono generati a partire da una rete Transformer, a cui viene aggiunto un livello di pooling che calcola i vettori dei singoli documenti facendo la media oppure sommando i vettori dei singoli token. Questa semplice rete, formata da un Transformer e da un livello di pooling, può essere allenata tramite un task di Semantic Textual Similarity (STS), che consiste nel calcolare un punteggio di similarità tra due frasi. Per far ciò si utilizza un rete neurale siamese in cui si generano i vettori di due documenti separatamente e poi si utilizza come funzione di loss l'errore quadratico medio tra essi. La rete viene allenata utilizzando un dataset annotato contenente coppie di frasi la cui similarità è conosciuta ed indicata tramite un punteggio, permettendo così di imparare a generare vettori di embedding vicini per documenti con contesti semantici simili.

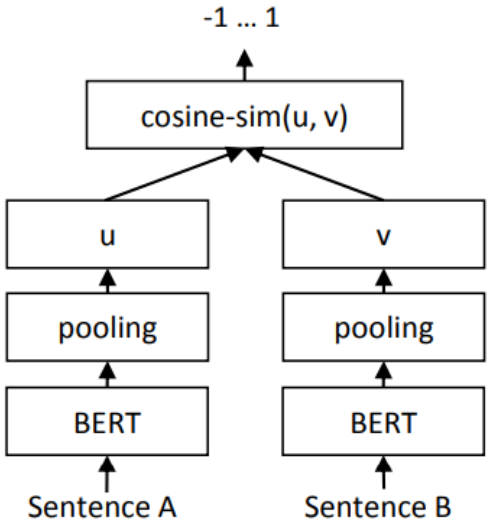

Figura 3.1: Architettura della rete neurale siamese utilizzata per l'allenamento di un modello di SentenceTransformer estratta dall'articolo *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks* [36]

SentenceTransformer è stato utilizzato nell'implementazione del lavoro di tesi per ottenere delle rappresentazioni vettoriali dei comunicati stampa a partire dal modello Transformer desiderato della libreria HuggingFace.

3.3 Dataset

Si è deciso di utilizzare come comunicati stampa di riferimento, su cui sono state testate varie tecniche di topic modelling, quelli redatti dal governo inglese. In particolare le notizie sono state estratte tramite un crawler dal sito `www.gov.uk` nella sezione *News and communications*. Il sito contiene comunicati stampa riguardanti tutto ciò che è inerente alla politica, all'amministrazione e al territorio inglese. Ogni giorno vengono pubblicate decine di notizie che spaziano dalla situazione sanitaria alla politica estera, dall'economia all'ambiente. Questo può permettere quindi di ricavare numerosi topic diversificati tra di loro.

Il crawler utilizzato per estrarre i comunicati stampa dal sito *gov.uk* è stato realizzato

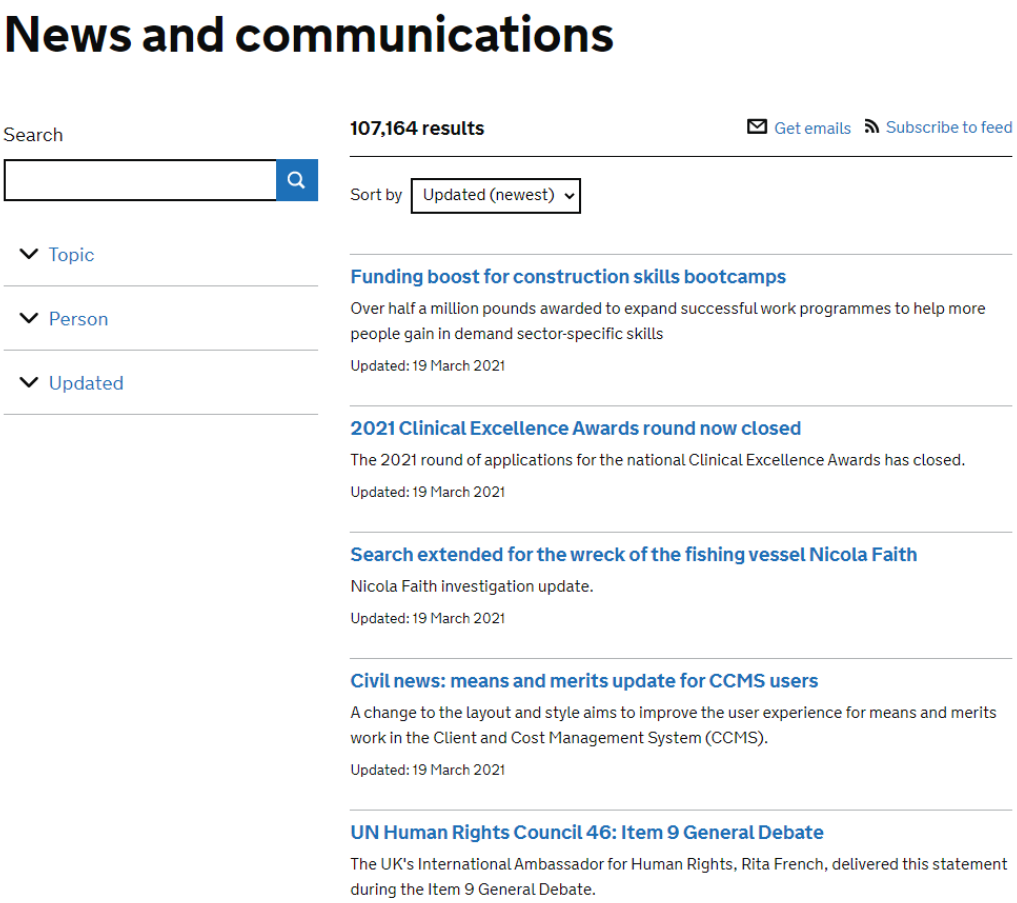

Figura 3.2: Esempio di pagina web del sito `https://www.gov.uk/search/news-and-communications`

utilizzando la libreria Scrapy. In particolare è stato creato uno spider che, partendo dalla pagina web `https://www.gov.uk/search/news-and-communications`, ricerca tutti i titoli dei comunicati stampa presenti in essa. Per ogni titolo viene poi estratto dallo spider, ricercando il tag HTML *href*, l'indirizzo url della pagina contenente il relativo articolo completo. A questo punto il crawler scarica in modo asincrono ognuna di queste pagine ed estrae da esse i seguenti elementi:

- Il titolo del comunicato stampa è estratto dalla pagina tramite CSS selector *'a::text'*
- La data di pubblicazione del comunicato stampa è estratta dalla pagina tramite XPath selector *'//time/text()'*

- Il testo del comunicato stampa è esratto dalla pagina tramite XPath selector *//div[@class='govspeak']*

Inoltre, una volta estratti tutti gli articoli dalla pagina, viene ricercato dallo spider il link alla pagina successiva contenente ulteriori comunicati stampa, tramite CSS selector *'.gem-c-pagination_list'*. Il crawler passerà quindi a questa pagina e ricercherà nuovamente i titoli degli articoli in maniera iterativa. E' stato utilizzata una variabile per tenere conto del numero di comunicati stampa estratti. Una volta che si è raggiunta la quantità desiderata lo spider cesserà l'esecuzione ed esporterà tutti i dati salvati in un file in formato JSON con la seguente struttura:

```
[
{"date": "data_articolo_1", "title": "titolo_articolo_1", "link":
    "link_articolo_1", "text": "testo_articolo_1"},
{"date": "data_articolo_2", "title": "titolo_articolo_2", "link":
    "link_articolo_2", "text": "testo_articolo_2"},
.
.
.
{"date": "data_articolo_n", "title": "titolo_articolo_n", "link":
    "link_articolo_n", "text": "testo_articolo_n"}
]
```

Per utilizzare i comunicati stampa estratti come dataset, il file JSON creato è stato letto tramite la libreria Pandas e caricato in un DataFrame. Si è ottenuto quindi un dataset di 20.000 comunicati stampa, come mostrato nella tabella 3.1.

I comunicati stampa del dataset risalgono al periodo che va dal 5 Ottobre 2018 fino al 4 Marzo 2021. E' interessante notare anche quali sono le caratteristiche di questi documenti:

- La lunghezza media dei documenti è di 529 parole

	date	title	link	text
0	2021-03-04	Project Servator goes live at Heysham – trust ...	https://www.gov.uk/government/news/project-ser ...	Project Servator tactics are used by 23 UK pol ...
1	2021-03-04	Change of Her Majesty's Ambassador to Chad: ...	https://www.gov.uk/government/news/change-of-h ...	Mr Mark Matthews has been appointed Her Majest ...
2	2021-03-04	Jeff Halliwell appointed Chair of the Coal Aut ...	https://www.gov.uk/government/news/jeff-halliw ...	Jeff Halliwell has been appointed as Chair of ...
...	...	...	...	...
20000	2018-10-05	Consultation on Inshore Vessel Monitoring Syst ...	https://www.gov.uk/government/news/introducing ...	The Department of Environment, Food and Rural ...

Tabella 3.1: Dataset dei comunicati stampa estratto dal sito *www.gov.uk*

- Il documenti più lungo contiene 8655 parole
- Il documento più corto contiene una sola parola

Il dataset è quindi formato da testi generalmente corti e con poche parole, come mostrato nella figura 3.3.

Inoltre è utile osservare come le parole più frequenti all'interno dei comunicati stampa siano tutte stop words della lingua inglese, ovvero parole prive di significato se prese singolarmente e che spesso è necessario rimuovere dai testi prima di eseguire dei task di NLP perché generano rumore. Queste parole, come si può notare dalla figura 3.4 sono per esempio articoli, congiunzioni o preposizioni.

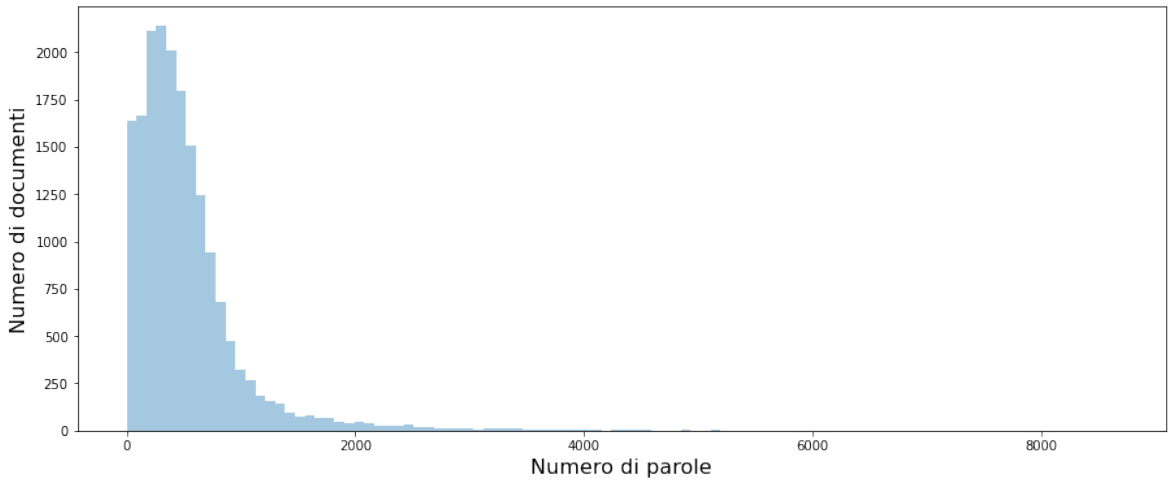

Figura 3.3: Distribuzione del numero di parole dei documenti che costituiscono il dataset dei comunicati stampa

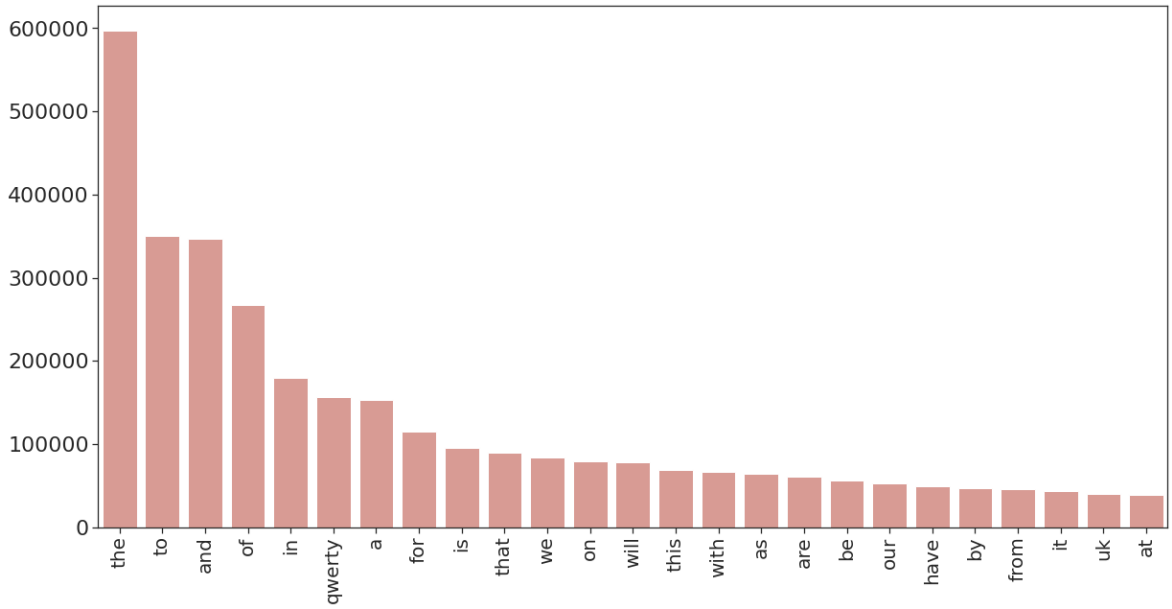

Figura 3.4: Numero di occorrenze delle 25 parole più frequenti nel dataset dei comunicati stampa

Un ulteriore dataset che è stato spesso utilizzato durante il lavoro di tesi è *20Newsgroups*. Esso è composto da circa 18.000 documenti estratti da messaggi in gruppi di discussione online di 20 argomenti diversi. 20Newsgroups è stato creato da Ken Lang per la realizzazione del suo articolo *Newsweeder: Learning to filter netnews paper* [29]

ed è diventato uno dei dataset più utilizzati per verificare algoritmi di machine learning applicati a testi. Si è scelto di utilizzare questo dataset in fase di test proprio perché è facile trovare risultati di tecniche di topic modelling applicate ad esso, permettendo quindi un utile confronto con esse.

20Newsgroups è organizzato in 20 topic differenti tra di loro. Alcuni di essi sono molto simili, come per esempio *comp.sys.ibm.pc.hardware* e *comp.sys.mac.hardware*, mentre altri sono completamente non correlati tra di loro, come per esempio *talk.politics.misc* e *talk.religion.misc*.

20Newsgroups è disponibile ed utilizzabile facilmente tramite la libreria Python *Scikit-learn* [8], dove ad ogni documento del dataset è associata un'etichetta in base al topic di appartenenza. Essendo i documenti estratti da gruppi di discussione online sono spesso presenti in essi parole che è meglio rimuovere per ottenere risultati migliori, come ad esempio le università di appartenenza dei mittenti oppure i loro indirizzi email. Per questo la funzione di Scikit-learn utilizzata per caricare 20Newsgroups mette a disposizione un parametro per indicare quali elementi rimuovere dai documenti che formano il dataset. Per caricare 20Newsgroups è stata usata la seguente funzione:

```
1 from sklearn.datasets import fetch_20newsgroups
2 import pandas as pd
3
4 newsgroups = fetch_20newsgroups(subset='all', remove=('headers',
    'footers', 'quotes'))
5 df = pd.DataFrame([newsgroups.data, newsgroups_train.target.tolist()]).T
6 df.columns = ['text', 'target']
```

3.4 Valutazione

Uno degli obbiettivi di questo lavoro di tesi è stato quello di testare varie tecniche di topic modelling applicate al dataset di comunicati stampa decritto precedentemente, al fine di trovare il metodo che permetta di ottenere i risultati migliori possibili. Per far ciò è stato necessario definire alcune metriche di valutazione da utilizzare per effettuare un confronto tra tutte le tecniche provate. Le seguenti metriche di valutazione sono state scelte perché sono spesso usate negli articoli scientifici inerenti il topic modelling e hanno permesso dunque di comparare i risultati ottenuti durante questo lavoro con quelli mostrati da tecniche già pubblicate.

3.4.1 Adjusted Rand index

L'Adjusted Rand Index (ARI) è una misura della similarità tra due diverse assegnazioni di etichette. Essa viene spesso utilizzata come metro di valutazione per le classificazioni fatte da un algoritmo di clustering, in quanto permette di confrontare le etichette di ground truth con quelle calcolate. L'Adjusted Rand Index è in grado di ignorare le permutazioni, ovvero non necessita che le etichette associate dall'algoritmo abbiano gli stessi valori di quelle di ground truth.

L'Adjusted Rand Index genera valori compresi tra -1 e 1. Nel caso l'algoritmo di clustering sia perfetto, ovvero che produce etichette che corrispondono a quelle di ground truth, si otterrà un valore di ARI pari ad 1. Un'assegnazione completamente casuale dei cluster produce un Adjusted Rand Index di 0, mentre il punteggio peggiore che si può ottenere è -1.

Prima di vedere come calcolare l'Adjusted Rand Index è necessario definire il Rand Index, proposto da Hubert e Arabie [26]. Dato un set di elementi $S = \{o_1, ...o_n\}$ e due partizioni di S da comparare, $X = \{X_1, ...X_r\}$ e $Y = \{Y_1, ...Y_s\}$, viene definito il Rand

Index come:

$$RI = \frac{a+b}{a+b+c+d}$$

dove:

- a è il numero di coppie di elementi di S che sono nello stesso set sia in X che in Y
- b è il numero di coppie di elementi di S che sono in set differenti sia in X che in Y
- c è il numero di coppie di elementi di S che sono nello stesso set in X ma in set differenti in Y
- d è il numero di coppie di elementi di S che sono in set differenti in X ma nello stesso set in Y

Il Rand Index ha il difetto che non restituisce il valore 0 per un'assegnazione di cluster casuale. Pertanto si utilizza l'Adjusted Rand Index calcolato come segue:

$$ARI = \frac{RI - valore\ stimato}{Valore\ massimo - valore\ atteso}$$

Per misurare l'Adjusted Rand Index è stata utilizzata la relativa funzione presente nella libreria Scikit-learn, mostrata nel seguente esempio:

```
1 from sklearn.metrics.cluster import adjusted_rand_score
2 adjusted_rand_score(ground_truth, labels)
```

3.4.2 Normalized Mutual Information

La Normalized Mutual Information (NMI) è una misura usata per valutare un partizionamento, per esempio ottenuto tramite un algoritmo di clustering, confrontandolo con un altro partizionamento di riferimento. Essa è la versione normalizzata della Mutual

Information (MI), utilizzata per misurare la mutua dipendenza tra due variabili casuali indipendenti tramite la formula:

$$I(X,Y) = \sum_{y \in Y} \sum_{x \in X} P(x,y)\ log \frac{P(x,y)}{P(x)P(y)}$$

La Normalized Mutual Information (NMI) è invece calcolata nel seguente modo:

$$NMI(X,Y) = \frac{2 \times I(X,Y)}{H(X) + H(Y)}$$

dove H(X) rappresenta l'entropia dei cluster X. NMI è un'ottima misura della qualità dei cluster ed, essendo normalizzata, può essere anche utilizzata per confrontare raggruppamenti con un numero diverso di cluster. Essa è stata calcolata usando la relativa funzione presente nella libreria Scikit-learn, mostrata nel seguente esempio:

```
1 from sklearn.metrics.cluster import normalized_mutual_info_score
2 normalized_mutual_info_score(ground_truth, labels)
```

3.4.3 Coerenza

Come si è già descritto, estrarre degli argomenti da un corpus significa ottenere degli insiemi di parole rappresentati ogni singolo topic che meglio descrivono i documenti. Se con l'Adjusted Rand Index si è valutato come si è suddiviso il corpus in base ai topic, è necessario anche usare una metrica di valutazione per misurare la qualità delle parole che descrivono gli argomenti. Si è deciso di utilizzare per questo scopo la coerenza (topic coherence in inglese).

La coerenza è misurata per ogni singolo topic calcolando la similarità semantica tra le parole più rappresentative di essi. Questa misura permette di distinguere bene topic formati da parole correlate semanticamente, ovvero identificabili in un contesto ben preciso, da topic che invece sono frutto dell'inferenza statistica.

Esistono varie misure per calcolare la coerenza. Durante la fase di test si è deciso di utilizzare la coerenza c_v, proposta nell'articolo *Exploring the Space of Topic Coherence*

Measures [37]. Essa considera le co-occorrenze di tutte le coppie di parole appartenenti ad un topic utilizzando una finestra scorrevole all'interno dei documenti. Per ognuna delle n parole più rappresentative del topic viene calcolato un vettore tramite la media delle Normalized Pointwise Mutual Information (NPMI) tra la parola stessa e tutte le altre del topic. La coerenza c_v del singolo argomento è trovata infine come la media delle similarità tra ognuno di questi vettori delle parole calcolate tramite la similarità del coseno.

La coerenza c_v è un valore compreso tra 0 e 1, dove 0 significa che le parole non sono per niente correlate mentre 1 significa che tutte le parole sono uguali tra di loro. Per questo il valore ideale di coerenza che si può ottenere è compreso tra 0.7 e 0.8.

Per calcolare la coerenza c_v è stata utilizzata la funzione presente all'interno della libreria gensim, mostrata nel seguente esempio:

```
1 from gensim.models.coherencemodel import CoherenceModel
2 cm = CoherenceModel(topics=topics, texts=token_lists,
      corpus=model.corpus, dictionary=model.dictionary, coherence='c_v')
3 cm.get_coherence()
```

Capitolo 4

Prima soluzione

In questo capitolo viene descritta la prima delle due soluzioni implementate per il problema presentato nel paragrafo 3.1. Essa è basata su top2vec, una tecnica di topic modelling pubblicata recentemente che permette di ottenere ottimi risultati. Vengono però effettuate alcune modifiche ad essa, in particolare utilizzando delle reti Transformer per generare l'embedding dei documenti. Dopo aver descritto top2vec ed analizzato la soluzione implementata durante il lavoro di tesi, vengono presentati i risultati qualitativi e quantitativi ottenuti.

4.1 Top2vec

Top2vec è un algoritmo di topic modelling ideato da Dimo Angelov e pubblicato ad Agosto 2020 nell'articolo *Top2Vec: Distributed Representations of Topics* [13]. Sebbene sia una tecnica molto recente, essa sfrutta algoritmi già conosciuti da tempo, ovvero Doc2vec, Universal Sentence Encoder, UMAP e HDBSCAN. Top2vec permette di estrarre i topic da un corpus con risultati davvero eccezionali e presenta inoltre tre vantaggi rispetto ad algoritmi come LDA:

1. Non è necessario rimuovere le stop-word prima di effettuare l'estrazione dei topic

2. Non è necessario effettuare lo stemming o il lemming delle parole

3. Top2vec identifica in maniera automatica il numero di topic presenti nel corpus. Non è necessario dunque definire questo numero a priori, come in LDA. Inoltre è possibile ridurre il numero di topic aggregando tra di loro quelli simili fino ad ottenere la quantità desiderata

Angelov dichiara nel suo articolo che tramite top2vec è possibile ottenere topic molto più informativi e rappresentativi del corpus rispetto ai modelli probabilistici generativi come LDA.

Top2vec si basa su metodi di embedding dei documenti e delle parole, descritti nel capitolo 2, per creare una rappresentazione distribuita dove testi che trattano gli stessi argomenti sono posizionati vicini. Come scritto nell'articolo di Angelov, per capire questo concetto è utile spiegare il significato di rappresentazione distribuita dei documenti e delle parole. Essa significa che in una rete neurale ogni singolo concetto è imparato e rappresentato da più neuroni. Per questo quando un peso della rete cambia per incorporare nuova conoscenza, questi cambiamenti influenzano anche i concetti simili a quello che si è imparato. Le rappresentazione distribuite hanno il vantaggio di portare ad una generalizzazione automatica dei concetti e per questo sono spesso utilizzate nelle tecniche di NLP per imparare le rappresentazioni vettoriali delle parole e dei documenti.

Un altro concetto importante è quello dell'ipotesi distributiva, descritta da John Rupert Firth [22]. Essa dice che parole che occorrono nello stesso contesto tendono ad avere significati simili. Proprio per questo è più probabile trovare termini simili all'interno dello stesso documento, piuttosto che in testi diversi.

I modelli CBOW e Skip-gram di word2vec hanno introdotto il concetto di rappresentazione distribuita che cattura le relazioni semantiche e sintattiche tra le parole. Doc2vec ha esteso questi modelli riuscendo ad imparare anche una rappresentazione distribuita di singoli paragrafi di testi oppure di interi documenti. Tramite word2vec

e doc2vec viene quindi generato uno spazio semantico, dove le distanze tra i vettori rappresentano le associazioni semantiche tra le parole o i documenti. E' importante notare come, utilizzando queste tecniche, nello spazio ottenuto unendo quello creato dall'embedding delle singole parole e quello creato dall'embedding degli interi documenti, le parole sono posizionate vicine ai documenti con significato semantico simile. Questa proprietà può essere dunque utilizzata per molte operazioni, come per esempio per ricercare documenti inerenti ad una determinata parola oppure sopratutto per estrarre i termini che rappresentano meglio i topic trattati da un testo. Top2vec utilizza questa assunzione per proporre un metodo capace di imparare una rappresentazione distribuita anche per i topic. Genera dunque uno spazio congiunto degli embedding delle parole, dei documenti e dei topic, tale per cui la distanza tra essi rappresenta lo loro similarità semantica.

Osserviamo adesso il funzionamento di top2vec e come esso genera i topic a partire da un corpus:

1. Si utilizza un modello di rete neurale per generare l'embedding dei documenti e di tutte le singole parole che li compongono. Top2vec permette di utilizzare due differenti modelli:

    - Doc2vec è usato quando si ha un dataset molto grande e con molti termini specifici. La rete di doc2vec sarà allenata a partire dal corpus utilizzando il modello Distributed Bag of Words (PV-DBOW), visto che Angelov ha osservato ch peremette di ottenere risultati migliori rispetto al modello Distributed Memory Version of Paragraph Vector (PV-DM)
    - Universal Sentence Encoder (USE) [18] è utilizzato quando si ha un dataset più piccolo dato che esso è un modello pre-allenato. E' generalmente più veloce di doc2vec ed è utilizzabile anche in varie lingue diverse dall'inglese. L'Universal Sentence Encoder è stato rilasciato da Google nel 2019 ed è infat-

ti un modello di embedding addestrato in 16 lingue e contemporaneamente su più compiti

Si ottiene quindi uno spazio n-dimensionale dove ogni punto presente in esso può essere descritto dai vettori delle parole più vicine.

2. Nello spazio semantico ottenuto, se si ha un'area densa di documenti significa che essi sono molto simili tra di loro e che è probabile che esista un topic che li accomuni tutti quanti. Per trovare queste aree viene utilizzato Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN), un algoritmo di clustering basato sulla densità. Tuttavia i vettori dei documenti, trovandosi in uno spazio di dimensione elevata, sono molto sparsi e per questo è difficile, oltre che oneroso dal punto di vista computazionale, riuscire a trovare i cluster correttamente. Top2vec perciò riduce la dimensione dello spazio attraverso l'algoritmo Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) e solo successivamente ricerca i cluster tramite HDBSCAN.

3. Una volta che sono stati trovati i cluster e i documenti sono stati associati ad essi tramite un'etichetta, è possibile identificare i vettori che rappresentano i topic calcolando il centroide di ogni cluster

4. Nello spazio semantico identificato, ogni punto rappresenta un topic che è descritto al meglio dalle parole che gli sono vicine. Per questo top2vec utilizza i termini meno distanti da ogni vettore che rappresenta un topic per descriverli. Inoltre la distanza tra le parole e i topic indicano la similarità semantica tra di essi.

4.2 Implementazione

Viene di seguito analizzata la prima soluzione implementata per risolvere il problema descritto nel capitolo 3. Essa si basa sull'architettura generale di top2vec ma ne

modifica alcune parti per provare ad ottenere dei risultati migliori.

4.2.1 Embedding

Per calcolare i vettori di embedding dei documenti del corpus si è deciso di testare un approccio differente rispetto a top2vec. Si è infatti utilizzata una rete neurale Transformer pre-allenata per ottenere le rappresentazioni vettoriali dei documenti. Come spiegato nel capitolo 2, le reti Transformer utilizzano un modello seq2seq costituito da un enccoder e da un decoder. L'encoder serve per codificare la sequenza testuale presa in input in un vettore di uno spazio n-dimensionale. Il decoder invece calcola la rappresentazione della sequenza di output tramite il task desiderato. Con il processo di allenamento l'encoder impara quindi ad ottenere una rappresentazione vettoriale coerente della sequenza di input, in modo simile a doc2vec, mentre il decoder impara a produrre il risultato di output. Sia l'encoder che il decoder sfruttano inoltre il meccanismo di attenzione descritto nel paragrafo 2.3.3. L'obbiettivo dell'embedding resta comunque lo stesso di top2vec, ovvero quello di ottenere uno spazio distribuito che cattura le relazioni semantiche tra i documenti e dove i vettori dei testi simili sono posizionati vicini.

Per testare questo metodo si è deciso di provare ad utilizzare due reti differenti di Transformer, BERT e T5. BERT è già stato analizzato nel paragrafo 2.3.4 ed è uno dei modelli di Transformer più utilizzati nel settore del NLP. T5 è invece un modello di rete neurale realizzata da Google nel 2019 [35]. La sua particolarità è di essere un modello text2text, ovvero che prende in ingresso un testo e produce in output sempre un altro testo. Questa formattazione consente a T5 di essere allenata su numerosi task di NLP. Il modello di base di T5 è pre-allenato su C4 (Colossal Clean Crawled Corpus), un dataset estremamente grande di testi in inglese non etichettati. Successivamente l'autore ha allenato nuovamente il modello tramite fine-tuning su cinque task differenti, come per esempio la traduzione tra lingue o la generazione di un riassunto a partire

da un testo. T5 permette di ottenere risultati che rappresentano lo stato dell'arte su numerose operazioni, ma si può utilizzare su esso tecniche di fine-tuning per imparare nuovi task semplicemente avendo a disposizione un dataset di allenamento contenente esempi di testi di input e di output.

Tutti i modelli di BERT e di T5 utilizzati sono stati estratti dalla libreria Huggingface descritta nel paragrafo 3.2.4. Si sono però usati due metodi diversi per calcolare i vettori di embedding dei documenti.

Per ottenere la rappresentazione tramite BERT si è sfruttata la libreria Sentence-Transformer descritta nel paragrafo 3.2.5. Essa permette infatti di generare facilmente una rete capace di calcolare l'embedding dei documenti a partire da un modello di Transformer desiderato della libreria Huggingface. SentenceTransformer è stata implementata come mostrato nel seguente esempio di codice:

```
from sentence_transformers import SentenceTransformer, models

word_embedding_model =
    models.Transformer('bert-base-uncased', max_seq_length=256)
pooling_model =
    models.Pooling(word_embedding_model.get_word_embedding_dimension())
model =
    SentenceTransformer(modules=[word_embedding_model, pooling_model])

documents_vectors = model.encode(documents)
```

La rete è generata unendo un modello di Transformer con un livello di pooling. La classe *Transformer* definisce infatti tramite il suo primo parametro il nome del modello presente nella libreria Huggingface da utilizzare e, tramite il parametro *max_seq_length*, il numero massimo di token che possono essere generati da ogni documento. La classe *Pooling*, invece, definisce come utilizzare gli embedding dei singoli token ottenuti tramite il Transformer per creare un unico vettore che rappresenti il documento. Essa

calcola di base la media tra i vettori dei token, ma possono essere usate anche altre tecniche come per esempio la somma di essi. Il pooling utilizza come parametro la dimensione dello spazio di embedding.

Trovare invece i vettori dei documenti tramite T5 è stato più elaborato, dato che esso non è ancora supportato dalla libreria SentenceTransformer. Si sono usate varie funzioni offerte da Huggingface per ottenere comunque lo stesso risultato. Innanzitutto è stata utilizzata la classe *T5Tokenizer* per generare i token di ogni documento. Essa necessita in ingresso un testo e ritorna un oggetto contenente gli input ids, ovvero un array in cui per ogni token della frase è associato un numero intero unico. Per esempio se convertiamo la frase "I will go to school tomorrow" otteniamo i token:
['I', 'will', 'go', 'to', 'school', 'tomorrow', '</s>']
a cui vengono associati i seguenti input ids:
[27, 56, 281, 12, 496, 5721, 1]
T5Tokenizer inserisce sempre il valore </s> come ultimo token perché è utilizzato successivamente dalla rete di T5 per capire quando termina il documento. Per riuscire a calcolare i token di un insieme di documenti è necessario definire anche i parametri padding e truncate. Il padding serve per inserire al termine dell'ultimo input id di ogni token degli elementi con valore 0 fino ad arrivare alla dimensione del vettore pari a 512. Il parametro truncate è utilizzato invece per considerare solamente i primi 512 token del vettore se esso ha dimensione maggiore. In questo modo si potrà rappresentare tutti i documenti con un'unica matrice di larghezza 512. Un esempio di utilizzo della funzione T5Tokenier è il seguente:

```
1 from transformers import T5Tokenizer
2
3 tokenizer = T5Tokenizer.from_pretrained('t5-base')
4 input_ids = tokenizer(documents, padding=True, truncation=True).input_ids
```

Una volta ottenuti gli id dei token, essi sono stati utilizzati per ottenere i corrispondenti vettori tramite la parte di encoding di un modello di rete T5. Si è infatti usata la classe T5EncoderModel di Huggingface, inizializzata definendo il modello di T5 desiderato e passandogli come parametro gli input ids. La rappresentazione vettoriale dei token è ottenuta prendendo i valori dei pesi dell'ultimo livello della rete T5 generati da essa. Infine si sono calcolati i vettori di embedding dei singoli documenti calcolando la media dei vettori dei token. Ogni testo è quindi rappresentato tramite un vettore di dimensione 512, calcolato come mostrato nel seguente esempio:

```
from transformers import T5EncoderModel

model_encode = T5EncoderModel.from_pretrained('t5-base',
    output_hidden_states = True)
batch_size = 100
token_embeddings = []

for i in range(0, len(df), batch_size):
  with torch.no_grad():
    result = model_encode(input_ids =
        input_ids[i:i+batch_size]).last_hidden_state
    token_embeddings.append(result)

token_embeddings = torch.cat(token_embeddings)
documents_embeddings = torch.mean(token_embeddings, dim=1)
```

Utilizzando sia BERT che T5 si ottengono quindi delle rappresentazioni dei documenti in uno spazio multidimensionale dove la distanza tra vettori è indice della loro similarità semantica. Si sono testati vari modelli di BERT e T5 presenti nella libreria Huggingface per cercare di ottenere una rappresentazione che rispetti il meglio possibile la semantica dei documenti. Un esempio di spazio di embedding ottenuto tramite BERT sul dataset 20Newsgroups è mostrato nella figura 4.1.

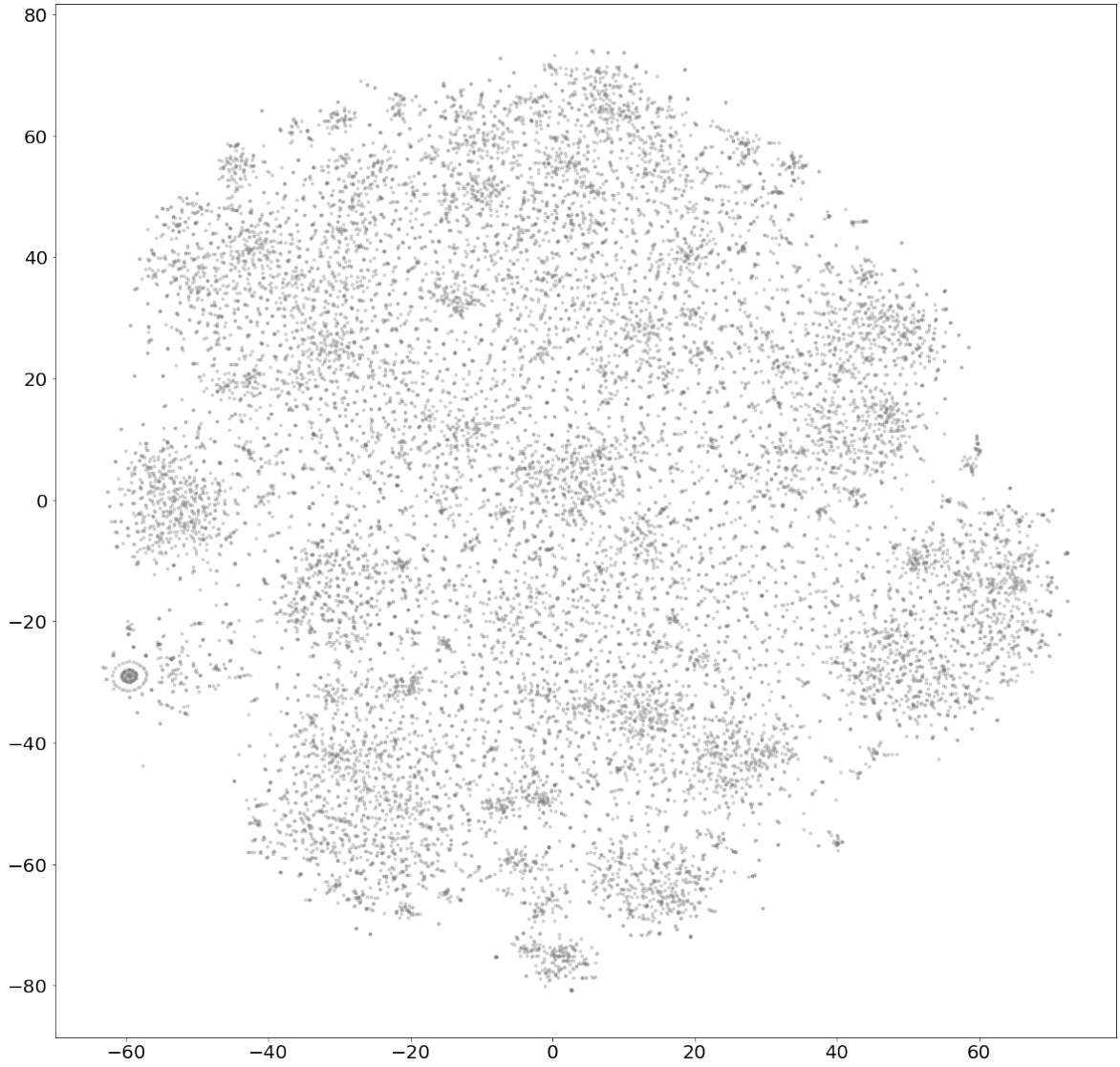

Figura 4.1: Embedding dei documenti del dataset 20Newsgroups ottenuto utilizzando BERT e visualizzato riducendo la dimensione tramite t-SNE

4.2.2 Clustering

Nello spazio semantico identificato tramite l'embedding con una rete Transformer, un'area densa di vettori può essere interpretata come una zona dove i corrispondenti documenti sono molto simili tra di loro e probabilmente tratteranno degli stessi argomenti. Per trovare queste aree sono state testate e confrontate due diverse tecniche di clustering: K-Means e HDBSCAN.

K-Means è un algoritmo di apprendimento non supervisionato, proposto da Stuart

Lloyd nel 1957, che trova in un insieme di dati un numero fisso di cluster, ovvero gruppi che dividono gli oggetti a seconda della presenza o meno di una certa somiglianza tra di loro. L'algoritmo è di per sé molto semplice ed esegue in maniera iterativa i seguenti passaggi:

1. Si inizializza l'algoritmo definendo un numero k di cluster desiderati e quindi scegliendo k centroidi disposti casualmente
2. Ogni elemento viene associato al centroide con cui la distanza è minima
3. Si ricalcola la posizione dei centroidi come il punto medio di tutti gli elementi associati allo stesso cluster
4. Si ripetono i passaggi 2 e 3 fino a quando le assegnazioni degli elementi nei cluster non cambiano oppure si è raggiunto il numero di iterazioni desiderato

Quando si utilizza K-Means è dunque fondamentale scegliere il numero k di cluster corretto. Nei test effettuati tramite il dataset 20Newsgroup questo valore era già conosciuto a priori, visto che si sapeva che i documenti erano suddivisi in 20 topic differenti. Quando invece si è testato K-Means sul dataset formato dai comunicati stampa del governo inglese è stato invece necessario provare vari valori diversi di k.

Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) è invece un algoritmo di clustering ideato da Campello, Moulavi e Sander nel 2013 [17]. A differenza di K-Means, HDBSCAN non richiede di definire a priori il numero di cluster desiderato, dato che l'algoritmo li identifica ricercando le aree dello spazio dense di vettori. In particolare HDBSCAN si può suddividere in tre passi principali:

1. Si stima la densità di ogni elemento utilizzando la core distance. Definito un numero k di vicini che si vuole trovare rispetto ad un punto , la core distance indica il raggio minimo dell'ipersfera che contiene il numero k di vicini considerando il

punto al centro. Punti in regioni dense avranno un valore di core distance basso mentre punti in regioni dove gli elementi sono sparsi avranno delle core distance maggiori. Inoltre calcolando l'inverso delle core distance di un punto si può derivare una stima della densità degli elementi in quell'area

2. Si costruisce una struttura ad albero considerando le densità calcolate precedentemente ed ottenendo i cluster più significativi

3. Attraverso l'applicazione di una misura di stabilità l'algoritmo valuta se ogni cluster può essere suddiviso in due o più cluster

E' importante osservare come con HDBSCAN non tutti gli elementi dello spazio vengono associati ad un cluster. L'algoritmo ricerca le zone dense di punti ma tutti i vettori che sono al di fuori di esse vengono identificati come outliers.

Sorgono però due problemmi nell'utilizzare direttamente un algoritmo di clustering nello spazio ottenuto tramite l'embedding con un Transformer. Il primo è che la dimensione dello spazio è molto elevata, per esempio utilizzando T5-base si ottengono vettori di dimensione pari a 512. Per questo i vettori saranno molto sparsi e sarà difficile identificare correttamente i cluster in base alla densità. Il secondo problema è che avendo una dimensione così elevata il costo computazionale degli algoritmi di clustering sarà molto alto. E' necessario perciò utilizzare una tecnica di riduzione della dimensionalità sui vettori dei documenti. In particolare si è usato per la soluzione l'algoritmo UMAP.

Uniform Manifold Approximation and Projection (UMAP) è un algoritmo di riduzione della dimensionalità pubblicato nel 2018 [32]. Il suo funzionamento è molto simile a quello di t-SNE, in quanto entrambi utilizzano una rappresentazione a grafi dei dati nella dimensione originale per ottimizzare un grafo in una dimensione minore in modo che siano il più simile possibile dal punto di vista strutturale. Per determinare il grafo originale UMAP utilizza un determinato raggio e considera connessi tutte le coppie di punti dello spazio che hanno una distanza minore di esso. Ad ogni connessione è associato un peso che dipende dalla vicinanza tra i punti. E' importante quindi scegliere

il valore di tale raggio correttamente, in quanto se è troppo basso si avranno molti punti isolati mentre se è troppo alto si avranno troppi punti connessi. Per risolvere questo problema UMAP utilizza un raggio variabile per ogni punto che dipende dalla distanza di un numero n di punti più vicini ad esso. Questo permette di preservare la struttura locale dei punti bilanciandola con quella globale. Una volta che il grafo nella dimensione originale è stato costruito, UMAP lo utilizza per ottimizzare quello nella dimensione ridotta in modo simile a t-SNE.

La riduzione della dimensionalità e la ricerca dei cluster sono stati implementati utilizzando varie librerie:

- L'algoritmo di UMAP è stato implementato utilizzando l'omonima libreria scritta in Python [12], impostando alcuni valori. Il parametro *n_neighbors* indica il numero di elementi da osservare per stimare la struttura del grafo dei dati. Minore è il valore e più si preserverà la struttura locale. Il parametro *n_components* indica la dimensione dello spazio che si vuole ottenere tramite la riduzione. Il parametro *metric* indica invece quale metrica utilizzare per calcolare le distanze.

- L'algoritmo di K-Means è stato implementato utilizzando la relativa funzione presente nella libreria Scikit-learn. Essa necessita solamente di impostare il parametro *n_clusters* che identifica il numero di cluster che si vuole ottenere.

- L'algoritmo di HDBSCAN è stato implementato utilizzando l'omonima libreria scritta in Python [2]. Il parametro *min_cluster_size* indica il numero minimo di elementi che devono essere presenti in un cluster.

Un esempio di utilizzo di queste funzioni è il seguente:

```
1 from umap import UMAP
2 from sklearn.cluster import KMeans
3 from hdbscan import HDBSCAN
4
5 umap = UMAP(n_neighbors=15, n_components=5, metric='cosine')
```

```
6 documents_reduced = umap.fit_transform(documents)
7
8 kmeans = KMeans(n_clusters=20)
9 clusters = kmeans.fit(documents_reduced)
10
11 hdbscan = HDBSCAN(min_cluster_size=7)
12 clusters = hdbscan.fit(documents_reduced)
```

Un esempio dei cluster identificati tramite K-Means nel dataset 20Newsgroups è mostrato nelìa figura 4.2.

Se utilizzando l'algoritmo K-Means si ottiene il numero desiderato di cluster, usando HDBSCAN si identifica invece un raggruppamento per ogni area densa di elementi che viene trovata. Può succedere dunque che il numero di cluster identificato sia davvero elevato e quindi non rappresenti il numero effettivo dei topic esistenti nel corpus. Quando il numero di cluster trovato da HDBSCAN è maggiore di una certa soglia si è utilizzato dunque il seguente algoritmo di riduzione:

1. Si calcolano i centri di ogni cluster
2. Si sceglie il cluster con il minor numero di documenti e il cluster più vicino ad esso, trovato calcolando la distanza euclidea tra i loro centri
3. Si aggregano i due cluster associando la stessa etichetta a tutti i documenti presenti in essi
4. Si ripetono i passi precedenti fino ad ottenere il numero di cluster desiderato

4.2.3 Estrazione dei topic

Tramite il processo di clustering si ottiene quindi una classificazione dei documenti. Ad ognuno di essi è infatti associata un'etichetta che li raggruppa, presupponendo che tutti

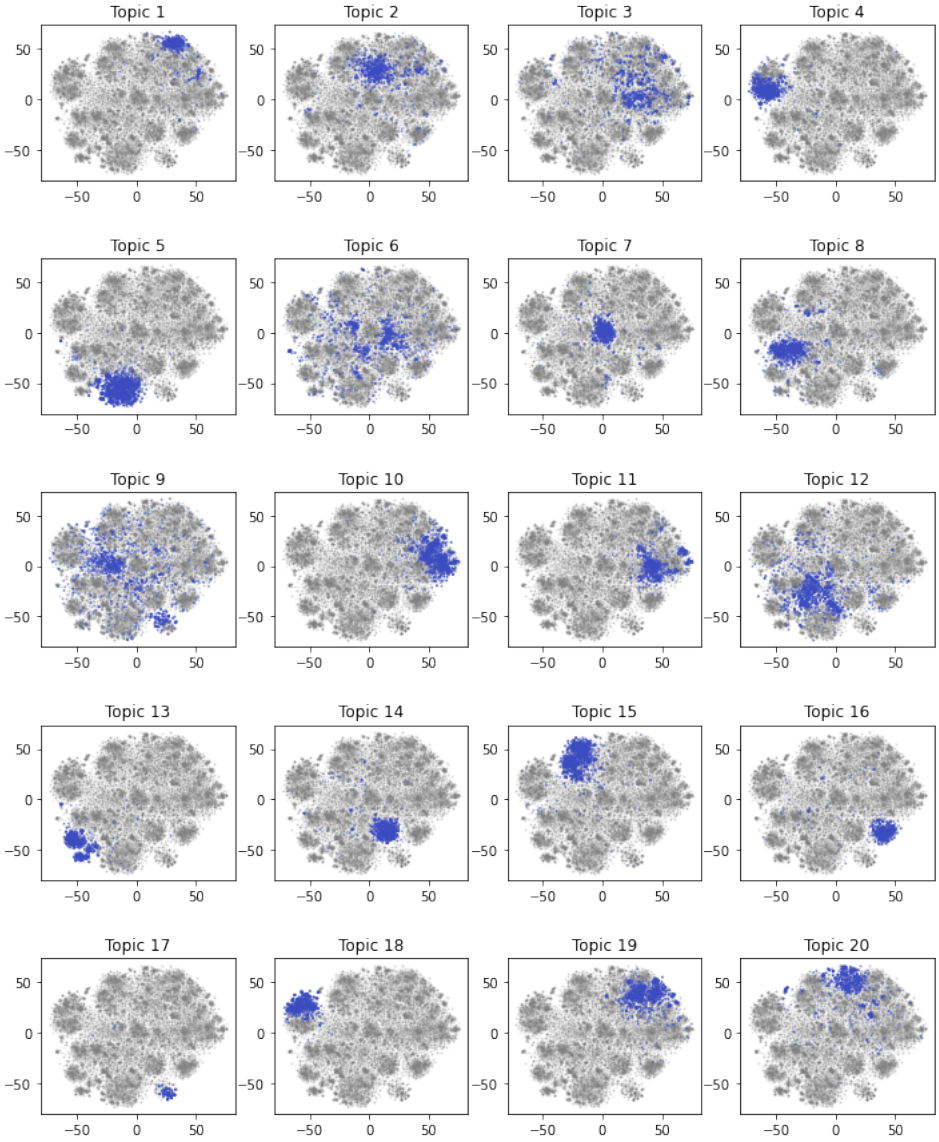

Figura 4.2

i documenti di uno stesso cluster trattino degli stessi argomenti semantici. Come per gli altri algoritmi di topic modelling, è necessario infine ottenere le parole che meglio rappresentano gli argomenti tematici trattati dai documenti di ogni cluster.

Top2vec utilizza l'embedding di tutte le singole parole che costituiscono il corpus per cercare quelle più vicine ai vettori dei topic. Questo è possibile perché l'algoritmo di embedding utilizzato da top2vec, ossia doc2vec oppure Universal Sentence Encoder, mappa la stessa parola presente in due documenti differenti con lo stesso vettore. Utilizzando invece un modello di rete Transformer, l'embedding delle parole viene generato

a partire dal contesto della frase e sfruttando il meccanismo di attenzione. Per questo è possibile che la stessa parola sia mappata con due vettori distanti se è presente in documenti che trattano contesti differenti. Il metodo utilizzato da top2vec per trovare i termini che descrivono i topic non può perciò essere sfruttato utilizzando una rete Transformer.

E' stato dunque utilizzato per estrarre le parole l'algoritmo Class-based TF-IDF, abbreviato come C-TF-IDF. Esso si basa sui principi di TF-IDF, descritto nel paragrafo 2.2.1, ma li applica non a livello di singolo documento ma a livello di documenti appartenenti allo stesso cluster. TF-IDF associa ad ogni parola in un documento un valore proporzionale al numero di volte che essa è presente nel documento e inversamente proporzionale al numero di volte che essa è presente in tutto il corpus. C-TF-IDF utilizza lo stesso metodo ma lo applica ad ogni classe, invece che ad ogni singolo documento. Si ottiene dunque un punteggio per ogni parola rispetto ad ogni cluster, calcolato con la seguente formula:

$$(C - TF - IDF)_i = \frac{t_i}{w_i} \cdot log \frac{m}{\sum_j^n t_j}$$

dove t_i è il numero di volte che la parola t compare all'interno della cluster i, w_i è il numero totale di parole presenti nel cluster i, m è il numero totale di documenti del dataset e n è il numero di cluster. Infine, per selezionare le parole più rappresentative di ogni topic, vengono scelti i termini con il punteggio C-TF-IDF più alto all'interno di ogni cluster.

4.2.4 Fine-tuning

Per migliorare la rappresentazione vettoriale generata tramite le reti Transformer e per ottenere quindi dei cluster più precisi, si è deciso di utilizzare delle tecniche di fine-tuning sui modelli utilizzati estratti dalla libreria Huggingface. Il fine-tuning è un meccanismo con il quale un modello di rete, pre-allenato su un determinato dataset, viene continuato ad essere allenato tramite un task su un altro dataset, che può essere

per esempio quello su cui si vogliono ottenere i risultati. La rete modifica quindi i propri pesi sulla base dei nuovi dati utilizzati e producendo quindi numerosi vantaggi:

1. I pesi di un modello pre-allenato contengono molte informazioni riguardo un linguaggio. Per questo il processo di fine-tuning richiede poco tempo di allenamento per essere effettuato e per ottenere netti miglioramenti nei risultati

2. Il dataset che viene utilizzato per il fine-tuning può essere molto piccolo rispetto a quello utilizzato per allenare il modello originario

3. Il fine-tuning permette di ottenere risultati migliori rispetto al modello di rete originale. In particolare, applicandolo ad una rete Transformer, essa si specializzerà maggiormente sul dataset utilizzato per il fine tuning e questo porta a molti vantaggi quando, per esempio, si hanno dei documenti riguardanti argomenti molto specifici o con alcuni termini che non erano presenti nel dataset originario

Il fine-tuning è stato applicato sia a BERT che a T5, utilizzando però due tecniche differenti.

Per quanto riguarda i modelli di BERT, è stato applicato loro il fine-tuning utilizzando uno script presente direttamente all'interno della libreria Huggingface. Esso permette di continuare ad allenare il modello di Transformer selezionato tramite il task di Masked LM descritto nel paragrafo 2.3.4. Esattamente come è stato allenato il modello originale di BERT, il 15% dei token generati per ogni documento del dataset viene mascherato con il token [MASK] e la rete sarà allenata cercando di predire questi termini mancanti ed utilizzando una funzione di loss. Il dataset su cui è effettuato il fine-tuning viene inoltre suddiviso nel dataset di allenamento, selezionando casualmente l'80% dei documenti del corpus, e nel dataset di validazione, utilizzando il restante 20%. Lo script necessita inoltre come parametri il batch size, ovvero il numero di esempi che vengono utilizzati ad ogni passaggio, il learning rate e il numero di epoche.

Il fine-tuning tramite BERT è stato testato sia sul dataset 20Newsgroup che su quello generato con i comunicati stampa.

Per il fine-tuning dei modelli di Transformer T5 è stata invece utilizzata Simple Transformers [10], una libreria che permette di allenare alcune delle reti presenti in Huggingface su numerosi task di NLP. Come descritto precedentemente, T5 è un Transformer con una architettura costituita da un encoder e da un decoder, dove entrambi utilizzano come dati in ingresso e in uscita delle sequenze testuali. Questo permette quindi di avere molta flessibilità nell'eseguire qualsiasi tipo di task di NLP senza dover applicare nessuna modifica all'architettura del modello, essendo sufficiente fornire alla rete degli esempi in forma testuale su cui allenarsi. Per insegnare a un modello T5 un nuovo task è necessario inserire un prefisso unico all'inizio di ogni sequenza di input, come per esempio "summarize:". E' inoltre possibile insegnare più task alla volta al modello durante un unico allenamento utilizzando prefissi differenti. Per migliorare la rappresentazione vettoriale dei documenti, T5 è stato allenato tramite fine-tuning nel task di predizione dei titoli dei comunicati stampa a partire dal testo dell'articolo. Sono stati dunque forniti come esempi alla rete in input il testo del comunicato preceduto dal prefisso "get_title:" e in output il relativo titolo. Il fine-tuning tramite T5 è stato potuto essere testato solamente sul dataset dei comunicati stampa estratti dal sito *gov.uk* e non su 20Newsgroups, in quanto esso non presenta i titoli dei testi del corpus. Alcuni esempi di predizione di titoli ottenuti dal modello di T5 a cui è stato applicato il fine-tuning sono mostrati nella tabella 4.1.

4.3 Risultati

Il metodo di estrazione dei topic descritto in questo capitolo è stato testato utilizzando i seguenti modelli di BERT e T5:

- *bert-base-uncased*: modello pre-allenato di BERT in lingua inglese che non distin-

Titolo originale	Titolo predetto
Reducing size of government estate secures more than £2bn	Government reduces size of government estate by a third over 10 years
Oliver Dowden speech to the Law Family Commission on Civil Society	Lord Dowden's speech on the Law Family Court
First virtual citizenship ceremony welcomed by Home Office minister	Home Office's first virtual citizenship ceremony

Tabella 4.1: Esempi di predizione di titoli di comunicati stampa ottenuti tramite il fine-tuning di T5

gue parole che iniziano con una lettera minuscola da quelle che iniziano con una lettera maiuscola. Produce vettori con dimensione 768. E' stata testata anche la versione con fine-tuning tramite il task di Masked LM.

- *distilbert-base-uncased*: modello leggero e veloce di Transformer ottenuto da un processo di distillazione di bert-base-uncased. Ha infatti il 40% di parametri in meno ed è il 60% più veloce, preservando comunque le stesse performance di BERT. Produce vettori con dimension 768. E' stata testata anche la versione con fine-tuning tramite il task di Masked LM.

- *T5-small*: modello di T5 pre-allenato sul dataset C4. Produce vettori di dimensione 512.

- *T5-base*: modello di T5 pre-allenato sul dataset C4. Produce vettori di dimensione 768. E' stata testata anche la versione con fine-tuning tramite il task di predizione dei titoli

- *T5-large*: modello di T5 pre-allenato sul dataset C4. Produce vettori di dimensione 1024.

Per valutare i risultati sono invece state utilizzate le metriche descritte nel paragrafo 3.4. In particolare è stato usato l'Adjusted Rand Index (ARI) per confrontare la classificazione effettuata tramite i metodi di clustering con i valori delle etichette originali del dataset 20Newsgroups. In questo modo si può verificare quanto i documenti sono stati raggruppati correttamente in base al topic di cui trattano, verificando quindi la qualità dell'embedding prodotto e della tecnica di clustering. Per valutare invece la scelta delle parole che descrivono ogni topic è stata utilizzata la coerenza c_v, una misura che descrive la similarità semantica tra le parole di ogni argomento. I valori di ARI e di coerenza ottenuti sono stati utilizzati per confrontare i modelli tra di loro e scegliere quello migliore. Sono stati inoltre calcolate le stesse metriche utilizzando top2vec, in modo da avere dei risultati di riferimento da cercare di migliorare.

I risultati ottenuti utilizzando i vari modelli di Transformer e K-Means come algoritomo di clustering sono mostrati nella tabella 4.2. Si può notare innanzitutto come sia i due modelli di BERT che quello di T5 siano migliorati se gli viene applicato il fine-tuning. Questo è dovuto al fatto che il modello si specializza maggiormente sui documenti che formano il corpus e quindi permetterà di ottenere dei vettori di embedding che rappresentano meglio la loro semantica.

Se paragonati con top2vec, i risultati mostrati nella tabella 4.2 sono nettamente inferiori. In particolare i valori di ARI ottenuti da tutti i modelli testati sono molto bassi. Questo indica che i vettori di embedding generati utilizzando delle reti Tranfsormer non sono particolarmente rappresentativi dei contesti dei documenti e quindi successivamente non saranno raggruppati bene dall'algoritmo di clustering. Tuttavia, come si nota dalla tabella 4.3, per alcuni dei topic sono state estratte delle parole coerenti tra di loro e che identificano argomenti ben specifici presenti anche nel dataset di 20Newsgroups, come ad esempio i topic 4, 6 e 14. Questo dimostra che comunque una parte dei documenti sono stati raggruppati in maniera abbastanza precisa e che il metodo di estrazione delle parole tramite C-TF-IDF si dimostra corretto. Confrontando infatti i topic descritti nella tabella 4.3 con quelli originali di 20Newsgroups si

Modello	ARI	NMI	MI	Coerenza
bert-base-uncased	0.186	0.335	0.978	0.446
bert-base-uncased con fine-tuning	0.204	0.345	1.010	0.472
distilbert-base-uncased	0.189	0.338	0.981	0.453
distilbert-base-uncased con fine-tuning	0.245	0.393	1.152	0.478
T5-small	0.091	0.193	0.568	0.429
T5-base	0.150	0.297	0.876	0.469
T5-base con fine-tuning	0.163	0.305	0.902	0.471
top2vec	0.314	0.466	1.381	0.636

Tabella 4.2: Risultati ottenuti utilizzando K-Means come algoritmo di clustering e C-TF-IDF per l'estrazione delle parole

nota come gli argomenti che trattano temi particolarmente specifici siano stati identificati correttamente in quanto i vettori dei loro documenti saranno stati posizionati vicini, come ad esempio *rec.motorcycle*, *sci.space* o *talk.religion.misc*. Topic differenti ma che trattano di argomenti simili, invece, non sono stati distinti bene dall'algoritmo e questo comporta dei valori di ARI e di NMI bassi. In conclusione si può comunque dire che utilizzare un approccio basato esclusivamente sull'embedding tramite modelli di Transformer non risulta consigliato, in quanto usando reti come word2vec o doc2vec si ottengono rappresentazioni migliori dei documenti.

topic 1	turks	istanbul	genocide	ottoman	armenia
topic 2	nsa	koresh	clinton	arabs	compound
topic 3	serial	motherboard	modem	pins	irq
topic 4	jehovah	scripture	heaven	sin	eternal
topic 5	ini	icon	config	exe	compile
topic 6	bike	helmet	ride	riding	bikes
topic 7	espn	pitching	braves	playoffs	players
topic 8	yeah	joke	hey	flame	yup
topic 9	appreciate	thank	advance	appreciated	responses
topic 10	motherboard	shipping	meg	sale	mhz
topic 11	lib	03	usr	buf	wolverine
topic 12	feustel	chemistry	netcom	lavrencic	tcora
topic 13	mov	apartment	sumgait	mamma	db
topic 14	orbit	mission	shuttle	spacecraft	moon
topic 15	ford	toyota	honda	engine	rear
topic 16	vitamin	chronic	doctor	pain	diet
topic 17	45th	incredibly	subscrive	loser	retarded
topic 18	2tm	2di	giz	75u	7ey
topic 19	barbara	santa	mcguire	campus	whirrr
topic 20	consistently	dir	exit	ken	ignore

Tabella 4.3: Parole maggiormente rappresentative dei topic estratti da 20Newsgroups utilizzando BERT, K-Means e C-TF-IDF

Capitolo 5

Seconda soluzione

In questo capitolo viene descritta la seconda soluzione implementata per il problema presentato nel paragrafo 3.1. Essa riprende alcuni elementi discussi nel capitolo precedente, in particolare l'embedding generato tramite reti Transformer, ma aggiunge ulteriori tecniche per produrre una soluzione che permette di ottenere risultati migliori. Dopo aver descritto nel dettaglio l'architettura della soluzione implementata, vengono presentati i risultati ottenuti con essa applicandola al dataset 20Newsgroups e a quello composto dai comunicati stampa.

5.1 Implementazione

Viene di seguito analizzata la seconda soluzione implementata per risolvere il problema descritto nel capitolo 3. Essa è basata sull'architettura generale presentata da Steve Shao nell'articolo *Contextual Topic Identification* [38], aggiungendo però il fine-tuning della rete Transformer utilizzata e modificando la tecnica di estrazione delle parole che rappresentano i topic.

L'architettura della soluzione è mostrata nella figura 5.1. Si può notare come essa sia costituita da quattro sezioni, esattamente come la prima soluzione implementata.

Innanzitutto si ottengono le rappresentazioni vettoriali dei documenti tramite la concatenazione delle distribuzioni dei topic ricavate con LDA e i vettori ottenuti tramite BERT. Successivamente si utilizza un autoencoder per imparare una rappresentazione dei documenti in uno spazio di dimensione minore, nel quale si utilizza una tecnica di clustering per classificare i testi in base al loro topic di appartenenza. Infine si estraggono dai cluster le parole più rappresentative degli argomenti utilizzando C-TF-IDF. Questo approccio cerca dunque di risolvere la bassa accuratezza della soluzione mostrata nel capitolo precedente nel generare vettori di embedding che siano rappresentativi dei contesti semantici dei documenti, sfruttando le informazioni ottenute tramite LDA.

5.1.1 Preprocessing ed embedding

Come descritto nel capitolo precedente, l'utilizzo esclusivo di una rete Transformer per ottenere le rappresentazioni vettoriali dei documenti si è dimostrato essere meno accurato di altre reti come doc2vec. Per soccombere a questo problema e per sfruttare comunque la capacità dei Transformer di imparare i contesti semantici dei documenti, si è testata la soluzione mostrata da Steve Shao nell'articolo *Contextual Topic Identification* [38], modificandone alcune parti. Essa propone di ricavare l'embedding dei documenti dalla concatenazione dei valori di ogni vettore di un documento ottenuti tramite BERT con le probabilità, ricavate tramite LDA, dei topic di appartenere al documento. In questo modo si uniscono i vantaggi di entrambe le tecniche:

- Tramite BERT si ha il vantaggio di ottenere una rappresentazione basata sul contesto semantico imparato dalla rete utilizzando il meccanismo di attenzione.
- Tramite LDA si ha il vantaggio di avere un assegnamento dei topic nei documenti basato su un metodo probabilistico che osserva le distribuzioni delle parole.

LDA, come descritto nel paragrafo 2.2.5, è un modello generativo probabilistico che, osservando le parole contenute nel corpus, impara le distribuzioni dei topic nei

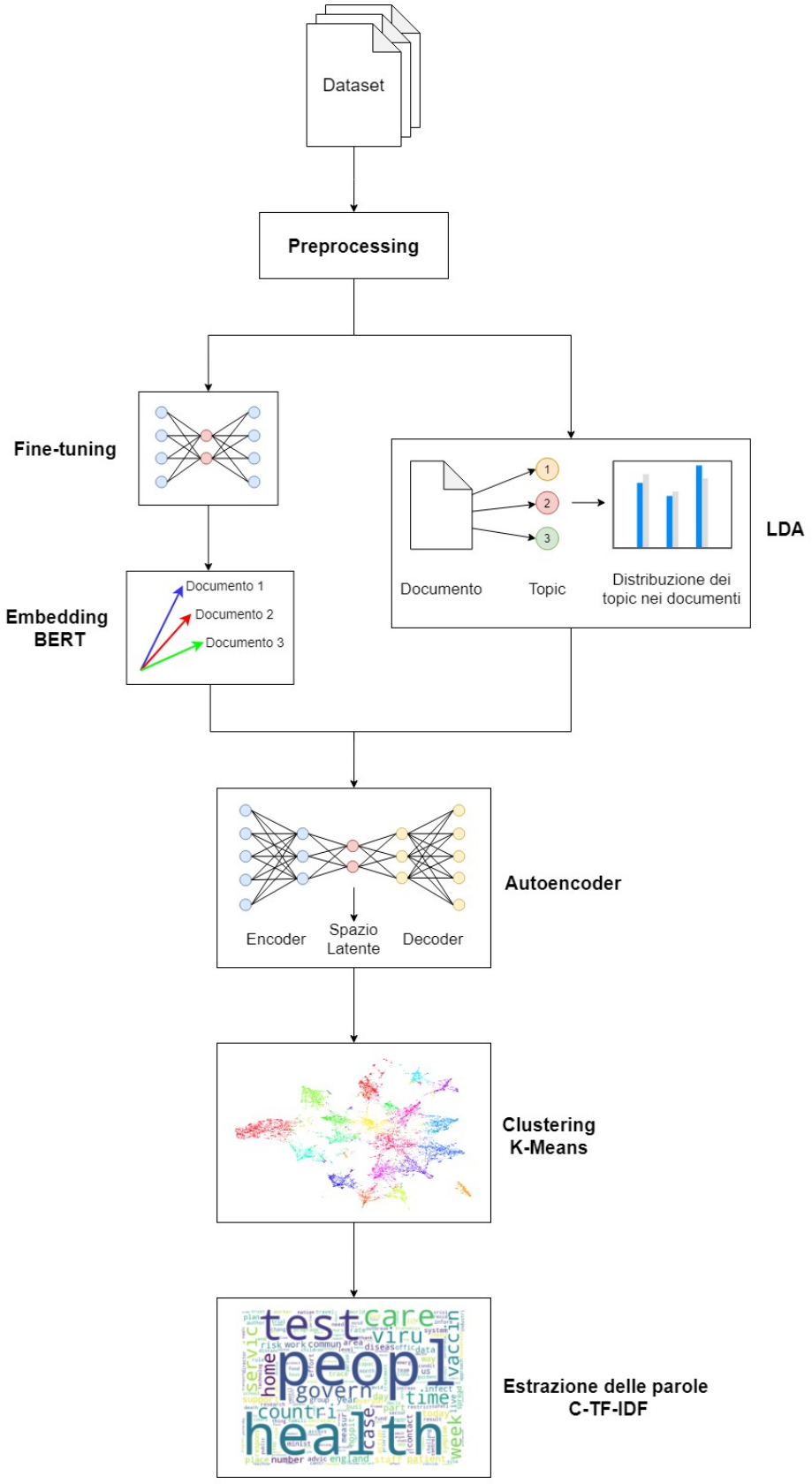

Figura 5.1: Architettura della soluzione

documenti e delle parole nei topic. Essendo quindi un metodo statistico che ricerca le occorrenze delle parole è necessario, prima di utilizzare LDA, effettuare una serie di operazioni di preproccesing sui documenti per rimuovere tutti quei dati che generano rumore e che abbasserebbero la qualità dei topic identificati. In partcolare sono state effettuate le seguenti operazioni di preprocessing sia sul dataset 20Newsgroups che su quello costituito dai comunicati stampa:

1. Si effettua la tokenizzazione dei documenti del corpus, ovvero si dividono le sequenze di caratteri in unità minime di analisi chiamate token. Ogni singolo documento viene quindi trasformato in un array di stringhe contenente tutti i suoi token. Questa operazione è stata effettuata tramite la funzione *word_ tokenize()* della libreria *NLTK* [5].

2. I caratteri di tutti i documenti vengono portati in minuscolo. In questo modo non verranno considerate come diverse le stesse parole che iniziano con una lettera maiuscola o minuscola. Questa operazione è stata effettuata tramite espressioni regolari utilizzando il modulo *re* di Python.

3. Si eliminano tutte le parole che contengono caratteri che non sono lettere. Questa operazione è stata effettuata sfruttando il metodo *isalpha()* delle stringhe in Python.

4. Tramite POS tagging vengono eliminate tutte le parole dei documenti che non sono dei nomi. Questa operazione è stata effettuata utilizzando il POS tagger contenuto nella libreria *NLTK*.

5. Viene applicato lo stemming a tutti i termini dei documenti, riducendoli dalla forma flessa alla forma radice. In questo modo, per esempio, parole in forma singolare o plurale vengono identificate come stessa parola da LDA. Questa operazione è stata effettuata utilizzando la classe *PorterStemmer* contenuta nella libreria *NLTK*.

6. Vengono rimosse dai documenti le stop words, ovvero parole usate frequentemente e che non aggiungono significato alla frase. Esempi di stop words della lingua inglese sono articoli, congiunzioni oppure verbi come "have" o "get". Questa operazione è stata effettuata utilizzando la libreria *stop-words* [11].

Alla fine del processo di preprocessing si ottengono dunque dei documenti in cui sono stati rimossi tutti quegli elementi che avrebbero causato rumore durante l'individuazione dei topic tramite LDA. A questo punto è possibile generare i vettori di embedding concatenando i valori ottenuti tramite l'encoding con BERT e quelli ricavati da LDA.

Per ottenere la rappresentazione tramite BERT viene utilizzato esattamente lo stesso metodo descritto nel paragrafo 4.2.1. Si è sfruttata, infatti, la libreria SenteceTranformer per generare facilmente una rete capace di calcolare l'embedding dei documenti a partire da un modello di Transformer scelto da Huggingface. Si è deciso di testare solamente dei modelli di BERT e di DistilBERT, in quanto si è osservato nei risultati quantitativi della prima soluzione che essi permettono di ottenere una rappresentazione più accurata rispetto a T5. Ai modelli utilizzati è stato inoltre applicato la tecnica di fine-tuning descritta nel paragrafo 4.2.4, in modo da migliorare la qualità dell'embedding prodotto.

Per quanto riguarda LDA si è invece utilizzata la sua implementazione contenuta in Gensim [1], una libreria che racchiude numerose funzioni specifiche per il topic modelling. La classe *ldamodel* presente in Gensim permette infatti di allenare un modello di LDA osservando un corpus e di generare la distribuzione dei topic presenti in nuovi documenti. E' necessario definire i seguenti parametri per utilizzare la classe:

- *corpus*: insieme dei documenti del corpus dove ognuno è rappresentato tramite Bag of Words, ovvero come un vettore contenente per ogni parola del dizionario un id univoco e il numero di volte che essa appare nel documento.

- *num_ topics*: rappresenta il numero di topic che si vuole ottenere. Come descritto nel paragrafo 2.2.5, LDA necessita di sapere a priori la quantità di argomenti che

si vuole identificare nel corpus. Questo valore è molto importante in quanto definisce anche la specificità dei topic che si vogliono ottenere, come verrà descritto meglio successivamente.

- *id2word*: vettore contenente per ogni id univoco delle parole nel dizionario la corrispondente parola. E' utilizzato per poter stampare e visualizzare i termini dopo aver modellato le distribuzioni dei topic.
- *passes*: numero di passaggi che vengono effettuati su tutto il corpus da LDA per modellare le distribuzioni.

Un esempio di allenamento di un modello di LDA tramite la classe offerta da Gensim è il seguente:

```
from gensim.models.ldamodel import LdaModel

model = LdaModel(corpus, num_topics=20, id2word=dictionary, passes=20)
model.print_topics(num_topics=20, num_words=10)
```

Il modello di LDA, dopo essere stato allenato sul corpus, ritornerà quindi due elementi:

1. La distribuzione dei topic nei documenti, ossia una matrice che contiene la percentuale di presenza di ogni argomento in ogni documento.
2. La distribuzione delle parole nei topic, ossia una matrice che contiente la percentuale di presenza di ogni parola in ogni argomento.

In particolare, il vettore che contiene la distribuzione degli argomenti in un determinato documento può essere visto come una sua rappresentazione in uno spazio di dimensione pari al numero di topic e quindi può essere utilizzato come embedding. Ogni testo sarà posizionato dunque nello spazio in una posizione che dipende dalla presenza maggiore o minore dei topic.

Dopo aver ottenuto l'embedding dei documenti tramite BERT e tramite LDA, i due vettori di ogni testo vengono concatenati per ricavare una rappresentazione in uno spazio di dimensione pari alla somma di quelle dei due vettori. Per esempio utilizzando il modello *bert-base*, che restituisce vettori di dimensione 512, e ricercando 20 topic nei documenti si otterrà uno spazio di dimensione uguale a 532. Inoltre viene utilizzato un parametro, chiamato gamma, che serve a definire quanto prendere in considerazione i vettori di LDA rispetto a quelli ottenuti con BERT. Come mostrato nel seguente esempio, i vettori ricavati da LDA vengono moltiplicati per il valore di gamma e successivamente sono concatenati con i vettori di BERT.

```
1 vec_lda = self.vectorize(sentences, token_lists, method='LDA')
2 vec_bert = self.vectorize(sentences, token_lists, method='BERT')
3 vec_ldabert = np.c_[vec_lda * self.gamma, vec_bert]
```

Nella figura 5.2 viene mostrato un esempio di embedding dei documenti ottenuto utilizzato il metodo descritto. Si nota come i vettori siano maggiormente raggruppati rispetto all'embedding visualizzato nella figura 4.1 della soluzione precedente, a dimostrazione che l'aggiunta di LDA contribuisce ad isolare maggiormente i documenti inerenti gli stessi topic.

5.1.2 Riduzione della dimensione e clustering

Come descritto nella soluzione mostrata nel capitolo precedente, un'area densa di vettori nello spazio generato può essere interpretata come una zona dove i corrispondenti documenti è probabile che trattino degli stessi argomenti. Si può dunque utilizzare un algoritmo di clustering per trovare queste aree. I vettori però, trovandosi attualmente in uno spazio di dimensione molto elevata, saranno molto sparsi e per questo sarà difficile identificare delle zone dense di vettori, oltre che particolarmente dispendioso computa-

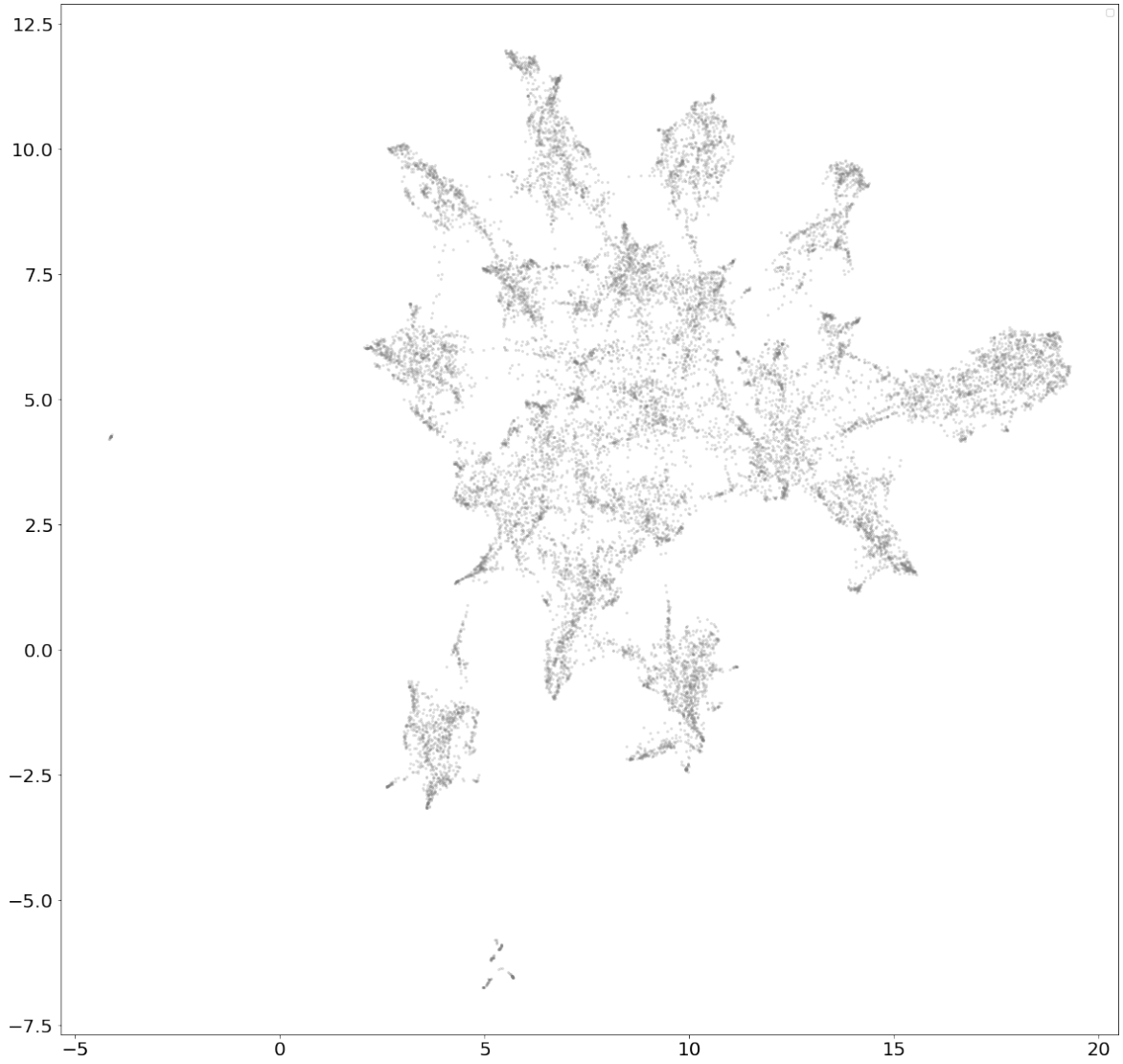

Figura 5.2: Embedding dei documenti del dataset dei comunicati stampa ottenuti tramite la concatenazione dei vettori ricavati con BERT e quelli ricavati con LDA

zionalmente. A differenza della prima soluzione, che utilizzava l'algoritmo UMAP, per ridurre la dimensione dello spazio di embedding si è sfruttato un autoencoder.

Un autoencoder è una rete neurale che ha lo scopo di generare nuovi dati comprimendo l'input in uno spazio di variabili latenti e, successivamente, ricostruendo l'output sulla base delle informazioni acquisite. Gli autoencoder sono quindi costituiti da due parti:

1. L'encoder è la parte della rete che comprime l'input in uno spazio di variabili

latenti di dimensione minore e che può essere rappresentato dalla funzione di codifica $h = f(x)$.

2. Il decoder è la parte che si occupa di ricostruire l'input sulla base delle informazioni presenti nello spazio latente. E' rappresentato dalla funzione di decodifica $r = g(h)$.

L'obbiettivo di un autoencoder è quello di imparare uno spazio latente che presenti caratteristiche utili e tale per cui l'output della rete, calcolato come $r = g(f(x))$, sia il più simile possibile all'input x. Allenando quindi un autoencoder a ricostruire l'input si vuole imparare ad estrarre le informazioni più significative da esso e le si vuole rappresentare in uno spazio di dimensione minore.

Si è utilizzato pertanto un autoencoder per ridurre la dimensione dello spazio generato con l'embedding dei documenti. Esso è stato costruito, utilizzando la libreria *Keras* [4], tramite un encoder ed un decoder con un solo livello fully connected. Come dimensione dello spazio latente è stato scelto il valore 32 mentre la rete è stata allenata per 200 epoche utilizzando l'errore quadratico medio come funzione di loss.

Una volta ottenuto lo spazio di dimensione ridotta si è utilizzato K-Means per estrarre i cluster desiderati. Come valore del numero di raggruppamenti ricercati è stato scelto quello utilizzato precedentemente per trovare le distribuzioni dei topic con LDA. Un esempio di cluster identificati tramite K-Means è mostrato nell'immagine 5.3.

5.1.3 Estrazione delle parole

Una volta che i documenti sono stati raggruppati tramite l'algoritmo di clustering e sono quindi stati classificati in base al topic di appartenenza, è possibile estrarre da essi le parole che descrivono al meglio gli argomenti. Per far ciò si è utilizzata la stessa tecnica mostrata nella prima soluzione, ovvero C-TF-IDF. Come descritto nel paragrafo 4.2.3, essa permette di associare ad ogni parola presente in un dizionario un

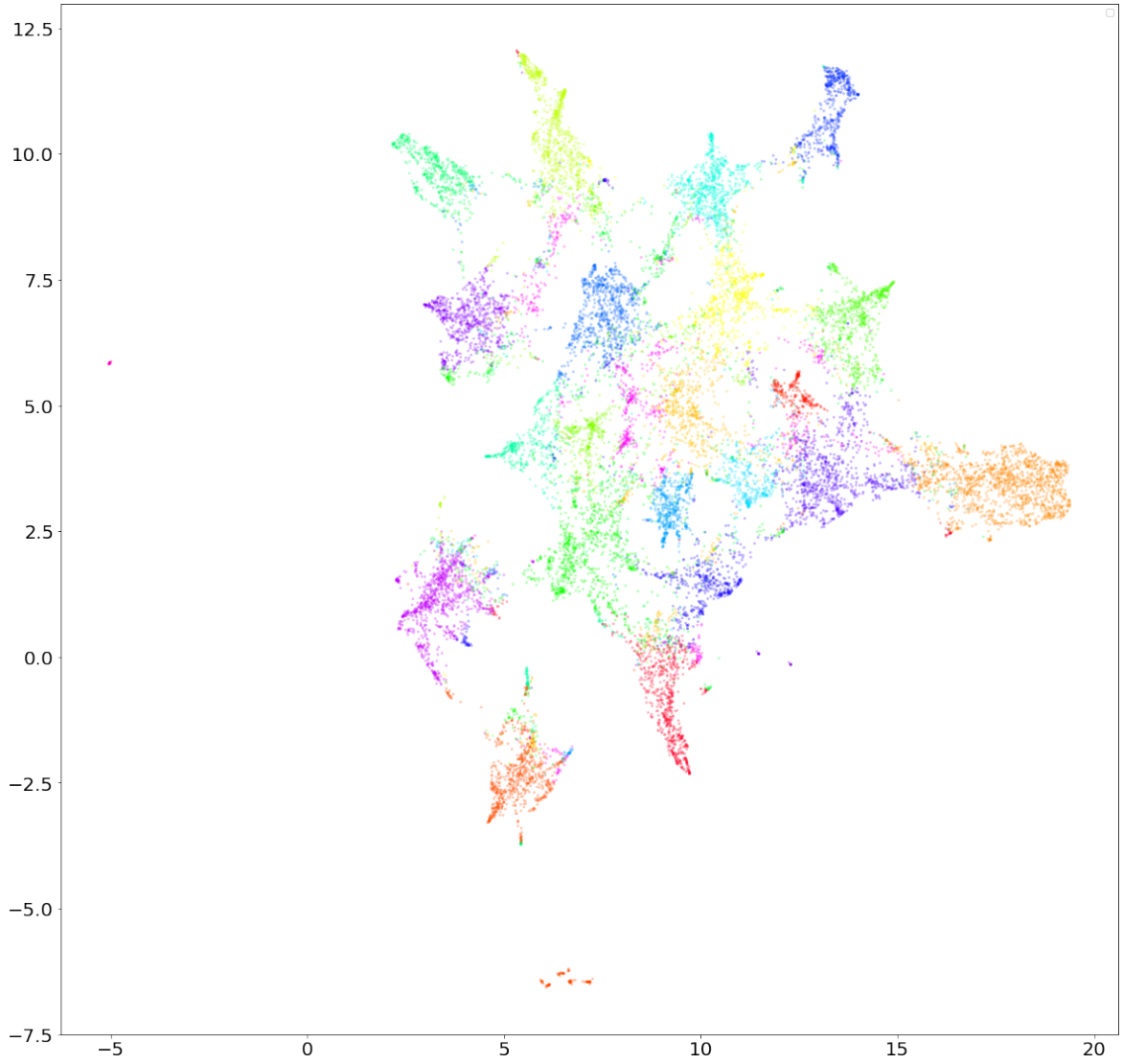

Figura 5.3: Cluster identificati nei documenti del dataset dei comunicati stampa tramite K-Means

punteggio relativo ad ogni topic che è proporzionale al numero di volte che essa appare all'interno dei documenti appartenenti ad un argomento ed inversamente proporzionale al numero di volte che essa appare in tutto il corpus. In questo modo si può ricavare le parole più significative di un topic scegliendo quelle con i punteggi maggiori. Esempi di argomenti estratti dal dataset formato dai comunicati stampa sono mostrati nelle figure 5.4 e 5.5.

5.2 Risultati

Il metodo di estrazione dei topic di questa seconda soluzione è stato testato utilizzando solamente i modelli base di BERT e di DistilBERT, entrambi con e senza il fine-tuning sul dataset da analizzare. I risultati quantitativi, mostrati nella tabella 5.1, sono stati ottenuti sul dataset 20Newsgroups tramite le stesse metriche della soluzione precedente e che sono descritte nel paragrafo 3.4.

Modello	ARI	NMI	MI	Coerenza
bert-base-uncased	0.234	0.389	1.157	0.474
bert-base-uncased con fine-tuning	0.247	0.401	1.181	0.502
distilbert-base-uncased	0.274	0.440	1.278	0.614
distilbert-base-uncased con fine-tuning	0.294	0.449	1.346	0.626
top2vec	0.314	0.466	1.381	0.636

Tabella 5.1: Risultati ottenuti utilizzando K-Means come algoritmo di clustering e C-TF-IDF per l'estrazione delle parole sul dataset 20Newsgroups

Si può innanzitutto osservare come i risultati ottenuti siano nettamente migliori rispetto a quelli della prima soluzione. Questo indica che l'aggiunta delle distribuzioni ottenute con LDA ai vettori di embedding ha permesso ad essi di essere raggruppati in modo più preciso e quindi di essere classificati meglio in base ai topic di appartenenza. Inoltre si può notare, in particolare per il modello di DistilBERT con fine-tuning, come sia i valori di ARI che quelli di coerenza si avvicinino molto a quelli ottenuti con top2vec, algoritmo che al momento permette di generare i risultati migliori per il task del topic modelling.

Nelle figure 5.4 e 5.5 sono invece mostrare le parole che rappresentano i topic identificati nel dataset dei comunicati stampa. Si nota come tutti gli argomenti sono contraddistinti bene e che le parole sono specifiche del tema che rappresentano. E' inoltre interessante osservare come il valore del numero di topic da ricercare, scelto

per poter utilizzare LDA e K-Means, condizioni la specificità degli argomenti trovati. Scegliendo infatti un valore alto per il numero di argomenti si otterranno dei topic riguardanti temi maggiormente specifici mentre, al contrario, utilizzando un valore basso si individueranno degli argomenti più generali e che ingloberanno diversi temi minori. Ad esempio, si è osservato che ricercando un numero di argomenti pari a 30, i topic riguardanti il COVID-19, i vaccini e la sanità in Inghilterra vengono identificati come distinti. Utilizzando invece un valore pari a 10, tutti e tre questi temi verranno considerati come un unico argomento. Avere perciò la possibilità, data dall'algoritmo, di modificare a piacere il numero di argomenti da ricercare può essere particolarmente utile.

Un altro elemento interessante da osservare è il parametro gamma, utilizzato per definire quanto prendere in considerazione i vettori di LDA rispetto a quelli ottenuti con BERT. Si è notato che all'aumentare di questo parametro e visualizzando gli embedding riducendo la loro dimensione tramite t-SNE, i documenti appaiono maggiormente raggruppati in zone isolate. Questo è dovuto proprio al fatto che si prende di più in considerazione le distribuzioni ottenute con LDA e quindi saranno posizionati vicini i documenti con tali distribuzioni simili. Tuttavia si è osservato che un valore troppo alto di gamma comporta risultati bassi per quanto riguarda la coerenza tra le parole dei topic, in quanto l'inferenza basata sulla distribuzione statistica delle parole si è mostrata non essere una buona soluzione per ricavare gli argomenti da dei documenti brevi come lo sono i comunicati stampa. Riducendo invece il valore di gamma si ottengono risultati maggiormente simili a quelli della prima soluzione, in quanto l'embedding viene ottenuto considerando maggiormente i vettori ricavati con BERT. Per riuscire a trovare il valore ideale di gamma si sono dunque testati vari valori compresi tra 1 e 20, scegliendo alla fine quello che massimizzava la coerenza tra le parole ottenuta.

In conclusione si può dire che questa seconda soluzione rappresenta decisamente un grosso passo in avanti rispetto alla prima mostrata. La classificazione dei documenti ottenuta tramite i cluster è maggiormente simile a quella reale di 20Newsgorups, come

dimostrano i valori di ARI, e le parole che descrivono i topic indicano degli argomenti ben chiari e riconoscibili.

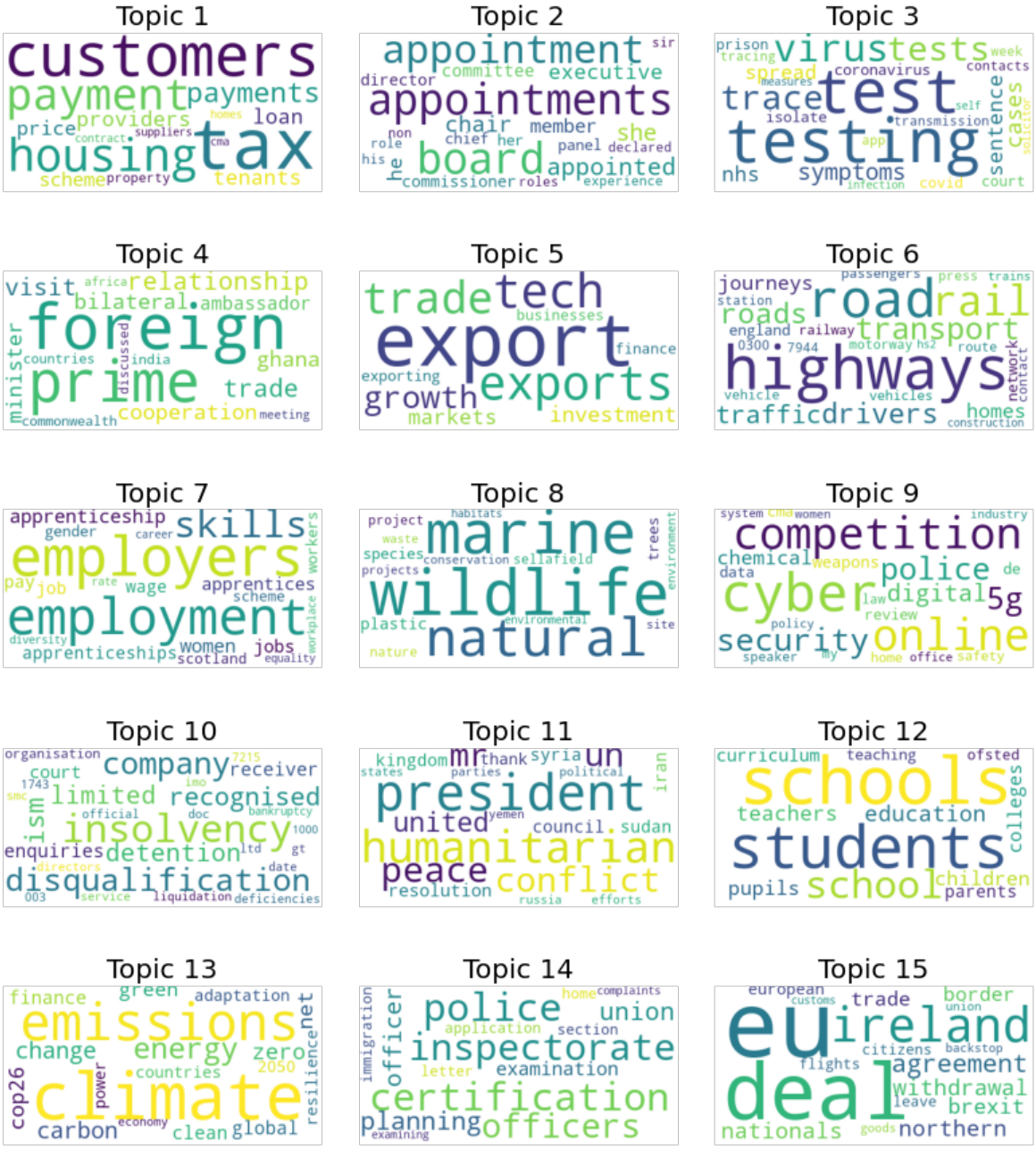
Topic 1
customers
payment
payments
price
providers
loan
housing
tax
scheme
property
tenants
Topic 2
appointment
committee
executive
director
appointments
chair
member
she
board
appointed
commissioner
roles
experience
Topic 3
virus
tests
tracing
spread
coronavirus
contacts
trace
test
isolate
transmission
testing
nhs
symptoms
infection
covid
court
sentence
cases
Topic 4
visit
africa
relationship
bilateral
ambassador
foreign
minister
countries
india
discussed
ghana
prime
trade
commonwealth
cooperation
meeting
Topic 5
trade
tech
businesses
export
finance
exporting
exports
growth
markets
investment
Topic 6
journeys
passengers
press
trains
station
roads
road
rail
england
railway
transport
motorway
hs2
route
highways
vehicle
vehicles
network
contact
traffic
drivers
homes
construction
Topic 7
apprenticeship
skills
gender
career
employers
pay
job
rate
wage
apprentices
scheme
employment
diversity
apprenticeships
women
scotland
jobs
equality
workplace
Topic 8
project
habitats
marine
waste
species
conservation
sellafield
trees
environment
projects
wildlife
plastic
environmental
site
nature
natural
Topic 9
system
cma
women
industry
competition
chemical
weapons
police
data
de
cyber
law
digital
5g
review
security
online
policy
speaker
my
home
office
safety
Topic 10
organisation
company
7215
court
receiver
1743
imo
limited
recognised
official
doc
bankruptcy
insolvency
1000
enquiries
detention
ltd
gt
directors
date
disqualification
003
service
liquidation
deficiencies
Topic 11
kingdom
mr
thank
syria
un
states
parties
political
iran
president
united
yemen
council
sudan
humanitarian
peace
conflict
resolution
russia
efforts
Topic 12
curriculum
teaching
ofsted
schools
teachers
education
colleges
students
pupils
school
children
parents
Topic 13
finance
green
adaptation
emissions
net
change
energy
zero
countries
2050
cop26
power
climate
resilience
carbon
economy
clean
global
Topic 14
home
complaints
police
union
immigration
officer
application
section
inspectorate
letter
examination
certification
planning
officers
examining
Topic 15
european
customs
trade
border
union
eu
ireland
citizens
backstop
flights
agreement
deal
withdrawal
leave
brexit
nationals
goods
northern

Figura 5.4

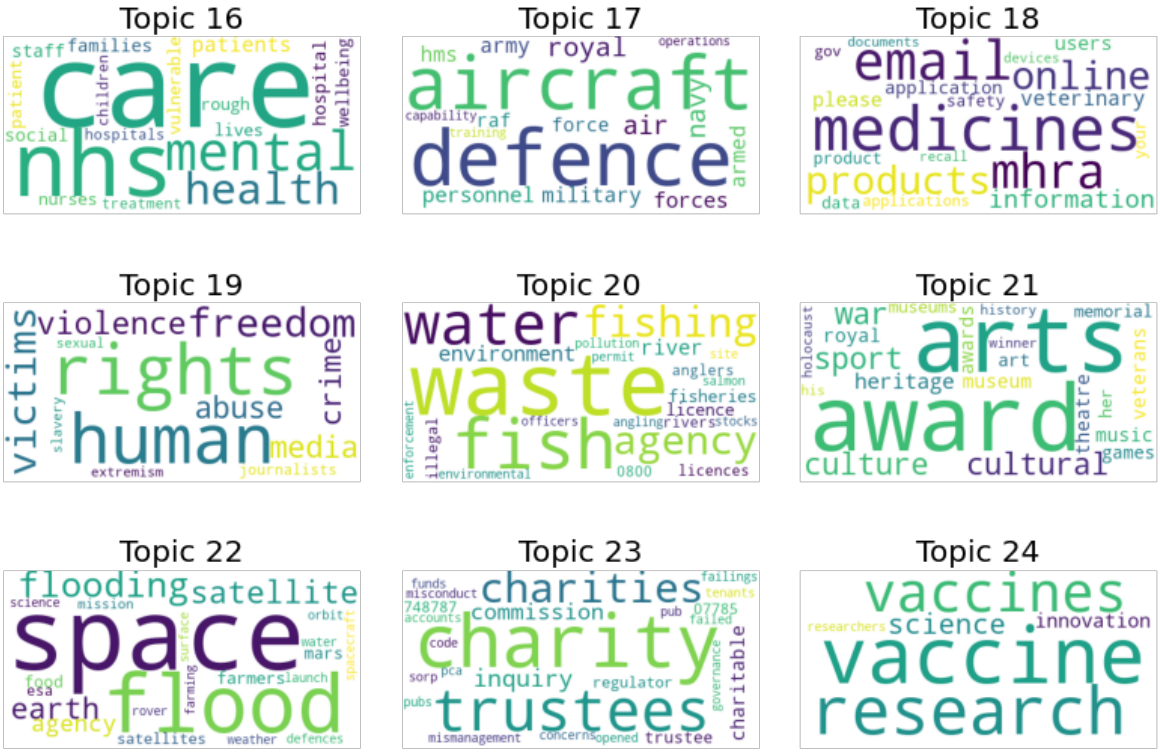
Topic 16
care
nhs
mental
health
staff
families
patients
children
vulnerable
hospital
wellbeing
patient
rough
social
hospitals
lives
nurses
treatment
Topic 17
aircraft
defence
hms
army
royal
operations
navy
capability
raf
force
air
training
armed
personnel
military
forces
Topic 18
email
online
medicines
mhra
products
gov
documents
users
devices
application
safety
veterinary
please
product
recall
your
data
applications
information
Topic 19
victims
violence
freedom
rights
crime
sexual
abuse
slavery
human
media
extremism
journalists
Topic 20
water
fishing
environment
pollution
river
permit
site
waste
anglers
salmon
fisheries
licence
officers
anglingrivers
stocks
enforcement
fish
agency
illegal
environmental
0800
licences
Topic 21
arts
award
war
museums
history
memorial
royal
sport
awards
winner
art
holocaust
heritage
museum
his
theatre
her
veterans
culture
cultural
music
games
Topic 22
flooding
satellite
space
flood
science
mission
orbit
water
mars
surface
spacecraft
food
esa
farming
farmers
launch
earth
rover
agency
satellites
weather
defences
Topic 23
charities
funds
misconduct
failings
tenants
748787
commission
pub
07785
accounts
failed
charity
code
governance
sorp
pca
inquiry
regulator
charitable
pubs
trustees
mismanagement
concerns
opened
trustee
Topic 24
vaccines
researchers
science
innovation
vaccine
research

Figura 5.5

Capitolo 6

Conclusioni

In questo lavoro di tesi, dopo avere osservato l'evoluzione del topic modelling negli ultimi anni, sono state dunque presentate due soluzioni differenti a tale problema applicato ad un corpus costituito da comunicati stampa:

1. Una prima soluzione incentrata esclusivamente sull'embedding dei documenti utilizzando una rete Transformer

2. Una seconda soluzione incentrata sull'embedding dei documenti ottenuto dalla concatenazione dei vettori calcolati con una rete Transformer e quelli derivanti dalle distribuzioni dei topic ricavate con LDA

Per entrambe le soluzioni è stata poi utilizzata una tecnica di riduzione della dimensione per ottenere uno spazio dove i vettori siano meno sparsi. Nel primo approccio si è infatti usato l'algoritmo UMAP mentre nel secondo si è sfruttato un autoencoder. Infine i documenti sono stati classificati in base al topic di appartenenza, in entrambi i casi utilizzando K-Means, e sono state estratte le parole più significative di ogni argomento tramite C-TF-IDF.

La principale innovazione mostrata in questo lavoro è stata comunque quella di aver testato i vantaggi offerti dalle reti Transformer al fine di realizzare un algoritmo non

supervisionato di topic modelling. La prima soluzione è stata pensata partendo dall'algoritmo di top2vec, utilizzando un modello di BERT o di T5 per ottenere l'embedding dei documenti. Tuttavia, testando il metodo sul dataset 20Newsgroups tramite le metriche descritte nel paragrafo 3.4, non si sono ottenuti risultati soddisfacenti, sopratutto se paragonati con quelli di top2vec. Questo ha dimostrato che i vettori di embedding generati esclusivamente utilizzando delle reti Transformer non sono particolarmente rappresentativi dei contesti semantici dei documenti. Si è osservato però che l'utilizzo di una tecnica di fine-tuning per continuare ad allenare il modello di Transformer ha permesso di ottenere risultati migliori rispetto al modello di base.

Con la seconda soluzione mostrata si è cercato dunque di ottenere un embedding dei documenti maggiormente preciso, usando anche le distribuzioni dei topic ottenute con LDA. I risultati quantitativi dimostrano che questo secondo approccio è nettamente più efficiente rispetto al primo. Si è mostrato quindi che analizzando un corpus di testi brevi, così come lo sono generalmente i comunicati stampa e i documenti che costituiscono 20Newsgroups, un utilizzo esclusivo di LDA o delle reti Transformer non garantisce di ottenere topic precisi, ma una loro unione permette di sfruttare i vantaggi di entrambe le tecniche per identificare gli argomenti in maniera migliore. Si è inoltre osservato come il numero di topic che vengono ricercati dall'algoritmo determina la loro specificità, risultando quindi un parametro molto utile.

Tuttavia, come è stato mostrato tramite il confronto con l'algoritmo top2vec, entrambe le soluzioni non permettono di ottenere risultati migliori rispetto ad alcune tecniche già esistenti. Gli algoritmi presentati, però, possono rappresentare un punto di partenza per possibili sviluppi futuri. Innanzitutto si potrebbe testare entrambi i metodi sostituendo i vettori dei documenti generati tramite le reti BERT e T5 con quelli ricavati da altre tecniche di embedding dei testi, verificando quindi se esse permettono di ottenere rappresentazioni migliori dei contesti semantici. Per esempio si potrebbero osservare i risultati che si otterrebbero utilizzando altre reti Transformer, come GPT2 o XLM, oppure sfruttando altre tecnologie per l'embedding dei documenti, como EL-

Mo o XLNet. Inoltre, essendo il settore del NLP in costante sviluppo, è possibile che negli anni a venire vengano ideati nuovi algoritmi o tecnologie di deep learning che permetteranno di ottenere rappresentazioni vettoriali migliori dei documenti. Un altro cambiamento che si potrebbe apportare è quello di allenare ulteriormente il modello di Transformer tramite fine-tuning su un dataset esterno costituito da articoli di giornale o comunicati stampa, in modo da specializzarlo maggiormente. Oppure, invece che utilizzare un Transformer scelto da Huggingface, si potrebbe provare ad applicare il fine-tuning ad un modello pre-allenato tramite la libreria SentenceTransformer, in modo simile a quanto descritto nel paragrafo 3.2.5.

In conclusione, si può dire che questo lavoro di tesi ha posto diversi interrogativi sull'utilizzo delle reti Transformer per risolvere il problema dell'estrazione degli argomenti trattati in un corpus di documenti. I risultati ottenuti sono stati più che soddisfacenti, sopratutto per quanto riguarda la seconda soluzione descritta, ma rappresentano solamente un punto di partenza per quelli che potranno essere studi futuri su un task utile e interessante come quello del topic modelling.

Bibliografia

[1] gensim. https://radimrehurek.com/gensim.

[2] Hdbscan. https://github.com/scikit-learn-contrib/hdbscan.

[3] Huggingfface. https://huggingface.co/.

[4] Keras. https://keras.io/.

[5] Natural language toolkit. https://www.nltk.org/.

[6] Numpy. https://numpy.org/.

[7] Pandas. https://pandas.pydata.org/.

[8] Scikit-learn. https://scikit-learn.org/.

[9] Scrapy. https://scrapy.org/.

[10] Simple transformer. https://simpletransformers.ai/.

[11] stop-words. https://github.com/Alir3z4/python-stop-words.

[12] Umap. https://github.com/lmcinnes/umap.

[13] Dimo Angelov. Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*, 2020.

[14] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155, 2003.

[15] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[16] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[17] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

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

[19] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[20] 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.

[21] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural computation*, 23(9):2421–2456, 2011.

[22] John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

[23] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[24] Michael J. McGill Gerard Salton. *Introduction to Modem Information Retrieval*. McGraw-Hill, 1983.

[25] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[26] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[27] Pooja Kherwa and Poonam Bansal. Topic modeling: a comprehensive review. *EAI Endorsed transactions on scalable information systems*, 7(24), 2020.

[28] Ubiquitous Knowledge Processing Lab. Sentence transformer. `https://www.sbert.net`.

[29] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

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

[31] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[32] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

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

[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

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

[37] Michael Röder, Andreas Both, and Alexander Hinneburg. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408, 2015.

[38] Steve Shao. Contextual topic identification. `https://blog.insightdatascience.com/contextual-topic-identification-4291d256a032`.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.