
Università degli Studi del Sannio

DIPARTIMENTO DI ECONOMIA
MANAGEMENT E METODI
QUANTITATIVI

Corso di Laurea Magistrale in
Scienze Statistiche ed Attuariali



Classificazione semantica di documenti
tramite metodi di Text Mining

RELATORE:

Ch.mo Prof. **Antonio Lucadamo**

CANDIDATO:

Francesco Cuozzo

Matr.: 709000613

CORRELATORE:

Ch.mo Prof. **Pietro Amenta**

ANNO ACCADEMICO 2024/2025

Indice

| | |
|--|-----------|
| Introduzione | 3 |
| 1 Contesto generale | 5 |
| 1.1 Inizi del Text Mining | 5 |
| 1.2 Tipologia dei dati | 7 |
| 1.3 Due differenti approcci al text mining | 8 |
| 1.4 Selezione dei dati | 12 |
| 1.5 Formato dei documenti | 13 |
| 1.6 Normalizzazione dei testi | 13 |
| 2 Natural Language Processing | 14 |
| 2.1 Modelli statistici per l’NLP | 15 |
| 2.2 Esempi di applicazione | 15 |
| 2.3 CRF | 17 |
| 2.4 N-grams: | 18 |
| 2.4.1 Limiti degli N-grams | 19 |
| 2.4.2 Probabilità di una frase | 20 |
| 3 Pre-processing | 21 |
| 3.1 Fasi principali del preprocessing: | 21 |
| 3.2 Pulizia del testo : rimozione della punteggiatura, dei simboli e trattamento delle lettere maiuscole | 22 |
| 3.3 Rimozione delle stopwords | 23 |
| 3.4 Tokenizzazione | 23 |
| 3.5 Embedding | 25 |
| 3.6 La distanza Euclidea | 29 |
| 3.7 Word2vec | 30 |
| 3.8 Doc2vec | 33 |
| 3.9 Skip-gram | 34 |
| 3.10 CBOW | 35 |
| 3.11 Distributed Memory | 36 |
| 3.12 Distributed Bag-of-Words | 37 |

| | | |
|----------|--|------------|
| 3.13 | Stemming | 38 |
| 3.14 | Limiti dello stemming | 40 |
| 3.15 | Approccio bag-of-words | 41 |
| 3.16 | Matrice <i>document-term</i> | 43 |
| 3.17 | Riduzione della dimensionalità | 44 |
| 3.17.1 | SVD | 45 |
| 3.18 | Named entity recognition | 47 |
| 3.18.1 | Esempio di Named Entity Recognition | 48 |
| 3.19 | Part of speech (POS) tagging | 49 |
| 3.20 | Parsing | 50 |
| 4 | Costruzione del dataset | 51 |
| 4.1 | Classificazione | 52 |
| 4.2 | Information Gain | 54 |
| 4.3 | Classificazione Supervisionata | 57 |
| 4.4 | Decision Rules | 59 |
| 4.5 | Similarità tra documenti | 61 |
| 4.6 | K-nearest neighbors | 62 |
| 4.7 | Naive Bayes | 62 |
| 4.7.1 | Notazioni | 62 |
| 4.8 | Valutazione dei classificatori | 65 |
| 4.9 | Esempio applicativo: AI vs Human Content Detection | 68 |
| 4.9.1 | Zero-shot e Few-shot Learning | 68 |
| 4.9.2 | Analisi delle performance dei modelli | 76 |
| 5 | Descrizione del Dataset | 78 |
| 5.1 | Analisi sintattica | 80 |
| 5.2 | Bag of words | 85 |
| 5.3 | Text Clustering | 88 |
| 5.4 | Rappresentazione generale | 91 |
| 5.5 | Analisi delle emozioni primarie | 92 |
| 5.6 | Criteri di valutazione | 94 |
| 5.7 | Modello di regressione logistica | 96 |
| 5.8 | Analisi discriminante | 96 |
| 5.9 | Gradient boosting | 101 |
| 5.10 | Scelta degli iperparametri | 102 |
| 5.11 | Risultati | 104 |
| 5.12 | Confronto dei risultati | 110 |
| 5.13 | Conclusioni e approfondimenti finali | 112 |
| | Bibliografia | 119 |

Introduzione

Il text mining, noto anche come text data mining, è il processo di conversione di testo non strutturato in un formato strutturato per identificare pattern significativi. Dunque ci riferiamo ad un'ampia classe di procedure statistiche, linguistiche e informatiche volte ad analizzare grandi insiemi di testi al fine di ricavarne informazioni. Il concetto fondamentale è quello di riuscire ad estrarre pochissime informazioni rilevanti da una vasta quantità di dati a disposizione senza doverli elaborare tutti. Spesso si fa riferimento al text mining come a una branca del data mining, ovvero l'estrazione di informazioni da grandi insiemi di dati. Solo che nel data mining ricerchiamo pattern nei dati mentre nel text mining ricerchiamo pattern nel testo. Tuttavia il primo può essere meglio caratterizzato come l'estrazione dai dati di informazioni precedentemente sconosciute e potenzialmente utili e dunque l'informazione non può essere estratta senza ricorrere a tecniche automatiche di data mining.

Capitolo 1

Contesto generale

1.1 Inizi del Text Mining

Sebbene i primi tentativi di analisi automatica dei testi risalgano agli anni '60, si può ipotizzare che il text mining prende forma nel contesto del data mining soltanto alla fine degli anni '90 come già suggerito da [Dörre et al., 1999] nell'articolo “*Text Mining: Finding Nuggets in Mountains of Textual Data*”, il text mining si configura come l'estrazione di *piccole pepite informative* da grandi volumi di dati testuali (senza dover leggere tutto). Il text mining dunque è da considerare come un'estensione naturale del data mining con il quale condivide buona parte delle tecniche e si può ricondurre a un problema di classificazione supervisionata dei testi dove le esplicative sono estratte dai testi stessi. Il problema principale è che le informazioni non sono formulate in modo strutturato e quindi non è possibile effettuare elaborazioni automatiche. Il text mining è nato dal contesto dei social network ed è legato alla sentiment analysis.

Un fattore che ha determinato lo sviluppo del text mining è l'enorme passo in avanti che è stato fatto nel settore linguistico, cioè di come è cambiata la visione della logica della “lingua”. Si è passati da una logica di tipo “linguistico” ad una “lessico-testuale”. La forma più naturale per la memorizzazione delle informazioni è il testo e in effetti la maggior parte della produzione di informazioni di una società proviene da chat, e-mail, forum e quindi testo scritto in linguaggio naturale, solo una piccola percentuale è espressa in dati numerici.

Secondo quanto riportato in [Ted, 2017], *per fare in modo che una frase sia "digeribile" da un algoritmo di stima bisogna spogiarla da tutte le parole e i simboli che ne aumentano la complessità e fare in modo che restino solo le parole chiave che da sole sono in grado di contenere gran parte dell'informazione.*

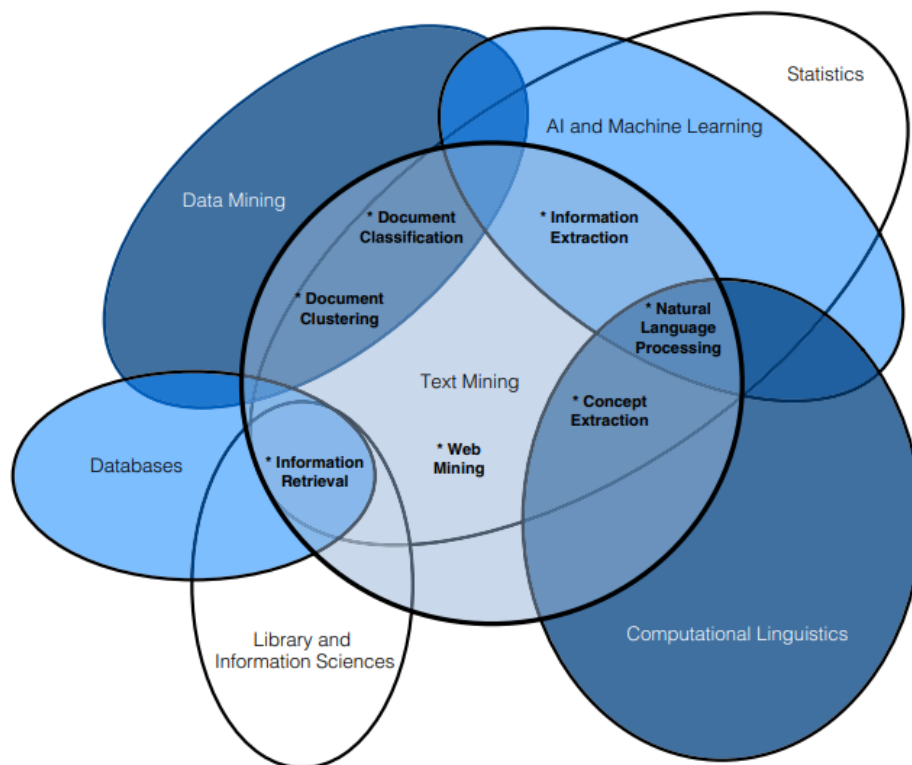


Figura 1.1: [Research, 2020] Il Text Mining come interdisciplina, al crocevia tra Data Mining, NLP, Machine Learning e Linguistica Computazionale

1.2 Tipologia dei dati

Il testo è uno dei tipi di dati più comuni all'interno dei database.

A seconda del database questi dati possono essere organizzati come:

- **Dati strutturati:** sono standardizzati in formato tabellare con numerose righe e colonne, il che facilita la loro elaborazione per gli algoritmi di machine learning. Possono includere input come indirizzi, nomi e numeri di telefono.
- **Dati non strutturati:** non hanno un formato di dati predefinito e possono riguardare testi provenienti da fonti come social media, recensioni di prodotti, oppure contenuti multimediali come video e file audio.
- **Dati semistrutturati:** come suggerisce il termine, sono un misto tra dati strutturati e non strutturati. Includono ad esempio file XML e HTML: hanno un minimo di organizzazione ma non una struttura sufficiente per soddisfare i requisiti dei dati strutturati.

1.3 Due differenti approcci al text mining

In generale l'estrazione di informazioni da testo può avvenire principalmente in due modi.

- **Approccio rule-based**

E' l'approccio più semplice ed intuitivo cioè basato su regole predefinite esplicite scritte manualmente (pattern, espressioni regolari, dizionari, alberi sintattici) ed è ancora ampiamente usato in diversi ambiti dove è necessario controllo, precisione o dove i dati sono strutturati in modo prevedibile. Se si vuole classificare un testo all'interno di due o più classi, si stabilisce un set di regole che mettano in corrispondenza alcune parole del testo (pattern) con una data classe in cui il testo deve essere classificato. Se ad esempio siamo interessati ad analizzare le recensioni dei clienti di un servizio online per determinare se esprimono un'opinione positiva, negativa o neutra, l'operazione di text mining si può ricondurre a un problema di classificazione del testo all'interno delle tre classi corrispondenti ai possibili sentimenti espressi dagli utenti. O ancora possiamo essere interessati a determinare l'esito di una sentenza a partire dal testo della motivazione. In questo caso avremo un problema di classificazione del testo all'interno delle classi corrispondenti ai possibili esiti processuali come accoglimento del ricorso, rigetto, oppure parziale accoglimento. Se si adotta l'approccio rule-based, il set di regole dovrà mettere in corrispondenza una data stringa di testo con la classe corrispondente. E' facile intuire che la definizione delle regole è lo step fondamentale che può rivelarsi molto dispendioso in termini di tempo. Se effettuato correttamente può garantire una classificazione accurata dei testi.

Vantaggi: L'approccio rule-base garantisce una buona semplicità interpretativa, in quanto le regole possono essere comprese e modificate con facilità anche da parte dell'utente. Inoltre, è possibile integrare questi sistemi con strumenti di supporto, come ad esempio dizionari elettronici, che consentono di ampliare le capacità di riconoscimento dei termini. Un ulteriore aspetto positivo è rappresentato dall'alta precisione raggiungibile, a condizione che le regole siano state definite in modo corretto e sufficientemente dettagliato.

Svantaggi: I sistemi rule-based risultano infatti sensibili alla qualità dei testi analizzati: in presenza di rumore, abbreviazioni o linguaggio non standardizzato, le regole possono facilmente fallire. Essi sono inoltre strettamente dipendenti dal contesto di sviluppo, il che ne riduce la portabilità in domini differenti. Infine, la definizione di un insieme completo di regole richiede tempi lunghi e un notevole sforzo manuale, rendendo l'approccio poco scalabile in scenari complessi o con grandi quantità di dati.

- **Approccio statistico**

Rappresenta lo standard per l'analisi dei testi online e nell'ambito dei social media. Si rifà completamente alle tecniche di data mining ed è anche chiamato approccio di machine learning. E' richiesta una prima fase di pre-processing dove il testo viene modificato e trasformato in variabili statistiche, successivamente vi è una fase di classificazione dei testi.

Vantaggi:

L'approccio statistico risulta più vantaggioso data la sua maggiore capacità di adattamento infatti la classificazione è più robusta alla presenza di sinonimi, abbreviazioni e errori ortografici. Questo rende tali metodi particolarmente flessibili e adattabili a domini differenti. Un ulteriore vantaggio è rappresentato dalla possibilità di trattare grandi volumi di dati in tempi ridotti, sfruttando algoritmi efficienti e tecniche di calcolo parallelo. Inoltre, i modelli statistici sono in grado di generalizzare meglio rispetto ai sistemi rule-based, cogliendo regolarità linguistiche anche in presenza di testi rumorosi o non standardizzati. Inoltre dal momento che un set predefinito di regole è in grado di classificare solo i testi inerenti a un determinato ambito, se cambia il contesto dovranno cambiare per forza anche le regole, le procedure di data mining alla base dell'approccio statistico possono essere applicate a ogni tipo di testo richiedendo soltanto piccole modifiche.

Svantaggi:

Accanto a questi punti di forza, si riscontrano tuttavia alcuni limiti: la svantaggiosità di questo approccio risiede nel fatto che la classificazione deve essere di tipo supervisionato, ovvero deve essere presente un insieme di testi dei quali si conoscono già le informazioni da estrarre. I metodi statistici richiedono la disponibilità di dataset di addestramento ampi e ben etichettati, il cui reperimento non è sempre immediato. Inoltre, l'interpretabilità dei modelli è spesso ridotta rispetto agli approcci rule-based: i risultati dipendono dalle stime probabilistiche o dai parametri appresi dagli algoritmi, il che rende difficile comprendere a fondo il motivo di alcune decisioni. Infine, l'addestramento di modelli complessi, come reti neurali profonde, può essere costoso in termini di risorse computazionali e richiedere hardware dedicato, limitandone l'applicabilità in contesti con vincoli di tempo o di calcolo. Detto in altre e più semplici parole di questo insieme di testi si deve conoscere la classificazione a priori se si vuole che le procedure statistiche siano in grado di classificare ulteriori testi.

In realtà i due approcci sono più simili di quanto si possa pensare. Infatti quello che accade all'interno del modello statistico di classificazione si può immaginare come la creazione automatica di regole che il modello stesso stabilisce in base ad algoritmi di stima. Le unità statistiche, cioè i testi, verranno poi classificate in base a quelle regole. Ad esempio, se prendiamo in considerazione l'albero di classificazione, nella fase di stima stabilisce un set di regole, cioè delle partizioni dello spazio delle variabili statistiche, scegliendole tra tutte le regole possibili. Il criterio di decisione di queste regole è automatico e si basa sulla minimizzazione della funzione loss.

Tuttavia il loro funzionamento è identico a quello delle regole stabilite manualmente nell'approccio rule-based.

1.4 Selezione dei dati

Questa fase è molto importante perché se viene eseguita in modo approssimativo e non relativo alla descrizione del problema si rischia di ottenere risultati non esatti. Con l'aumento dei documenti digitali è spesso necessario selezionare un sottoinsieme dei dati che risultano più pertinenti per l'obiettivo dell'analisi dal momento che è diventato impossibile analizzare l'intero corpus.

Questa selezione può avvenire secondo differenti criteri:

- **Temporal** (ad es. solo documenti pubblicati in un certo intervallo di tempo)
- **Categoriali** (ad es. solo notizie politiche o relative alla salute)
- **Linguistici** (escludere documenti troppo brevi o contenenti solo numeri)
- **Qualitativi** (rimuovere duplicati e spam)

1.5 Formato dei documenti

L'informazione contenuta nella formattazione e nella struttura del testo potrebbe risultare importante. Ad esempio la stringa di caratteri che compone il titolo sarà molto importante per individuare l'argomento trattato dal testo, così come il campo oggetto in una e-mail. Fra i vari formati disponibili, si è diffuso l'utilizzo dell'XML come standard per il text mining. L'XML si basa sulla divisione del testo in campi, che possono dare indicazioni sul contenuto (ad esempio i campi titolo, autore, sommario, ecc.) oppure sul formato del carattere (stile, dimensione, ecc.). L'XML permette di conservare la struttura del testo, rimuovendo l'informazione relativa alla formattazione, che spesso risulta inutile ai fini di un'analisi di text mining.

Ma l'uso dell'XML va oltre il semplice formato di input. Solitamente, i software di Natural Language Processing forniscono informazioni in output aggiungendo nel testo delle tags XML. Ad esempio, un'applicazione per l'analisi logica potrà indicare soggetto e complemento oggetto tramite opportune tags

1.6 Normalizzazione dei testi

Il text mining dispone di un insieme di documenti testuali liberi. I testi sono un formato di dati estremamente complesso e le frasi non sono semplici insiemi di parole ma tra le parole stesse sussistono delle relazioni ben precise che a volte ne modificano il significato. Inoltre in una frase sono presenti anche punteggiatura, simboli e numeri. La complicazione che il text mining aggiunge al data mining è nel primo passaggio: fare in modo che i testi diventino variabili statistiche. Questo dipende principalmente da due fattori cioè dalle caratteristiche del testo e dal contesto in cui il testo è stato scritto.

Capitolo 2

Natural Language Processing

Natural Language Processing (NLP) è un campo interdisciplinare dell'informatica, dell'intelligenza artificiale e della linguistica computazionale che si occupa dell'interazione tra linguaggio umano e computer.

L'obiettivo dell'NLP è permettere alle macchine di comprendere, interpretare, generare e analizzare testi o discorsi in linguaggio naturale (es. italiano, inglese, cinese, ecc.). L'NLP è basato su modelli statistici e probabilistici utilizzati per analizzare frequenze di parole, co-occorrenze.

Il text mining è strettamente legato agli ambiti di studio della linguistica computazionale e del natural language processing.

Un modello statistico del linguaggio consiste nell'assegnazione di una misura di probabilità sulle sequenze di parole (o simboli) che compongono una comunicazione orale o scritta in linguaggio naturale. Supponiamo che il nostro linguaggio sia composto dai simboli (ad esempio, parole e punteggiatura) contenuti nell'insieme Θ .

Se consideriamo una parte di comunicazione, ad esempio una frase, essa sarà composta da una sequenza $\{x_1, x_2, \dots, x_n\}$ di elementi di Θ .

Dunque vogliamo assegnare una probabilità ad ogni possibile sequenza.

Può essere quindi utile una brevissima introduzione sui modelli statistici di linguaggio più diffusi (tali modelli sono alla base del riconoscimento vocale e della traduzione automatica dei testi).

2.1 Modelli statistici per l’NLP

1. Modelli probabilistici

- N-gram : per stimare la probabilità di sequenze di parole
- Bayesian model : per la classificazione dei testi

2. Machine Learning

- Algoritmi supervisionati e non supervisionati
(ad es: Regressione logistica , SVM , clustering)
- Tecniche di selezione delle features (ad es: IDF , Information Gain)

Modelli generativi e deep learning Reti neurali e Long Short Term Memory che sono alla base dell’addestramento di modelli come BERT e GPT dove vengono utilizzati enormi quantità di dati testuali per il loro training.

2.2 Esempi di applicazione

- **Analisi del sentiment:** classificare recensioni/testi in positivi, negativi, neutri utilizzando ad esempio Naive Bayes o la logistica. Ad esempio durante la campagna presidenziale USA di Donald Trump, milioni di tweet contenenti hashtag come #MAGA o #NeverTrump sono stati analizzati con modelli di sentiment (Naive Bayes, logistica, ecc.), per classificarli come *positivi*, *negativi* o *neutri*. In questo modo si potevano osservare le variazioni del sentiment degli elettori durante i dibattiti o a seguito di notizie importanti.

-
- **Text classification:** assegnare etichette ai documenti con metodi supervisionati. Un archivio digitale di articoli di giornale può essere classificato automaticamente con modelli supervisionati (SVM, reti neurali, ecc.) per assegnare etichette come *politica*, *sport*, *economia*, *cultura*. Ad esempio: “Il governo approva la nuova legge finanziaria” → *politica/economia*; “La Juventus vince 3–0 in Champions League” → *sport*.
 - **Named Entity Recognition:** identificare entità (luoghi, date, nomi) con modelli di natura probabilistica come ad esempio i ***“Conditional Random Fields”***. Dato un insieme di email aziendali, un modello NER (ad esempio con Conditional Random Fields) può identificare entità come persone, luoghi e date. Frase di esempio: “La riunione con *Mario Rossi* è fissata a *Milano* per il *15 settembre 2025*”. Qui il modello riconosce: *Mario Rossi* → persona; *Milano* → luogo; *15 settembre 2025* → data.

2.3 CRF

Si tratta di modelli probabilistici discriminativi usati per prevedere sequenze di etichette e in più in generale per analizzare dati sequenziali come frasi o testi dove **l'ordine degli elementi conta**.

Un CRF è un modello che calcola la probabilità condizionata di una sequenza di etichette y dato un'osservazione x

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \left(\sum_t \sum_k \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}, t) \right) \quad (2.1)$$

dove :

- $\{f_k\}$: funzioni caratteristiche (feature)
- λ_k : pesi
- $Z(x)$: termine di normalizzazione

come spiegato in [Sutton and McCallum, 2012]

I **CRF** tengono conto del contesto: etichettare una parola può dipendere da quella precedente e successiva

2.4 N-grams:

Gli N-grams descrivono il linguaggio come un processo markoviano.

L'insieme degli stati coincide allora con Θ mentre indichiamo con X_i la variabile che descrive il sistema all'istante i .

Data la sequenza $\{x_1, x_2, \dots, x_{i-1}\}$, si cerca una distribuzione di probabilità per lo stato successivo $\{X_i\}$.

Per un modello di ordine k la proprietà di Markov si esprime nel modo seguente:

$$\begin{aligned}\mathbb{P}(X_i = x_i \mid X_{i-1}, X_{i-2} = x_{i-2}, \dots, X_1 = x_1) &= \\ &= \mathbb{P}(X_i = x_i \mid X_{i-1} = x_{i-1}, \dots, X_{i-k} = x_{i-k})\end{aligned}\tag{3.2}$$

- In un **modello di ordine 1** (bigramma), la probabilità di una parola dipende solo dalla parola immediatamente precedente:

$$P(X_i \mid X_{i-1})$$

- In un **modello di ordine 2** (trigramma), dipende solo dalle due precedenti:

$$P(X_i \mid X_{i-1}, X_{i-2})$$

- In generale, un **modello di ordine k** dipende solo dalle ultime k parole:

$$P(X_i \mid X_{i-1}, X_{i-2}, \dots, X_{i-k})$$

Dunque per definizione un N-gram è un modello markoviano di ordine $K - 1$. Di solito, nonostante il linguaggio naturale sia molto complesso, si utilizzano modelli di basso ordine, noti come *uni-gram*, *bi-gram*, *tri-gram*.

2.4.1 Limiti degli N-grams

- **Crescita esplosiva:** aumentando K , il vocabolario cresce tantissimo e diventa difficile da gestire.
- **Sparsità dei dati:** molti N-grams sono rari e poco informativi.
- Non catturano bene relazioni a lunga distanza (es. soggetto e verbo separati da molte parole).
- Oggi spesso si usano **embeddings** e **modelli neurali** (come BERT), che catturano contesto più ampio.

2.4.2 Probabilità di una frase

Per esprimere la probabilità di una sequenza di simboli $\mathbb{P}(\{x_1, \dots, x_n\})$ che possiamo considerare come una frase, si è soliti introdurre dei simboli deterministici di inizio frase, che indichiamo con s .

Prendendo ad esempio un *bi-gram*, avremo allora:

$$P(\{X_1 = x_1, \dots, X_n = x_n\}) = P(X_1 = x_1 \mid X_0 = s) P(X_2 = x_2 \mid X_1 = x_1) \\ \dots P(X_n = x_n \mid X_{n-1} = x_{n-1})$$

Disponendo di una sufficiente quantità di frasi di esempio, in genere si stima la probabilità

$$P(X_i = x_i \mid X_{i-1} = x_{i-1}, \dots, X_{i-k} = x_{i-k})$$

come:

$$P(X_i = x_i \mid X_{i-1} = x_{i-1}, \dots, X_{i-k} = x_{i-k}) \simeq \frac{\#\{x_i, \dots, x_{i-k}\}}{\#\{x_{i-1}, \dots, x_{i-k}\}}$$

dove “#” si riferisce al conteggio delle volte in cui la sequenza compare nelle frasi di esempio. In altre parole, la probabilità di una frase viene approssimata come il prodotto delle probabilità condizionate di ogni parola dato la parola precedente.

Ad esempio, per una frase con tre parole (x_1, x_2, x_3) , avremo:

$$P(x_1, x_2, x_3) \approx P(x_1 \mid s) \cdot P(x_2 \mid x_1) \cdot P(x_3 \mid x_2).$$

La proprietà di Markov semplifica il calcolo delle probabilità delle frasi assumendo che la parola successiva dipenda solo da un numero finito di parole precedenti (k), e non da tutta la storia passata.

Capitolo 3

Pre-processing

Il pre processing è l'insieme delle operazioni di pulizia e standardizzazione dei dati , per renderli utilizzabili dai modelli di machine learning. Nel caso di dati testuali questa fase consiste nella creazione e nella standardizzazione delle unità testuali (token). Dunque facciamo riferimento al contesto delle rappresentazioni vettoriali di dati testuali (gli embedding) basati su Word2vec e Doc2vec.

3.1 Fasi principali del preprocessing:

- Pulizia del testo: rimozione punteggiatura, simboli e maiuscole
- Rimozione Stopword
- Tokenizzazione
- Embedding (distanza Euclidea ; Word2vec ; Doc2vec)
- Stemming e approccio bag of words
- Named entity recognition
- Parsing
- Costruzione matrice "document-term"

3.2 Pulizia del testo : rimozione della punteggiatura, dei simboli e trattamento delle lettere maiuscole

Un approccio statistico richiede sempre un compromesso: si rinuncia a modellare parte dell'informazione contenuta nei dati in modo che il modello sia più semplice e acquisisca un adattamento più flessibile. A volte la presenza di punteggiatura modifica il significato delle parole in una frase, dunque la sua rimozione coincide con una perdita di informazione. Nonostante ciò, la rimozione della punteggiatura è eseguita nella fase di preprocessing in ogni lavoro di text mining perché sarebbe estremamente complesso tener conto di decine di migliaia di punti e virgole durante la stima dei modelli. Analogamente, vengono rimossi tutti i simboli (\$, n, &. . .).

Un'ulteriore operazione che di solito segue la rimozione dei simboli è la trasformazione di tutte le lettere maiuscole in lettere minuscole. Anche in questo caso si tratta di un'operazione che comporta una piccola perdita di informazione, ma se non fosse eseguita l'algoritmo di stima tratterebbe due parole uguali come parole diverse in presenza di almeno una maiuscola, aumentando dunque il rumore nel testo.

3.3 Rimozione delle stopwords

Si rimuovono dal testo una serie di parole, note come stopwords che sono così comuni da portare un quantitativo di informazione trascurabile.

I vantaggi della lista di stopwords sono la riduzione delle quantità di parole, abbassando così le risorse computazionali del processo e l'aumento della qualità dei dati.

Tabella 3.1: Esempi di stopwords in inglese e italiano

| Categoria | Inglese | Italiano |
|--------------------------|--|---|
| Articoli e pronomi | a, an, the, I, you, he, she, it, we, they | il, lo, la, i, gli, le, un, una, mi, ti, si, ci, vi |
| Preposizioni | in, on, at, for, of, to, with, from | a, di, da, in, con, su, per, tra, fra |
| Congiunzioni | and, or, but, if, because, although | e, o, ma, però, se, perché |
| Verbi comuni e ausiliari | is, are, was, were, be, been, have, has, had, do, does, did, not | essere, avere, è, sono, era, erano, ho, hai, hanno |

3.4 Tokenizzazione

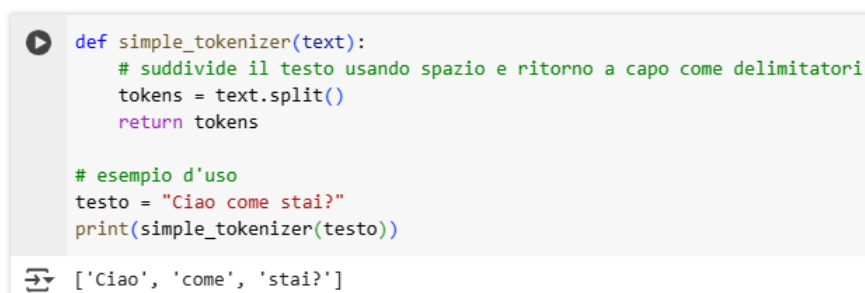
In questa fase si procede con la manipolazione dei dati estraendo un insieme di elementi rilevanti quali parole, frasi o lettere.

La tecnica più utilizzata per eseguire questa operazione è detta tokenization, la quale si occupa di suddividere il testo contenuto in un documento in token cioè ovvero scomporre il testo in una sequenza di unità fondamentali (caratteri) circondata dai delimitatori.

Come primo approccio per la definizione dei delimitatori si può pensare che gli spazi e i caratteri di ritorno a capo possano bastare per una corretta suddivisione. In realtà non è proprio così perché ci sono tante eccezioni come ad esempio i segni di punteggiatura presenti vicino alle parole. L'apostrofo di norma è situato in mezzo a due termini diversi che se ci basiamo sulla definizione data in precedenza, sarebbero erroneamente rappresentati come un unico token.

Un tokenizer è una funzione che riceve una stringa testuale e restituisce la lista dei token. Questa operazione è ciò che consente di creare un set di dati standardizzati e processabili dai modelli di elaborazione del linguaggio che non lavorano direttamente sul testo “grezzo”, ma su rappresentazioni strutturate (sequenze di token).

Esempio di tokenizer molto semplice in Python



```
def simple_tokenizer(text):  
    # suddivide il testo usando spazio e ritorno a capo come delimitatori  
    tokens = text.split()  
    return tokens  
  
# esempio d'uso  
testo = "Ciao come stai?"  
print(simple_tokenizer(testo))
```

['Ciao', 'come', 'stai?']

Figura 3.1: Esempio in python del processo di tokenizzazione

Nel caso in cui come token si utilizzino le singole parole, può essere effettuata la lemmatizzazione, un'operazione che consiste nel ricondurre ogni parola (token) al proprio lemma (ad esempio: mangio -> mangiare, alberi -> albero).

Un lemmatizer è una funzione che esegue questa operazione su ogni parola. Si possono però verificare situazioni con conflitti, cioè possono esserci parole con più di una possibile lemmatizzazione (es. accordo -> accordo, accordo -> accordare). Per questo motivo, molti lemmatizer individuano la parte del discorso relativa a ogni parola del testo (come verbo, aggettivo, ecc.), in modo da assegnare il lemma nella maniera più corretta, riuscendo a compiere la giusta distinzione nei casi ambigui. Dunque è fondamentale creare degli strumenti di tokenizzazione ad hoc in base al contesto in cui ci si trova. Le problematiche aumentano se vogliamo identificare non le singole parole ma intere frasi. Un'ulteriore operazione consiste nel ricondurre ogni parola alla propria radice. La motivazione è che le parole con la stessa radice sono semanticamente molto simili. Ogni parola viene quindi troncata (es calcolare -> calcol, calcolo -> calcol), in questo modo, diverse parole vengono ricondotte alla stessa radice, riducendo la variabilità dei dati (cioè il numero di token

diversi) senza ridurre quella semantica.

Questo passaggio è definito *stemming*, e la funzione che lo esegue è detta **stemmer**

3.5 Embedding

I procedimenti illustrati fino a questo punto hanno trasformato un testo in una collezione di elementi standardizzati. Questi dati necessitano però di una rappresentazione matematica, per poter essere trattati dagli algoritmi, come avviene per un qualsiasi altro tipo di dato (numerico, categorico, ecc..). Questo significa che è necessario costruire dei vettori che rappresentino nel miglior modo possibile il significato semantico del testo di partenza. La rappresentazione vettoriale degli elementi testuali è detta *embedding*. Questa operazione può essere svolta con metodi molto semplici, come Bag-of-Words o Term Frequency-Inverse Document Frequency (TF-IDF), oppure in maniera più elaborata usando particolari architetture di reti neurali come Word2vec o Doc2vec. L'obiettivo è rappresentare una serie di documenti con dei vettori numerici, uno per ogni documento. Sia con il BoW che con il TF-IDF ogni documento costituisce una riga di una matrice, mentre ogni colonna rappresenta una parola del dizionario di parole che costituiscono l'insieme di documenti. Il Bag-of-Words consiste nel contare il numero di occorrenze di una parola all'interno di un determinato documento e nel riportarlo nella corrispondente entrata della matrice. Si può notare che con questa strategia, le parole molto frequenti in tutti i documenti verranno ritenute importanti per ogni documento.

Il TF-IDF consiste invece nell'attribuire alla parola w nel documento d il peso

$$\text{TF-IDF}(w, d) = \frac{n(w, d)}{|\text{dim}(d)|} \cdot \log_{10} \frac{|D|}{|\{d : i \in D\}|} \quad (3.1)$$

dove :

- $n(w, d)$ è il numero di volte che la parola w compare in d
- $|\text{dim}(d)|$ è il numero totale di parole nel documento d
- $|D|$ è il numero di documenti
- $|\{d : i \in D\}|$ numero di documenti in cui il termine w appare

Oppure riscritto con una notazione differente più orientata al trattamento automatico del linguaggio avremo :

$$\text{tf-idf}_{x,y} = \left(\frac{N_{x,y}}{N_{\cdot,y}} \right) \cdot \log \left(\frac{D}{D_x} \right) \quad (3.2)$$

- $N_{x,y}$ è il numero di volte in cui lo stilema x appare nel testo D_y ;
- $N_{\cdot,y}$ è il numero totale di stilemi nel testo D_y ;
- D è il numero complessivo di testi nel *corpus*;
- D_x è il numero di testi in cui lo stilema x compare almeno una volta.

In entrambi i casi stiamo dando peso alle parole che sono molto presenti in uno specifico documento, ma meno comuni in tutti gli altri, ritenendole parole di maggiore rilevanza (cioè che caratterizzano un particolare testo). Detto in altre e più semplici parole l'idea alla base del peso tf-idf è che un termine con una alta frequenza all'interno di un testo dovrà ricevere una maggiore importanza a meno che non sia presente in un gran numero di testi. A quel punto si assume che quel termine sia estremamente comune fra i testi e quindi non sia in grado di discriminare un testo dall'altro.

In generale un peso (w, d) per un termine t nel documento d è calcolato come frequenza del termine $tf(d, t)$ moltiplicata per la frequenza inversa del documento $idf(t)$, che descrive la specificità del termine all'interno della collezione di documenti.

In [Salton et al., 1994] viene proposto uno schema di ponderazione con normalizzazione della lunghezza oltre alla frequenza dei termini e alla frequenza inversa dei documenti, definite come

$$idf(t) := \log \left(\frac{N}{n_t} \right) \quad (3.3)$$

viene utilizzato un fattore di normalizzazione della lunghezza per garantire che tutti i documenti abbiano le stesse probabilità di essere recuperati, indipendentemente dalla loro lunghezza:

$$w(d, t) = \frac{tf(d, t) \log(N/n_t)}{\sqrt{\sum_{j=1}^m tf(d, t_j)^2 (\log(N/n_{t_j}))^2}} \quad (3.4)$$

dove N è la dimensione della collezione dei documenti D e n_t è il numero di documenti in D che contengono il termine t .

In base allo schema di pesi proposto, un documento d è rappresentato da un vettore di pesi dei termini $w(d) = (w(d, t_1), \dots, w(d, t_m))$ e la **similitudine** S tra due documenti d_1, d_2 può essere calcolata tramite il prodotto scalare dei vettori (cioè se li assumiamo normalizzati)

$$S(d_1, d_2) = \sum_{k=1}^m w(d_1, t_k) \cdot w(d_2, t_k) \quad (3.5)$$

3.6 La distanza Euclidea

Una metrica utilizzata di frequente è quella Euclidea.

In questo contesto calcoliamo la distanza tra due documenti

$$d_1, d_2 \in D \quad (3.6)$$

come segue :

$$dist(d_1, d_2) = \sqrt{\sum_{k=1}^m |w(d_1, t_k) - w(d_2, t_k)|^2} \quad (3.7)$$

Tuttavia la metrica euclidea può essere utilizzata soltanto per vettori di norma unitaria.

Poiché senza normalizzazione documenti più lunghi tendono ad avere vettori con norme maggiori, il che può falsare le misure di distanza facendo sembrare due documenti lunghi e poco simili più distanti di due brevi ma anche meno affini.

Infatti per il prodotto scalare (coseno di similarità) per due vettori normalizzati a 1 abbiamo un comportamento simile alla distanza euclidea.

Dati due vettori

$$\cos(\varphi) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| |\vec{y}|} = 1 - \frac{1}{2} d^2 \left(\frac{\vec{x}}{|\vec{x}|}, \frac{\vec{y}}{|\vec{y}|} \right) \quad (3.8)$$

Questi semplici metodi di vettorizzazione consentono di ottenere un vettore numerico, di dimensione pari al numero di parole del dizionario che costituisce i testi, per rappresentare un testo. Il principale svantaggio di questi approcci è la dimensione molto elevata degli embedding generati e la conseguente naturale tendenza ad avere vettori molto sparsi. Un altro punto debole è il fatto che non si colgono le informazioni legate alla posizione delle parole, cioè due testi con le stesse parole, ma ordine delle parole diverso, hanno la stessa rappresentazione vettoriale.

3.7 Word2vec

Un obiettivo degli embedding testuali è di trasformare le singole parole in oggetti matematicamente trattabili, effettuando cioè un word embedding. L'idea alla base del word embedding è di rappresentare le parole come punti di uno spazio vettoriale di grandi dimensioni, in maniera che la loro posizione rappresenti il loro significato semantico. Significa che parole con significati simili devono essere rappresentate con vettori tra loro simili, mentre parole con significati molto diversi attraverso vettori molto lontani tra loro. Word2vec è un modello di rete neurale che, processando un testo, costruisce embedding delle parole che lo compongono. Il funzionamento di Word2vec è basato sull'ipotesi fondante della semantica distribuzionale, ovvero che il significato semantico di ogni parola sia dettato dal contesto in cui essa si trova, cioè dall'insieme di parole che stanno attorno ad essa (come discusso in [Lenci, 2018]).

Perciò due parole sono tanto più simili quanto più tendono a comparire nello stesso contesto. Per descrivere il modello Word2vec si fa riferimento al paper originale pubblicato in [Mikolov et al., 2013] e a [Lane et al., 2019].

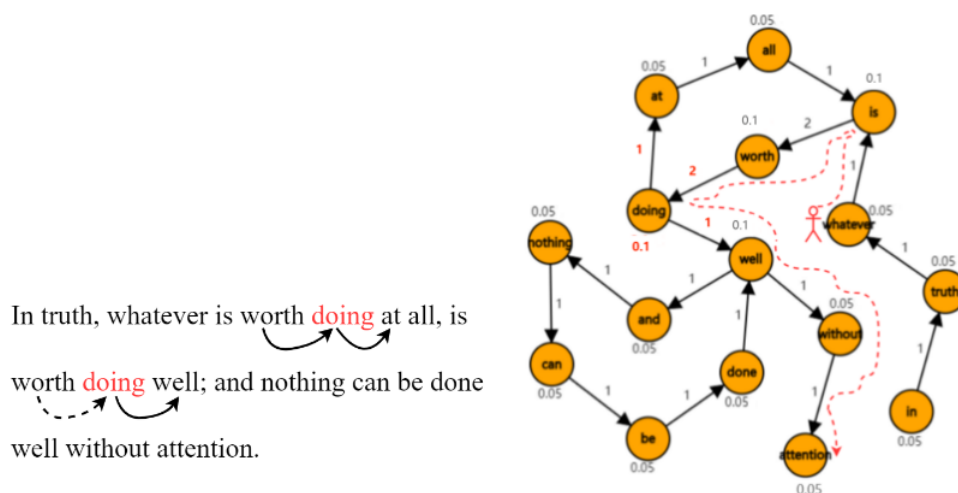


Figura 3.2: Rappresentazione Word2vec

Questa figura illustra come si può costruire un grafo di co-occorrenza delle parole:

-
- Ogni parola diventa un nodo
 - Due parole vengono collegate se compaiono vicine nel testo (cioè nello stesso contesto)
 - Lo spessore/etichetta degli archi indica la forza della connessione (quante volte le parole compaiono insieme)
 - Nel grafo a destra, ad esempio, **“doing”** è collegato a **“worth”** e **“well”** perché nella frase compaiono insieme.

Il percorso rosso tratteggiato rappresenta un random walk che va da **“whatever”** fino a **“attention”**, passando attraverso parole intermedie.

Questo è esattamente il principio alla base di molte tecniche di word embedding basate su grafi (es. Word2Vec): il significato di una parola è catturato dalle relazioni di vicinanza e co-occorrenza con le altre.

Perché random walk?

1. Simula il contesto delle parole: muoversi in modo casuale da una parola all'altra (seguendo i collegamenti del grafo) è come esplorare i contesti in cui le parole appaiono. Se due parole compaiono spesso nello stesso contesto, ci sarà un'alta probabilità che un random walk passi più volte da una all'altra.
2. Cattura relazioni semantiche indirette: non solo relazioni immediate (parole vicine), ma anche relazioni a più passi.

Esempio: “gatto” non compare sempre accanto a “animale”, ma tramite cammini che passano per parole come “cane” o “domestico” il modello può cogliere che “gatto” e “animale” sono concettualmente vicini.

3. Genera sequenze di parole simili a frasi: Il random walk produce delle sequenze di nodi (parole) → queste sequenze possono essere usate come dati di training per algoritmi tipo Word2Vec, che imparano embedding partendo proprio da sequenze.

-
4. Scalabilità: Fare random walk è molto più efficiente che analizzare tutti i possibili percorsi del grafo. Permette di lavorare su dataset molto grandi.

Ci si riferirà alle unità testuali con il termine parole, da momento che, con questa architettura, la tokenizzazione tipicamente utilizzata consiste nella suddivisione per parole. Il modello funziona grazie all'addestramento di una rete neurale, i cui pesi finali costituiscono la rappresentazione delle parole. Vi sono due architetture possibili, il Continuous Bag-of-Words (CBOW) e lo Skip-gram. La prima addestra la rete cercando di prevedere le parole corrente a partire dalle parole attorno ad essa, mentre la seconda cerca di prevedere le parole vicine data la parola corrente. In entrambi i casi, le operazioni vengono effettuate su una finestra di contesto di n parole, cioè si considerano le $n - 1$ parole attorno alla parola in esame. L'architettura è costituita da una rete poco profonda, con layer di input e di output un solo layer nascosto.

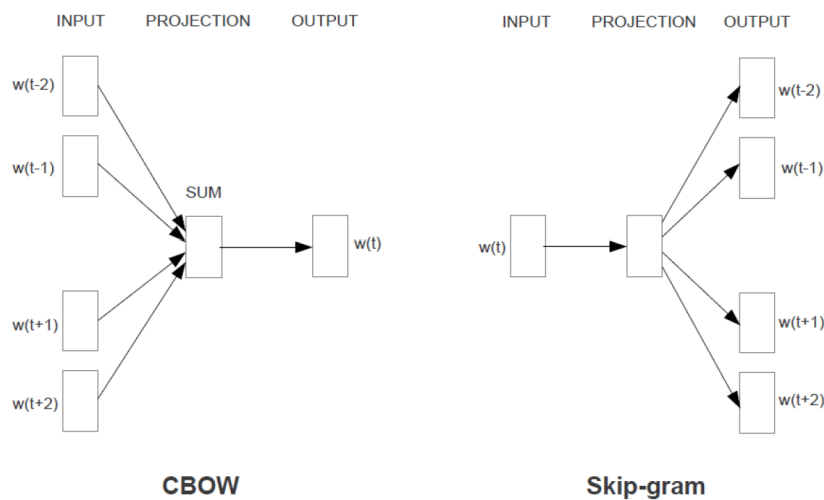


Figura 3.3: Schema delle 2 possibili architetture per il modello Word2Vec, riportato nel paper originale [Mikolov et al., 2013].

3.8 Doc2vec

Le architetture del modello Word2vec consentono di ottenere rappresentazioni semantiche per ogni singola parola. L'interesse più generale è però di ottenere una rappresentazione vettoriale di un'intera sequenza testuale. Il modo più semplice in cui ciò può essere fatto è calcolando la media aritmetica dei vettori Word2vec di tutte le parole di un testo. Esiste però un'architettura che esegue questa operazione in maniera più sofisticata: il Paragraph Vector, anche detto Doc2vec. Questo modello addestra una rete neurale in maniera simile al Word2vec, ma includendo anche un vettore che rappresenta l'intero testo. Esso viene aggiornato durante la scansione dei token, che anche in questo caso corrispondono alle singole parole. Vi sono due possibili architetture per la rete neurale, la Distributed Memory (DM) e la Distributed Bag-of-Words(DBOW), che sono i corrispettivi per il paragraph vector embedding di CBOW e Skip-Gram rispettivamente

3.9 Skip-gram

Uno Skip-gram è un n -gramma, cioè un insieme di $n-1$ parole che presenta una parola mancante al centro di esso.

Uno Skip-gram con n pari a 5 per una data parola sarà perciò costituito dalle 2 parole precedenti e dalle 2 successive alla parola data.

L'input della rete neurale è il vettore one-hot encoding per la parola a cui si riferisce lo Skip-gram (che è la parola da cui devono essere inferite le altre parole di contesto).

Per ogni parola, vengono create delle coppie di training costituite dalla parola di input e una parola dello Skip-gram. L'output della rete neurale è una distribuzione di probabilità sull'insieme delle parole della frase. L'obiettivo è ottimizzare i parametri per predire la parola dello Skip-gram relativa alla coppia di training, concentrando cioè la probabilità dell'output sulla parola di contesto che deve essere individuata. Questa operazione viene effettuata iterativamente su tutte le coppie di training per lo Skip-gram attuale e su tutti gli Skip-gram del testo. Alla fine dell'addestramento, i vettori di embedding delle parole si ottengono a partire dalla matrice di pesi del layer nascosto. Ogni riga della matrice di pesi costituisce l'embedding della parola corrispondente a tale riga. In altre parole, basta moltiplicare la matrice dei pesi per il vettore one-hot encoding della parola di interesse per ottenere il suo embedding.

3.10 CBOW

Il funzionamento del Continuous Bag-of-Words è simile all'architettura Skip-gram, con la differenza che il vettore di input è dato dalla somma dei vettori one-hot encoding dei vettori di contesto. Infatti, a partire da questi, la rete viene addestrata per individuare la parola al centro del contesto. Come per l'architettura Skip-gram, si itera l'addestramento su tutti gli n-grammi mascherati del testo in esame. Anche in questo caso, i vettori di embedding sono costituiti dalle righe della matrice dei coefficienti.

3.11 Distributed Memory

La differenza tra questa architettura e la CBOW è che, oltre alle parole di contesto, viene dato in input anche un vettore di one-hot encoding per il testo di riferimento nella lista dei testi presi in considerazione. In questo modo, oltre alla matrice di pesi W relativa alle parole di contesto, è presente un'altra matrice D che codifica l'intero testo (paragrafo) considerato. Ogni riga della matrice D costituirà quindi l'embedding per il testo a cui corrisponde. Il training avviene iterazioni su tutte le finestre di contesto del testo, includendo sempre il vettore relativo al paragrafo. L'idea di questa architettura è infatti di costruire una matrice che preservi l'informazione semantica di tutte le parole del testo in considerazione. Questa operazione viene poi iterata su tutto il corpus di testi in esame, ottenendo la matrice finale D che conterrà le rappresentazioni vettoriali di tutti i testi.

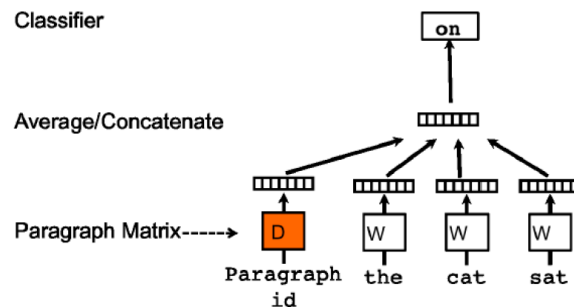


Figura 3.4: Rappresentazione dell'architettura Distributed Memory riportato in [Mikolov and Le, 2014]

3.12 Distributed Bag-of-Words

Questo secondo tipo di architettura consiste nel ricevere in input solamente il vettore one-hot encoding del paragrafo considerato e nel cercare di prevedere l'output, che consiste in una parola campionata dal paragrafo. In particolare viene effettuato un campionamento per quanto riguarda la finestra di contesto e poi un successivo campionamento sulla parola della finestra di contesto da prevedere. I pesi del modello vengono quindi ottimizzati di modo che dal paragrafo si riescano a prevedere le parole che lo compongono. In questo modo, la memoria richiesta si riduce notevolmente, in quanto bisogna memorizzare solamente i pesi della matrice D . Inoltre il training risulta anche più rapido e agevole. Di conseguenza, però, il modello non tiene conto dell'ordine delle parole, in quanto esse vengono campionate casualmente e per questo motivo l'accuratezza è generalmente inferiore rispetto a quella dell'architettura Distributed Memory.

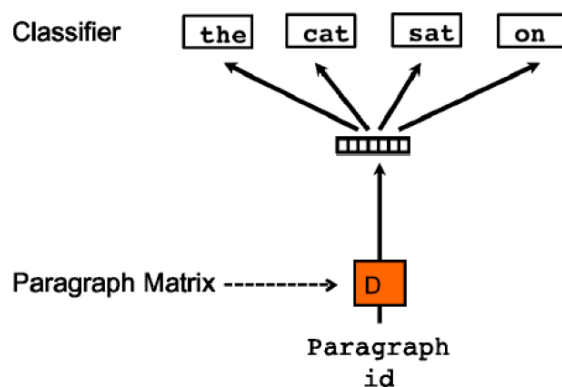


Figura 3.5: Rappresentazione dell'architettura Distributed Memory riportato in [Mikolov and Le, 2014]

3.13 Stemming

Si definisce stemming il processo di riduzione della forma flessa delle parole, alla forma base detta radice, tema o stilema. Dove per radice non si intende l'origine semantica della parola ma la troncatura della stessa, ad esempio le parole "**amando**", "**amava**", "**amato**", "**amare**" vengono tutte ricondotte allo stesso stilema: "**am**".

E' un'operazione successiva alla normalizzazione dei testi.

Questo procedimento non è obbligatorio per gli obiettivi di classificazione dei documenti ma in alcuni casi può offrire un piccolo beneficio. Questo processo consente di:

- **Ridurre** la dimensione del vocabolario del modello
- **Aggregare** parole morfologicamente simili, ma scritte in forme diverse
- **Migliorare** la generalizzazione dei modelli di classificazione testuale

Infatti l'unione di termini simili nella stessa radice potrebbe garantire risultati migliori in fase di classificazione. La creazione di un algoritmo di stemming è stato da sempre uno dei problemi più complicati perché dipende anche dal contesto in cui si sta lavorando e dalla lingua utilizzata. Ad esempio nella lingua inglese bisogna occuparsi anche delle diverse forme irregolari presenti nelle regole grammaticali. Di solito si applicano alcune conversioni del tipo forme plurali in forme singolari ad esempio: " cars " viene convertito in "car". I tempi verbali al passato vengono trasformati sempre nella forma base. Tuttavia è bene ricordare che tutti gli algoritmi che tentano di sviluppare questa tecnica senza tener conto delle regole grammaticali e del contesto o di parti del discorso possono comunque generare un gran numero di errori causati dall'ambiguità derivata dal linguaggio.

Esistono diversi algoritmi di stemming che si differenziano tra loro rispetto all'accuratezza e alle prestazioni.

- **Lookup algorithms (algoritmi di ricerca)** : si ricerca la forma flessa in una tabella di ricerca. Il vantaggio è rappresentato dal fatto che è un algoritmo semplice e veloce ed è facile gestire le eccezioni. Lo svantaggio invece è dato dal fatto che la tabella può essere molto grande perché tutte le forme flesse devono essere esplicitamente elencate: parole nuove o non familiari, non sono gestite anche se perfettamente regolari.
- **Suffix-stripping algorithms (algoritmi di rimozione dei suffissi)** : contengono di solito un elenco di regole che forniscono un percorso per l'algoritmo, infatti dato in ingresso una forma di un termine, viene restituita la sua forma radice (ad esempio se la parola termina in “ed” o in “ing” questi suffissi vengono rimossi). Hanno il vantaggio di essere semplici da gestire rispetto ai lookup purchè l'utilizzatore sia abbastanza esperto nella linguistica e nel creare delle regole di codifica.
- **Algoritmi stocastici** dove si crea un modello probabilistico attraverso una tabella contenente le relazioni tra le forme radici e quelle flesse. Questi modelli spesso sono espressi attraverso complesse regole linguistiche. Passando in ingresso una forma flessa si ottiene in uscita la sua forma base con più alta probabilità di essere corretta.

3.14 Limiti dello stemming

Nonostante lo stemming sia una procedura che nel tempo ha acquistato sempre maggior precisione, rimane un passaggio limitante nel contesto del text mining.

Le complicazioni legate a questa operazione sono evidenti in particolare se ne riscontrano tre:

- Alcune parole si riconducono allo stesso stilema ma hanno significati diversi, ad esempio “foglio” e “foglia” hanno stilema uguale cioè “fogl” e sarebbero considerati come una stessa variabile nonostante il significato diverso.
- Alcune parole hanno lo stesso significato ma non la stessa radice, come per esempio “assente” ed “esenti”. Il significato delle due parole è lo stesso ma gli stilemi a cui vengono ridotte sono differenti.
- Le regole di stemming cambiano a seconda della lingua utilizzata, quindi il fatto che in un testo siano presenti più lingue (cosa piuttosto frequente al giorno d’oggi) può rappresentare un problema.

3.15 Approccio bag-of-words

Si parla di bag of words quando non si considera più l'ordinamento delle parole in una frase ma si mantiene solo l'insieme dei termini scollegati tra loro. La perdita di informazione che si ha smettendo di considerare le parole come ordinate è considerevole, sicuramente superiore alle altre perdite dovute alla normalizzazione dei testi (eliminazione punteggiatura e delle stopwords). Anche questa operazione viene eseguita in tutti i lavori di text mining perché permette di creare una lista di parole presenti in un testo e queste parole possono essere utilizzate come variabili (features) estratte dal testo. Nonostante ci sia una perdita di informazione, l'approccio bag of words rappresenta una buona strategia per estrarre i concetti chiave da un testo.

Si supponga di avere due testi :

1. Il cane correva velocemente nel giardino mentre rincorreva la farfalla
2. I cani correvano nei giardini giocando con le farfalle

Successivamente vengono normalizzati :

1. cane correva velocemente giardino rincorreva farfalla
2. cani correvano giardini giocando farfalle

le due bag-of-words tratte da questi testi sono gli insiemi di termini:

1. {cane,correva,velocemente,giardino,rincorreva,farfalla}
2. {cani,correvano,giardini,giocando,farfalle}

Le parole **cane** / **cani** , **correva** / **correvano** , **giardino** / **giardini** , **farfalla** / **farfalle** sono trattate come termini diversi pur avendo significati quasi identici.

Questo aumenta il numero di variabili nel modello e può introdurre **ridondanza semantica**. Per evitare che il passaggio da frasi a bag of words generi un grande quantitativo di variabili tra cui molte con uguale significato, si ricorre proprio al processo di stemming.

1. **Così andiamo avanti , barche contro corrente , risospinti senza posa nel passato**
2. **Andava sempre solo , parlava poco e si teneva in disparte**

Il verbo andare compare nelle due frasi con forme diverse: **andiamo** e **andava**. Semanticamente hanno la stessa radice verbale ciò può causare ridondanze.

Bag-of-words

1. { and,barc,contr,current,risospint,pos,passat }
2. { and,sol,parl,ten,dispart }

Lo stilema “**and**” fa parte di entrambi gli insiemi *bag-of-words* e può essere considerato come una variabile estratta da entrambi i testi o come una feature comune dei testi. Dunque l’operazione di stemming è completamente automatizzata ed è basata sugli algoritmi visti in precedenza. Quello che ad oggi sembra essere l’algoritmo più utilizzato è stato ideato da Porter nel 1980 che si basa sui sotto-suffissi più piccoli ed è dotato di regole più complesse per la troncatura ma è più efficiente.

3.16 Matrice *document-term*

Avendo a disposizione un insieme di bag of words contenenti gli stilemi che rappresentano l'informazione estratta dai testi , si procede con la costruzione di una matrice document-term cioè una matrice che ha per ogni riga un testo e per ogni colonna uno stilema. Invece per ogni cella ha un indicatore della presenza dello stilema in colonna nel testo in riga. Questo indicatore di presenza può prendere una delle seguenti tre forme ([Miner et al., 2012]) :

- **Indicatore dicotomico:**

In ogni cella è presente un indicatore dicotomico (ad esempio: VERO o FALSO, 0 o 1...) che indica se lo stilema è presente nel testo. Questa rappresentazione è semplice ma efficace, solitamente viene scelta nel caso in cui la frequenza dei termini nel testo non sia di particolare interesse.

- **Conteggio:** In ogni cella è contenuto un numero intero che indica il conteggio del numero di volte che lo stilema si trova nel testo. Questa rappresentazione è indicata quando si vuole tenere conto della frequenza con cui le parole si trovano nei testi.

- **Conteggio pesato:** In ogni cella è contenuto un numero naturale che rappresenta l'importanza (peso) di stilema all'interno del corpus dei testi: esistono diversi pesi utilizzabili per questo scopo, ma la scelta più comune ricade sul peso già descritto in precedenza , ovvero il *tf-idf*. Solitamente il conteggio pesato è la migliore opzione per una document-term matrix.

3.17 Riduzione della dimensionalità

La document-term matrix può essere utilizzata come una comune matrice di regressione: ogni colonna rappresenta una variabile esplicativa estratta dal testo, mentre le etichette associate ai documenti costituiscono le variabili risposta. Si dispongono quindi di tutti gli elementi necessari per stimare un modello di classificazione supervisionata, in grado di prevedere la classe di appartenenza dei documenti sulla base delle informazioni estratte dai testi. Solitamente, una document-term matrix è una matrice sparsa le cui dimensioni risultano elevate anche per collezioni di testi relativamente piccole; è dunque prassi comune ridurre la dimensionalità della matrice, eliminando i termini che compaiono un numero molto limitato di volte. Questa operazione consente in genere di ridurre le dimensioni mantenendo comunque la maggior parte dell'informazione rilevante. Una document-term matrix può anche essere usata per valutare le associazioni tra gli stilemi. Se si immaginano le colonne della matrice come vettori a sé stanti, si può calcolare la correlazione tra due di essi per valutare il grado di associazione fra i termini. *Solitamente, in questo contesto una correlazione maggiore di 0.5 viene considerata ragionevolmente alta per ritenere due termini associati* [Feinerer et al., 2008]. Dunque le analisi di text mining si situano nel caso in cui le quantità di feature osservate è molto maggiore del numero di osservazioni (documenti), cioè $p \gg n$. Risultano perciò di fondamentale importanza metodi in grado di ridurre p .

3.17.1 SVD

Un metodo avanzato per la riduzione della dimensionalità è la **SVD – Singular Value Decomposition**, frequentemente utilizzata nel text mining e nel Natural Language Processing per ridurre la complessità dei dati e identificare le dimensioni latenti del significato.

La SVD consente di costruire combinazioni lineari ortogonali delle variabili (come le frequenze di parole), riducendo l'intero spazio vettoriale a un numero limitato di componenti principali che conservano la maggior parte dell'informazione contenuta nei dati originali.

Questo approccio permette una rappresentazione più compatta e significativa dei documenti testuali, migliorando le performance dei modelli di classificazione e clustering.

Ad esempio, attraverso la SVD si possono identificare le cosiddette *latent semantic dimensions*, ovvero dimensioni astratte del significato che aiutano a mappare i documenti in uno spazio ridotto ma semanticamente rilevante.

In [Manning and Schütze, 2002] viene sottolineato come ogni combinazione lineare ottenuta attraverso la SVD massimizzi l'informazione estratta, mantenendo l'indipendenza rispetto alle altre componenti. Questo rende la SVD uno strumento essenziale per l'analisi statistica di dati testuali ad alta dimensionalità.

$$A = \begin{pmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\ \text{cosmonaut} & 1 & 0 & 1 & 0 & 0 & 0 \\ \text{astronaut} & 0 & 1 & 0 & 0 & 0 & 0 \\ \text{moon} & 1 & 1 & 0 & 0 & 0 & 0 \\ \text{car} & 1 & 0 & 0 & 1 & 1 & 0 \\ \text{truck} & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Figura 3.6: un esempio di matrice document-term

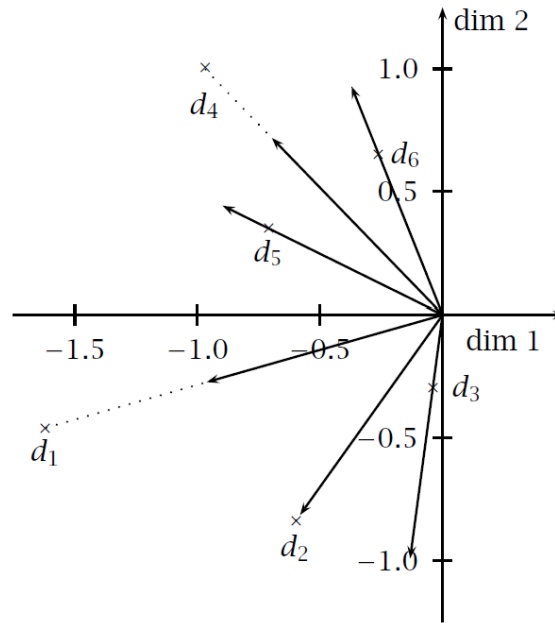


Figura 3.7: Riduzione della dimensionalità

I documenti nella matrice 3.6 sono rappresentati dopo che lo spazio dei termini (originariamente a cinque dimensioni) è stato ridotto a due dimensioni. Le rappresentazioni ridotte dei documenti sono tratte dalla figura 3.7. Oltre ai documenti d_1, \dots, d_6 , sono mostrati anche i loro vettori normalizzati, che evidenziano più direttamente la similarità tramite coseno, utilizzata nella Latent Semantic Indexing (LSI).

3.18 Named entity recognition

Per permettere una comprensione più precisa del testo, è necessario operare una fase detta *named entity recognition*. Le *named entity* sono elementi noti che apportano molta informazione. Solitamente, si vogliono riconoscere nomi di luoghi, di espressioni temporali come date ed ore e quantità interessanti (ad esempio, se si è interessati ad un'analisi di text mining per monitorare l'andamento di un certo indicatore economico, sarà indispensabile riconoscere le cifre che si riferiscono a tale indicatore). Anche in questo caso, il lavoro principale consiste nella redazione di dizionari specifici, che riportino tutti i termini di interesse, ma anche nella costruzione di specifici metodi di machine learning.

Questa fase serve ad estrarre informazioni **strutturate** da testi **non strutturati** come ad esempio (*articoli, documenti legali, e-mail*) e facilitare la ricerca e l'indicizzazione dei testi.

3.18.1 Esempio di Named Entity Recognition

Si consideri la seguente frase:

*Il CEO di **Apple Tim Cook** ha incontrato il primo ministro di **India** a **New Delhi** il **3 luglio 2023** per discutere di investimenti nel settore **tecnologico**.*

Un sistema di Named Entity Recognition correttamente addestrato dovrebbe classificare le entità come segue:

- **Apple, Tim Cook** → ORGANIZATION, PERSON
- **India** → LOCATION / GPS
- **New Delhi** → LOCATION / CITY
- **3 luglio 2023** → DATE
- **tecnologico** → DOMAIN / INDUSTRY

In questo esempio, il sistema non solo riconosce i nomi propri (come persone e luoghi), ma anche entità temporali e settori industriali, dimostrando la capacità del modello di lavorare su classi multiple e contesto semantico.

3.19 Part of speech (POS) tagging

Il “part-of-speech tagging” consiste nell’assegnare a ciascuna parola di una frase la sua parte del discorso. Se lo scopo dell’analisi è l’estrazione di informazioni dettagliate, è opportuno operare un’analisi grammaticale dei token. Dopo aver diviso il testo in frasi, si può quindi procedere con un’assegnazione del ruolo grammaticale (part of speech), attribuendo ad ogni token il ruolo di nome, verbo, aggettivo, avverbio ecc. Naturalmente, per questa fase di preprocessing l’apporto linguistico è fondamentale. Il riconoscimento del ruolo grammaticale di un termine si basa in gran parte sulla costruzione (manuale) di dizionari, detti proprio grammatiche, che riportano un elenco di termini associati ai ruoli. Esistono però molti casi ambigui. In tal caso, è possibile costruire degli appositi classificatori supervisionati, ancora una volta basati sui token presenti in un intorno del termine di interesse. La difficoltà maggiore risiede nella scarsità di testi già annotati che fungano da training set. L’immagine mostra

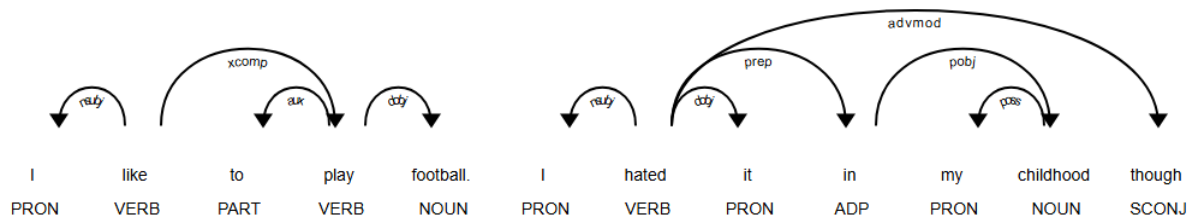


Figura 3.8: Esempio di analisi grammaticale e sintattica con assegnazione delle parti del discorso e relazioni di dipendenza tra parole

una frase in inglese scomposta nei suoi elementi grammaticali (sostantivi, verbi, pronomi, ecc.) con part-of-speech tagging e le relazioni di dipendenza sintattica (come soggetto, oggetto, complemento). Le frecce collegano le parole in base ai loro legami grammaticali, evidenziando la struttura della frase.

3.20 Parsing

Il parsing consiste nell'analisi delle relazioni tra i token di una frase, e nell'assegnazione di un ruolo semantico. L'output del parsing è solitamente un albero, i cui i nodi rappresentano le relazioni e le cui foglie sono i token.

Grazie a tale albero, è possibile ricostruire buona parte dell'analisi logica di una frase ed estrarne informazioni approfondite, che comprendono le relazioni tra i diversi elementi. Avendo già operato la fase di named entity recognition, è perciò possibile individuare le relazioni di interesse tra le named entities ritrovate. In tal modo, si possono costruire sistemi in grado di riconoscere nel testo fatti specifici. Si pensi ad esempio ad un'applicazione che riconosca, analizzando una collezione di articoli di giornali economici, le acquisizioni di società. I nomi delle società rientreranno nella lista delle named entity di interesse, l'acquisizione sarà indicata dalla presenza di verbi specifici ed i ruoli nell'acquisizione saranno riconosciuti grazie al parsing. Esistono diverse applicazioni in grado di generare l'albero di parsing, in particolare rivolte a testi in lingua inglese, ma adattabili anche ad altre lingue.

Capitolo 4

Costruzione del dataset

La creazione di vettori di feature a partire dai documenti si può fare in diversi modi. Dipende principalmente dal tipo di analisi che si vuole operare sui documenti e dalle fasi di pre-processing eseguite. Supponiamo di voler condurre un'analisi dove l'unità osservata è il documento senza concentrarci sulle informazioni apportate dalle diverse frasi. Di solito i problemi affrontati sono la classificazione o la categorizzazione dei documenti oppure un'esplorazione di una collezione di documenti tramite cluster analysis. Verrà costruito in tal caso un vettore di features per ogni documento. Per analizzare un argomento di un generico documento ci focalizziamo sui termini usati. Abbiamo già visto in precedenza le fasi di pre-processing, tuttavia più lo stemming sarà operato in profondità migliori saranno i risultati dell'analisi.

4.1 Classificazione

Una volta che il testo iniziale viene strutturalizzato in record o vettori, si procede all'applicazione di algoritmi di data mining specifici per il machine learning (ad esempio apprendimento automatico). Quest'ultimi, attraverso un processo induttivo, hanno l'obiettivo di trovare delle regole per discriminare i dati iniziali in base alle classi di appartenenza, quindi i testi devono già essere preclassificati.

Tali regole sono poi utilizzate per assegnare la classe appropriata a nuovi dati.

Dato un set iniziale di documenti

$$D = \{d_1, d_2, \dots, d_{|D|}\} \quad (4.1)$$

preclassificati in

$$C = \{c_1, c_2, \dots, c_{|C|}\} \quad (4.2)$$

Si dovrebbe ottenere una funzione $\phi : D \times C \rightarrow \{T, F\}$ che indica se un documento

$$d \in D$$

deve appartenere alla categoria

$$c \in C(\phi(d, c)) = F \quad (4.3)$$

Tale funzione deve essere simile alla funzione obiettivo

$$\phi' : D \times C \rightarrow \{T, F\} \quad (4.4)$$

la quale è considerata come la giusta funzione classificatrice.

Si può valutare l'efficienza utilizzando due metodi :

1. **Training e test set:** il set iniziale viene diviso in due sottoinsiemi T_R e T_S ma non per forza devono avere la stessa cardinalità

- Training set $T_R = \{d_1, d_2, \dots, d_{|T_R|}\}$ usato per costruire le regole del classificatore ϕ'
- Test set $T_S = \{d_{|T_R|+1}, d_{|T_R|+2}, \dots, d_{|D|}\}$ usato per verificare l'accuratezza del classificatore.

Per fare questo , tutti i risultati ottenuti da $\phi'(d_i, c)$ dove $d_i \in T_S$ vengono confrontati con $\phi(d_i, c)$

2. **K-fold cross validation :** consiste di k classificatori $\phi_1, \phi_2, \dots, \phi_k$ diversi, creati dividendo il set iniziale in k sotto-insiemi disgiunti $T_{S_1}, T_{S_2}, \dots, T_{S_k}$ che formeranno i test set.

In seguito si applica k volte il metodo training and test set aventi $T_{R_i} = D - T_{S_i}$

Per ricavare l'accuratezza totale , bisogna fare la media di quelle ricavate dai singoli classificatori.

4.2 Information Gain

Se siamo interessati ad un problema di classificazione supervisionata, è possibile selezionare le feature rilevanti in relazione al training set. La rilevanza della feature j -esima è espressa tramite l'indice *information gain*, indicato come $IG(j)$. Calcolando l'information gain di tutte le feature, è possibile selezionare solamente quelle con indice maggiore, riducendo notevolmente p .

L'information gain esprime quantitativamente il guadagno di informazione che si ha, conoscendo la presenza del termine j in un documento. Immaginiamo quindi di voler operare una classificazione supervisionata, assegnando ogni documento osservato ad uno tra G gruppi. Supponiamo ora che sia dato un training set, e indichiamo con P_g la proporzione di elementi del training set appartenenti al gruppo g , con $g \in \{1, \dots, G\}$. Indichiamo allora con L_{label} la misura di entropia definita come:

$$L_{label} := \sum_{g=1}^G P_g \log_G \left(\frac{1}{P_g} \right) \quad (4.5)$$

Consideriamo ora uno specifico token.
 Dividiamo il training set nei sottoinsiemi:

1. $R_j := \{\text{documenti del training set che contengono il token } j\}$
2. $S_j := \{\text{documenti del training set che **non** contengono il token } j\}$

All'interno di questi insiemi avremo quindi :

- P_j : Proporzione degli elementi del training set contenenti il token j

$$P_j := \frac{\# \text{ documenti del training set contenenti } j}{\# \text{ documenti del training set}} \quad (4.6)$$

- $P_g^j, R :=$ proporzione degli elementi di R_j appartenenti al gruppo g

$$P_g^j, R := \frac{\# \text{ documenti del training set contenenti } j \text{ nel gruppo } g}{\# \text{ documenti del training set contenenti } j}$$

- $P_g^j, S :=$ proporzione degli elementi di S_j appartenenti al gruppo g

$$P_g^j, S := \frac{\# \text{ documenti del training set non contenenti } j \text{ nel gruppo } g}{\# \text{ documenti del training set non contenenti } j}$$

Definiamo allora L_j come:

$$L_j := P_j \sum_{g=1}^G P_{g,R}^j \log_G \left(\frac{1}{P_{g,R}^j} \right) + (1 - P_j) \sum_{g=1}^G P_{g,S}^j \log_G \left(\frac{1}{P_{g,S}^j} \right) \quad (4.7)$$

L_j è tanto più grande quanto più i documenti del training set contenenti j sono distribuiti in modo uguale tra i vari gruppi. Se invece i documenti contenenti j appartengono ad un solo gruppo, L_j sarà minimo. Possiamo allora definire l'indice $IG(j)$ come:

$$IG(j) := L_{label} - L_j \quad (4.8)$$

In questo modo, più l'*information gain* di j è grande, più j dà indicazioni sull'appartenenza di un documento ad un dato gruppo. Per ridurre p si possono dunque scegliere solo le feature per cui IG è maggiore di un'opportuna soglia.

4.3 Classificazione Supervisionata

La classificazione supervisionata è una delle tecniche più usate nell'ambito del text mining. Se ne possono trovare innumerevoli applicazioni, dai filtri anti-spam per la posta elettronica alla categorizzazione di documenti in base all'argomento trattato, fino ad arrivare alla ricerca di documenti contenenti informazioni rilevanti o alla risposta a semplici domande sul contenuto di un documento. In questi casi, le risposte possibili sono predefinite, e il compito si riduce all'assegnazione del documento alla categoria corretta, corrispondente alla risposta appropriata. Definizione: data la collezione di documenti D ed un insieme di categorie C , possiamo definire un classificatore come un'applicazione

$$f : D \times C \rightarrow \{0, 1\} \quad (4.9)$$

Seguendo questa definizione, l'assegnazione delle categorie non è per forza univoca. Un documento potrebbe appartenere infatti contemporaneamente a più di un gruppo. Ciò è naturale nel caso della categorizzazione di documenti: lo stesso testo può infatti trattare più argomenti contemporaneamente. In tal caso, si costruisce per ogni argomento un classificatore binario, che opera indipendentemente dagli altri per l'assegnazione di una singola categoria. La classificazione supervisionata applicata al text mining richiede particolare attenzione per alcuni aspetti. In primo luogo, i testi scritti analizzati sono quasi sempre in continua evoluzione. Un cambiamento nell'autore o una leggera variazione nel modo di trattare un argomento possono causare una modifica sostanziale dello stile dei documenti.

È perciò di fondamentale importanza mantenere il training set in costante aggiornamento, per evitare il degrado nel tempo delle performances dei classificatori. In secondo luogo, si deve tenere presente che molti modelli si basano sull'ipotesi di indipendenza delle feature osservate. Ora, nel caso di un testo scritto, le feature osservate sono i token, generalmente parole, che difficilmente possono essere considerate indipendenti. Si osserveranno perciò dei risultati meno buoni di quanto ci si potrebbe aspettare da un'analisi di data mining in cui le feature sono indipendenti. I metodi più impiegati per la costruzione dei classificatori nell'ambito del text mining sono i metodi deterministici basati sulla definizione di:

1. Decision rules
2. Il Knn (k-nearest neighbors)
3. I classificatori detti bayesiani (anche se non ricorrono ai modelli propri della statistica bayesiana)
4. I classificatori basati su modelli lineari

4.4 Decision Rules

Una decision rule consiste in un elenco di condizioni da verificare, per assegnare un documento ad un dato gruppo g . Dato un documento x , la classificazione si fa semplicemente scorrendo tutte le decision rules pre-stabilite ed assegnando x ai gruppi per cui le condizioni sono soddisfatte. Le condizioni si traducono generalmente nella verifica della presenza di opportuni token o combinazioni di token.

Una decision rule può assumere ad esempio la forma seguente:

Se il documento x contiene i token $w_1; w_2; w_3$ e non contiene il token w_4 , allora x è assegnato al gruppo g

La difficoltà legata a questo tipo di classificazione non è ovviamente di tipo implementativo. Per stabilire delle decision rules è necessaria una conoscenza linguistica approfondita delle problematiche trattate dai testi. Perciò, se da un lato non serve disporre di un training set già suddiviso in gruppi, dall'altro si richiede un lavoro molto approfondito per scrivere delle regole che permettano di ottenere buoni risultati di classificazione. È possibile costruire degli algoritmi in grado di affiancare il lavoro sulla linguistica nella ricerca di pattern informativi. In tal caso, è necessario disporre di un training set annotato, all'interno del quale verranno ricercate delle successioni di parole ricorrenti associabili a un determinato gruppo.

Un grande vantaggio nell'uso delle decision rules è che le regole sono facilmente leggibili: anche un utente non esperto può ricavare informazioni utili dalla lettura delle regole. Inoltre, se si adottano algoritmi specifici per ritrovare le regole, il lavoro linguistico aggiuntivo per l'individuazione delle regole prive di significato sarebbe minimo. Infine, è sufficiente una rappresentazione dei documenti tramite pesi binari:

- **pesi binari:**

$$w_{ij} = \begin{cases} 1 & \text{se } j \text{ compare nel documento } i \\ 0 & \text{altrimenti} \end{cases} \quad (4.10)$$

- **term frequency:** corrisponde al conteggio delle occorrenze del token j nel documento i

$$w_{ij} = tf(i, j) \quad (4.11)$$

- **term frequency normalizzate** si normalizza la term frequency moltiplicandola per il coefficiente, introdotto in precedenza, di *inverse document frequency* riferito al token j :

$$w_{ij} = tf.idf(i, j) := tf(i, j) \times idf(j) \quad (4.12)$$

4.5 Similarità tra documenti

Prima di affrontare più da vicino la costruzione dei classificatori, è necessario richiamare quanto visto in precedenza circa le misure di similarità tra documenti (si noti che la misura di similarità può essere sostituita dal concetto inverso di misura di distanza).

Quelle più frequentemente usate sono:

- **Token comuni** : siano dati due documenti, descritti attraverso i vettori di feature x_1 e x_2 . Supponiamo di aver operato una scelta dei pesi binaria, il conteggio delle parole comuni si traduce nel calcolo del prodotto scalare tra i due vettori. La similarità s sarà allora definita come $s(x_1, x_2) = x_1 \cdot x_2$ dove il simbolo (\cdot) indica il prodotto interno di \mathbb{R}^p ovvero :

$$x_1 \cdot x_2 := \sum_{j=1}^p x_{1j} x_{2j} \quad (4.13)$$

- **Distanza Euclidea** Il modo più naturale per definire la distanza tra due vettori-documento è l'uso della distanza euclidea di \mathbb{R}^p . Dati quindi i documenti x_1 e x_2 costruiti con una qualsiasi scelta per i pesi, si avrà (lo riscriviamo con una notazione più snella del capitolo 3) :

$$d(x_1, x_2) = ||x_1 - x_2|| \quad (4.14)$$

- **Similarità del coseno** La misura di similarità più usata nell'ambito del text mining è la similarità del coseno. Essa è definita come il prodotto scalare normalizzato di due vettori, e coincide con il coseno dell'angolo di incidenza dei due vettori. Dati quindi due documenti x_1 e x_2 , costruiti utilizzando i pesi visti in precedenza, definiamo la similarità s_{\cos} come :

$$s_{\cos}(x_1, x_2) = \frac{x_1 \cdot x_2}{||x_1|| ||x_2||} \quad (4.15)$$

4.6 K-nearest neighbors

L'idea alla base del metodo knn è semplice. Dato un insieme di documenti già classificati (il training set) e dato un nuovo documento x , si ricercano i k documenti del training set più simili a x (equivalentemente meno distanti da x). Se la maggioranza di essi (o una quota Q prestabilita) appartiene al gruppo g , allora anche x è assegnato al gruppo g . La difficoltà principale legata all'algoritmo knn consiste nell'individuare dei valori ottimali per i parametri k e Q . Inoltre, è necessario disporre di algoritmi efficienti per l'estrazione dal training set dei k elementi più simili al documento in esame.

4.7 Naive Bayes

I classificatori chiamati *bayesiani* nella letteratura legata al text mining sono altrettanto noti come metodi naive Bayes. Non si basano infatti su modelli propriamente bayesiani, ma ricorrono alla formula di Bayes per la probabilità condizionata nella definizione del metodo.

4.7.1 Notazioni

Introduciamo in un primo momento alcune notazioni utili nell'esporre i classificatori bayesiani. L'unità statistica è sempre il documento, descritto tramite un vettore di feature. Modelliamo quindi un documento attraverso un vettore aleatorio X . Una singola realizzazione del vettore X sarà indicata con la lettera minuscola x . Diciamo quindi che per una singola istanza si osservano i valori x , realizzazione del vettore aleatorio X . Supponiamo inoltre di costruire il vettore documento attraverso pesi binari (4.10), quindi X assume valori in $\{0, 1\}^p$.

Per quanto riguarda la classificazione, supponiamo che l'assegnazione di un documento ad uno dei G gruppi sia espressa tramite una variabile aleatoria categorica Y , a valori in $\{1, 2, \dots, G\}$.

Supponiamo allora di avere un documento, quindi di conoscere una realizzazione x del vettore aleatorio X , ma di non conoscere direttamente la categoria cui appartiene. Potremmo allora stimare, per ogni possibile categoria $g \in \{1, 2, \dots, G\}$, la probabilità che il documento appartenga ad essa

$$\mathbb{P}(Y = g | X = x) \quad (4.16)$$

per $g = 1, \dots, G$

ed assegnarlo alla categoria con probabilità più alta. Utilizzando la formula di Bayes possiamo allora scrivere:

$$\mathbb{P}(Y = g | \mathbf{X} = \mathbf{x}) = \frac{\mathbb{P}(\mathbf{X} = \mathbf{x} | Y = g) \mathbb{P}(Y = g)}{\mathbb{P}(\mathbf{X} = \mathbf{x})} \quad (4.17)$$

In realtà servirebbero delle stime per queste distribuzioni $\mathbb{P}(X = x)$, $\mathbb{P}(Y = g)$, $\mathbb{P}(X = x | Y = g)$ che non sono note a priori. E possiamo aiutarci utilizzando il training set. Si può stimare la distribuzione delle categorie $\mathbb{P}(Y = g)$ attraverso le proporzioni del training set. Quindi indichiamo con P_g la proporzione di documenti del training set che appartengono al gruppo g .

$$\mathbb{P}(Y = g) \simeq P_g \quad (4.18)$$

Per stimare le distribuzioni di X e $X|Y$ si ipotizza che le componenti del vettore X siano indipendenti. Questa ipotesi è poco realistica: le parole presenti in un testo scritto possono essere difficilmente considerate indipendenti le une dalle altre. Questa ipotesi è perciò il punto debole del modello.

L'indipendenza tra le p componenti di X , indicate con $X_j, j = 1, \dots, p$ permette di scrivere X come prodotto delle marginali :

$$\mathbb{P}(X = x) = \prod_{j=1}^p \mathbb{P}(X_j = x_j) \quad (4.19)$$

allo stesso modo scriveremo

$$\mathbb{P}(X = x | Y = g) = \prod_{j=1}^p \mathbb{P}(X_j = x_j | Y = g) \quad (4.20)$$

Ora, è possibile stimare le distribuzioni delle X_j e le condizionali $X_j | Y$ attraverso le proporzioni nel training set. Ricordiamo che, scegliendo pesi binari, X_j assume valori in $\{0, 1\}$. Abbiamo allora, richiamando la (4.6):

$$\mathbb{P}(X_j = 1) \simeq P_j := \frac{\text{\#documenti del training set contenenti } j}{\text{\#documenti del training set}}$$

In modo analogo, si adotta la stima seguente per $X_j | Y$ e per Y :

$$\mathbb{P}(X_j = 1 | Y = g) \simeq \frac{\text{\#documenti del training set contenenti } j \text{ nel gruppo } g}{\text{\#documenti del training set nel gruppo } g}$$

$$\mathbb{P}(Y = g) \simeq \frac{\text{\#documenti del training set nel gruppo } g}{\text{\#documenti del training set}}$$

A questo punto possiamo stimare la (4.17) dato che le X_j assumono valori in $\{0, 1\}$ scriveremo:

$$\mathbb{P}(Y = g | \mathbf{X} = \mathbf{x}) = \frac{\mathbb{P}(Y = g)}{\mathbb{P}(\mathbf{X} = \mathbf{x})} \prod_{j=1}^p \left[\left(\frac{\mathbb{P}(X_j = 1 | Y = g)}{\mathbb{P}(X_j = 0 | Y = g)} \right)^{x_j} \mathbb{P}(X_j = 0 | Y = g) \right] \quad (4.21)$$

4.8 Valutazione dei classificatori

E' necessario introdurre degli indici per valutare la bontà dei classificatori introdotti. Supponiamo di avere a fianco del training set , un test set composto da documenti la cui categoria è nota, a cui applicare il classificatore per valutarne le performances. Attraverso i risultati ottenuti possiamo calcolare gli indici più usati.

- **error rate:**

$$e_{\text{rate}} := \frac{\# \text{ errori}}{\# \text{ documenti}}$$

- **standard error**

$$SE := \sqrt{\frac{e_{\text{rate}}(1 - e_{\text{rate}})}{\# \text{documenti}}}$$

Ciò che accade spesso nella classificazione binaria di testi, è che i casi di testi classificati positivi sono molto più rari rispetto a quelli classificati negativi. Ad esempio, se stiamo cercando, tra un grande numero di articoli di giornale, gli articoli relativi allo sport (classificati positivi), nella collezione di documenti saranno molto maggiori gli articoli che non trattano di sport, quindi i classificati negativi.

Si tratta di **class imbalance** cioè di sbilanciamento delle classi ed è un fenomeno molto comune nella classificazione binaria dei testi. Il motivo principale è la natura del problema e della distribuzione dei dati nel mondo reale. Quando cerchiamo un certo tipo di contenuto (ad es. articoli sportivi) in una grande collezione di testi (ad es. tutti gli articoli di un giornale), è più probabile che la maggior parte dei documenti non appartenga alla categoria specifica di interesse.

Questo sbilanciamento causa problemi nella modellazione. I modelli possono imparare a 'giocare sul sicuro' e classificare tutto come negativo, ottenendo comunque un'accuratezza alta.

In tal caso, gli indici appena introdotti sono poco indicativi. Se costruiamo infatti un classificatore che associa $y = -1$ ad ogni x , esso classificherebbe correttamente la maggior parte dei documenti, e si avrebbe $e_{rate} \ll 1$.

Per valutare meglio i classificatori in casi simili, si introducono allora altri indici:

- **precision:**

$$\frac{\text{\#classificati positivi corretti}}{\text{\#classificati positivi}}$$

- **recall:**

$$\frac{\text{\#classificati positivi corretti}}{\text{\#documenti positivi}}$$

- **F-measure:**

$$\left[\frac{1}{2} \left(\frac{1}{\text{precision}} + \frac{1}{\text{recall}} \right) \right]^{-1}$$

A seconda dello scopo della classificazione, sarà più utile adottare un indice o l'altro. Ad esempio, per un filtro anti-spam sarà molto importante che le mail classificate come spam (classificazione positiva) non siano in realtà interessanti. Sarà dunque più indicativa la precision rispetto al recall. Al contrario, se stiamo cercando dei documenti che contengono una certa informazione, classificando positivi i documenti rilevanti, sarà fondamentale che tutti i documenti effettivamente rilevanti siano ritrovati, poco importa se è stato individuato qualche documento fuori luogo. In tal caso sarà dunque più interessante il recall. Infine, è importante sottolineare un dettaglio sulla costruzione del test set e del training set. Una delle difficoltà principali per il text mining è legata all'evoluzione temporale dei documenti, che rendono meno efficaci gli algoritmi impiegati. Per avere una valutazione sulla predittività dei classificatori, è importante scegliere il test set in modo che sia composto da documenti più recenti del data set. Ciò permette di verificare le performances di un classificatore tenendo conto degli effetti dell'evoluzione temporale nella collezione di documenti.

4.9 Esempio applicativo: AI vs Human Content Detection

Per integrare gli aspetti metodologici della classificazione supervisionata con un esempio pratico, è stato considerato un piccolo dataset “AI vs Human Content Detection – 1000+ records (2025)” [Puri, 2025]. Tale collezione di testi è composta da 1.367 osservazioni etichettate in due classi principali: contenuti generati da esseri umani e contenuti generati da modelli di intelligenza artificiale (LLM).

Sebbene la dimensione ridotta del dataset non consenta un addestramento robusto di modelli complessi, esso rappresenta un valido proof-of-concept per testare approcci tradizionali di text mining (bag-of-words, TF-IDF, Naive Bayes, regressione logistica) e confrontarli con modelli di nuova generazione come ChatGPT, sfruttato in modalità zero-shot e few-shot learning.

4.9.1 Zero-shot e Few-shot Learning

Nel contesto dei modelli di linguaggio di grandi dimensioni (LLM) come GPT, si parla di **zero-shot learning** quando il modello affronta un compito senza ricevere alcun esempio esplicito nel prompt. In questo caso, il modello si affida esclusivamente alle conoscenze acquisite durante la fase di pre-training.

- **Zero-shot learning:** il modello riceve soltanto l’istruzione in linguaggio naturale. *Esempio:*

Classifica il seguente testo come scritto da un umano o da un’AI: "Il sole sorse dietro le montagne, tingendo il cielo di rosso."

In questo scenario, il modello deve comprendere il compito e fornire una risposta corretta senza alcun esempio di addestramento aggiuntivo.

Diversamente, si parla di **few-shot learning** quando il modello riceve nel prompt alcuni esempi del compito da svolgere. Tali esempi guidano il modello a comprendere meglio la logica di classificazione richiesta.

- **Few-shot learning:** il modello riceve alcuni esempi già etichettati.

Esempio:

Classifica il testo come "umano" o "AI".

Esempio 1: "Ciao, come stai?" → Umano

Esempio 2: "La temperatura media terrestre è aumentata di 1.2°C." → AI

Testo: "Il cane correva nel giardino rincorrendo una farfalla."
→ ?

La differenza principale tra i due approcci può essere riassunta come segue:

| Approccio | Input al modello | Vantaggi | Limiti |
|-----------|--|--|---------------------------------------|
| Zero-shot | Solo istruzione in linguaggio naturale | Non richiede dati annotati, rapido da applicare | Accuratezza talvolta limitata |
| Few-shot | Istruzione + pochi esempi etichettati | Migliore adattamento al compito, maggiore precisione | Richiede la scelta di esempi adeguati |

Tabella 4.1: Confronto tra zero-shot e few-shot learning

Nell'ambito della presente tesi, tali approcci verranno utilizzati per mettere in evidenza la capacità dei modelli generativi di svolgere compiti di classificazione senza la necessità di un dataset di addestramento esteso, mostrando come pochi esempi possano già migliorare significativamente le prestazioni del modello. L'obiettivo principale di questo esperimento non è quello di raggiungere prestazioni ottimali in termini di accuratezza o F1-score, ma piuttosto di mostrare come un problema concreto – il rilevamento di contenuti generati dall'AI – possa essere affrontato sia con metodologie statistiche classiche che con modelli generativi moderni, evidenziandone punti di forza e limiti.

Sono state fornite diverse caratteristiche testuali (features) che consentono di distinguere tra contenuti generati da esseri umani e contenuti generati da modelli di intelligenza artificiale.

Le feature comprendono misure di base come il numero di parole, caratteri e frasi, indici di complessità linguistica (ad esempio la leggibilità di Flesch e l'indice di Gunning Fog), indicatori di stile (uso della voce passiva, punteggiatura, errori grammaticali) e metriche di variabilità del testo (burstiness, prevedibilità). Sono inoltre incluse misure di *sentiment analysis* e indici di diversità lessicale.

La variabile target (`label`) è binaria: assume valore 1 se il testo è stato generato da un modello AI, e 0 se è stato scritto da un essere umano. La Tabella (4.2) riassume il significato delle feature principali.

| Nome Colonna | Descrizione |
|-----------------------------------|--|
| <code>text_content</code> | Contenuto testuale analizzato |
| <code>content_type</code> | Categoria/genere del testo |
| <code>word_count</code> | Numero totale di parole nel testo |
| <code>character_count</code> | Numero totale di caratteri (spazi inclusi) |
| <code>sentence_count</code> | Numero di frasi nel testo |
| <code>lexical_diversity</code> | Rapporto tra parole uniche e numero totale di parole |
| <code>avg_sentence_length</code> | Numero medio di parole per frase |
| <code>avg_word_length</code> | Lunghezza media (in caratteri) delle parole |
| <code>punctuation_ratio</code> | Rapporto tra punteggiatura e caratteri totali |
| <code>flesch_reading_ease</code> | Indice di leggibilità Flesch Reading Ease |
| <code>gunning_fog_index</code> | Indice di leggibilità Gunning Fog |
| <code>grammar_errors</code> | Numero di errori grammaticali rilevati |
| <code>passive_voice_ratio</code> | Rapporto di frasi alla forma passiva |
| <code>predictability_score</code> | Misura della prevedibilità del testo |
| <code>burstiness</code> | Misura della variazione nella lunghezza delle frasi |
| <code>sentiment_score</code> | Polarità del sentimento del testo |
| <code>label</code> | Variabile target: AI (1) oppure Umano (0) |

Tabella 4.2: Descrizione delle feature del dataset *AI vs Human Content Detection* (Puri, 2025).

Dalla Tabella (4.3) si osserva come la lunghezza media dei testi sia relativamente contenuta (circa 140 parole e 25 frasi per documento), con una *lexical diversity* prossima a 1, il che indica una ricca varietà di vocaboli. Il rapporto di punteggiatura è molto basso (circa il 2.7%), coerente con testi brevi e strutturati. La lunghezza media delle parole è stabile intorno a 5-6 caratteri, mentre la variabilità (*std*) suggerisce una certa eterogeneità nella complessità dei testi. Questi indicatori offrono spunti interessanti per distinguere contenuti umani e generati da AI.

| | word_count | character_count | sentence_count | lexical_diversity |
|-------|-------------------|------------------------|-----------------------|--------------------------|
| count | 1367 | 1367 | 1367 | 1367 |
| mean | 140.19 | 940.33 | 25.61 | 0.968 |
| std | 97.41 | 654.34 | 17.87 | 0.026 |
| min | 3 | 14 | 1 | 0.875 |
| 25% | 61.5 | 410.5 | 11 | 0.952 |
| 50% | 131 | 882 | 24 | 0.969 |
| 75% | 193 | 1294.5 | 35 | 0.989 |
| max | 443 | 2966 | 83 | 1.000 |

Tabella 4.3: Statistiche descrittive (parte 1) del dataset *AI vs Human Content Detection* (Puri, 2025).

| | avg_sentence_length | avg_word_length | punctuation_ratio |
|-------|----------------------------|------------------------|--------------------------|
| count | 1367 | 1367 | 1367 |
| mean | 5.49 | 5.72 | 0.027 |
| std | 0.447 | 0.280 | 0.003 |
| min | 3.0 | 4.0 | 0.019 |
| 25% | 5.27 | 5.59 | 0.026 |
| 50% | 5.48 | 5.71 | 0.027 |
| 75% | 5.70 | 5.83 | 0.028 |
| max | 8.0 | 8.33 | 0.071 |

Tabella 4.4: Statistiche descrittive (parte 2) del dataset *AI vs Human Content Detection* (Puri, 2025).

Non si procederà alla rimozione degli outliers perchè:

- La loro incidenza è bassa (tutti inferiori al 7%) e non influenzerà in modo significativo le prestazioni dei modelli.
- Gli outlier nelle variabili di *burstiness* e *predictability* potrebbero contenere pattern importanti per l'identificazione di testi generati dall'AI.
- E' plausibile riscontrare outlier in saggi o articoli più lunghi, dove è naturale avere un numero elevato di parole e una maggiore variabilità (*burstiness*).

| | outliers_count | outlier_pct | min_val | max_val | examples |
|----------------------|----------------|-------------|---------|---------|--|
| word_count | 42 | 3.271028 | 3 | 443 | [[academic_paper, 420], [academic_paper, 425]] |
| burstiness | 0 | 0.0 | 0.1019 | 0.7995 | [] |
| predictability_score | 0 | 0.0 | 20.03 | 119.93 | [] |
| passive_voice_ratio | 0 | 0.0 | 0.05 | 0.25 | [] |
| gunning_fog_index | 63 | 4.906542 | 1.2 | 17.09 | [[social_media, 2.4], [social_media, 2.0]] |

Figura 4.1: OutliersCount

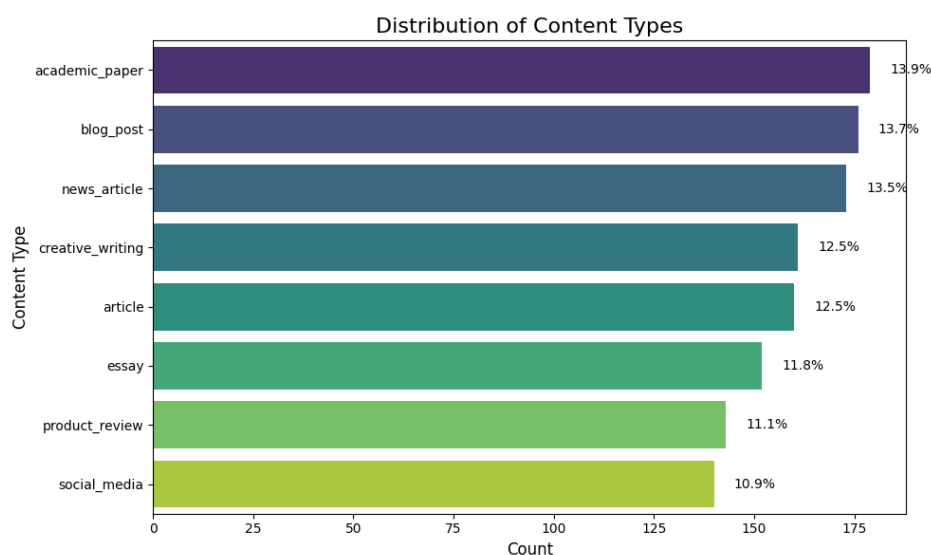


Figura 4.2: Distribuzione delle tipologie di contenuto nel dataset

Dalla Figura (4.2) si osserva una distribuzione bilanciata tra le diverse tipologie di contenuto. Le categorie più rappresentate sono *academic paper* (13.9%), *blog post* (13.7%) e *news article* (13.5%), seguite da *creative writing* e *article* (entrambe 12.5%). Le classi meno frequenti sono *essay* (11.8%), *product review* (11.1%) e *social media* (10.9%). Nel complesso, la distribuzione risulta abbastanza equilibrata e garantisce una buona rappresentatività dei diversi tipi di testo.

I contenuti scritti dalle persone mostrano una maggiore *diversità lessicale* (0.95–1.0 rispetto a 0.9–0.95 per i contenuti AI), in particolare nei testi più lunghi. Nei testi generati dall'AI, la diversità lessicale tende invece a diminuire all'aumentare del numero di parole.

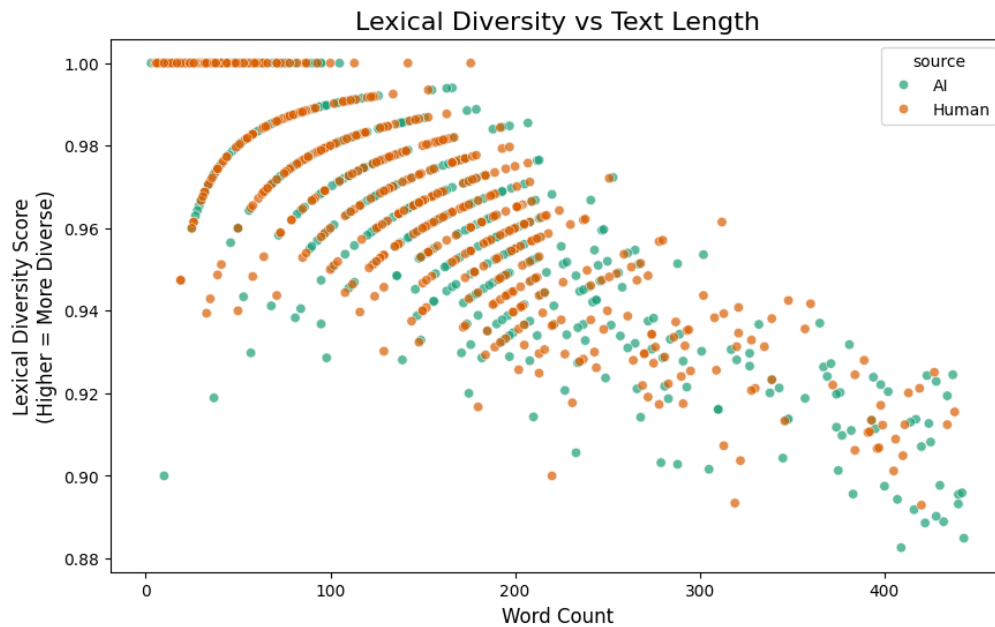


Figura 4.3: Differenza lessicale e lunghezza testo

Risultati principali dell'analisi testuale

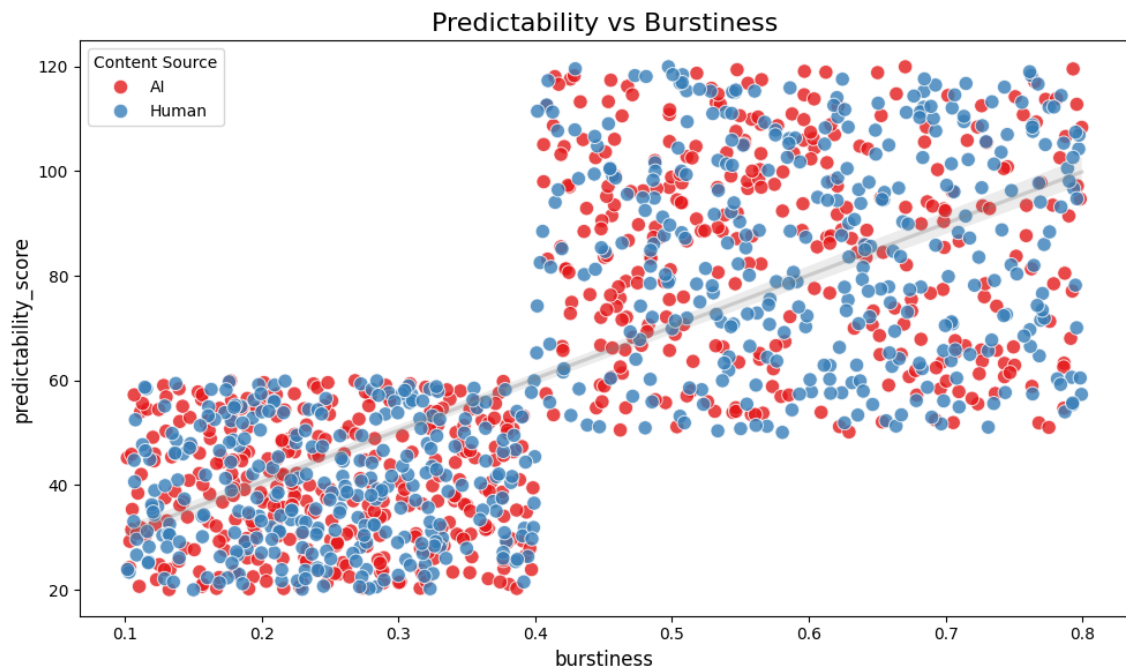


Figura 4.4: Predictability vs Burstiness

- I contenuti generati dall'AI mostrano un sentiment neutro $>$ del 30%.
- I testi umani risultano più soggettivi, con un valore superiore del 40%.
- Il rapporto di parole uniche è maggiore del 15% nei testi umani.
- Le parole più frequenti nei testi AI sono: “people”, “technology”, “development”, “economic”.

Il dataset evidenzia chiare differenze stilistiche tra scrittura umana e scrittura generata da AI:

- La scrittura umana tende ad avere una maggiore diversità lessicale, parole mediamente più lunghe e un uso più vario della punteggiatura.
- La scrittura AI tende ad essere più uniforme, con minore diversità lessicale e un rapporto di punteggiatura più costante.

Queste osservazioni suggeriscono che un modello di classificazione potrebbe sfruttare in particolare la *diversità lessicale*, il *rapporto di punteggiatura* e la *lunghezza media delle parole* come predittori chiave per rilevare testi generati dall'AI.

4.9.2 Analisi delle performance dei modelli

I risultati sono riassunti nei grafici sottostanti. Dal confronto delle curve ROC emerge che tutti i modelli hanno mostrato una capacità discriminativa molto limitata, con valori di AUC compresi tra 0.467 (Gradient Boosting) e 0.547 (Regressione Logistica). In particolare, la logistica risulta il modello con la performance migliore, sebbene il valore rimanga solo leggermente superiore a 0.5

L'analisi della matrice di confusione per la regressione logistica evidenzia una classificazione bilanciata ma con un numero elevato di errori (61 falsi positivi e 61 falsi negativi). La distribuzione dei punteggi di cross-validation conferma come le performance siano complessivamente basse e poco stabili: Logistica e Naive Bayes mostrano una variabilità minore, mentre Gradient Boosting e SVM presentano maggiore dispersione tra i fold.

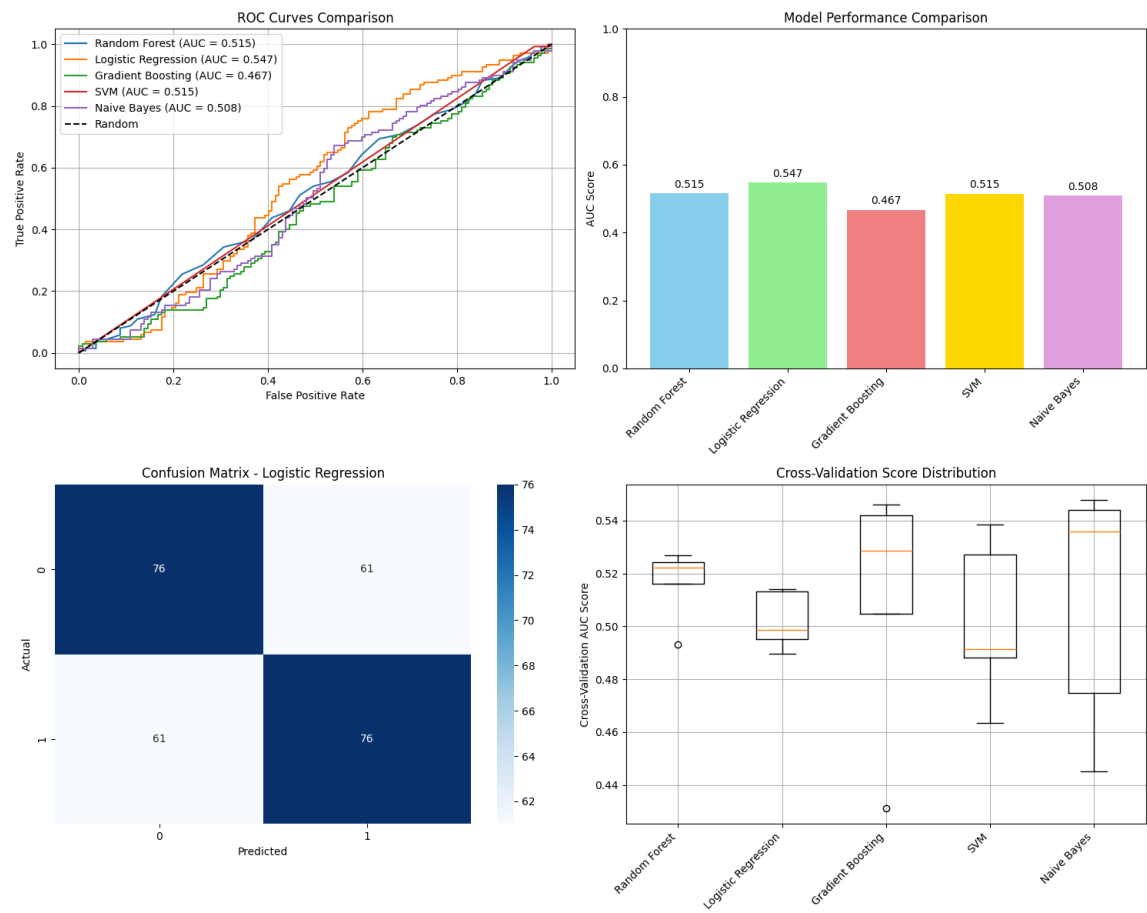


Figura 4.5: Modelli a confronto

Capitolo 5

Descrizione del Dataset

Il dataset è costituito partendo da una collezione di documenti di articoli di giornale provenienti da varie testate giornalistiche (NY Times, Bloomberg, The Guardian) e da post pubblicati su Facebook e Twitter inerenti la politica americana. Il totale dei dati raccolti sono 71357 articoli, con 36330 veri e 35027 falsi. L'unione di queste collezioni per ottenere una dimensione così elevata, presenta vantaggi dal punto di vista di overfitting. Per ogni documento sono stati rilevati il titolo, il testo e l'etichetta "FAKE" o "REAL". Per facilità d'uso, quest'ultima colonna è stata trasformata in una variabile dicotomica, pari a 1 quando la modalità del documento è "FAKE" e 0 altrimenti. Sono stati presi in considerazione il 75% dei dati per l'addestramento e il resto per la validazione: dunque, il primo conterrà 53517 documenti e il secondo 17840. Si parte da una collezione di documenti, etichettandoli. Poi si procede mediante una prima fase che prevede l'utilizzo di tecniche di text mining sui testi, in modo da ottenere dati strutturati. Nella seconda fase si usano le informazioni estratte per applicare le procedure di classificazione del testo, utilizzando modelli statistici differenti. L'obiettivo dunque è quello di rilevare le fake news e le loro caratteristiche mediante approcci statistici classici e reti neurali profonde applicando modelli come BERT.

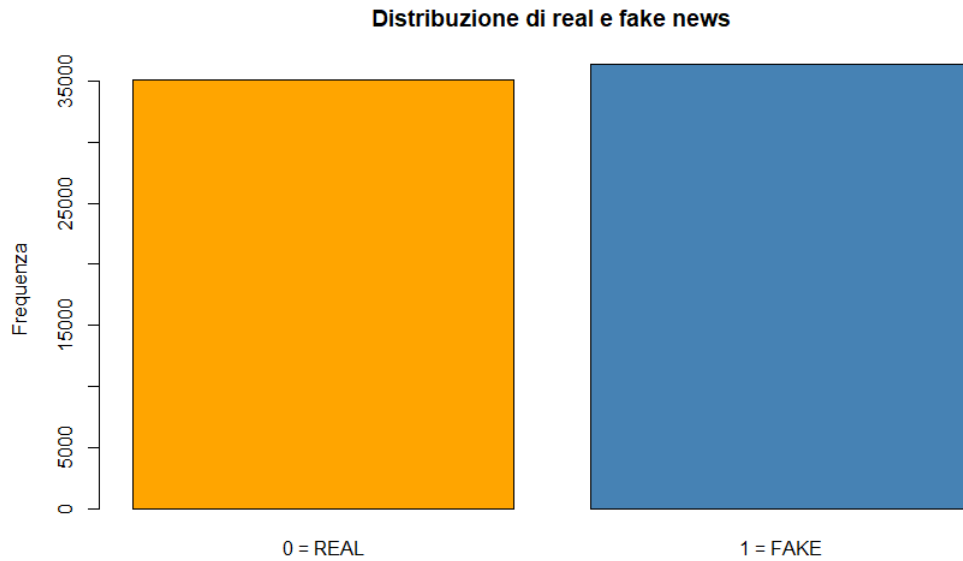


Figura 5.1: Frequenza di real e fake news

Codifica delle fake news: Date le caratteristiche del generico articolo a , l'obiettivo di un procedimento di fake news detection è quello di predire se " a " è fake news oppure no, data la funzione

$$F(a) = \begin{cases} 1 & \text{se } a \text{ è una fake news} \\ 0 & \text{altrimenti} \end{cases}$$

Dunque F è una funzione di classificazione

5.1 Analisi sintattica

Per ogni testo si è voluto estrarre la lunghezza del testo e la percentuale di parole lunghe e corte. Considerando che le fake news essendo meno argomentabili hanno una lunghezza media inferiore. Infatti se confrontiamo la percentuale di fake e real news nella figura (5.2) si vede che la frequenza di informazioni reali in testi con meno di 250 parole è inferiore alla percentuale di fake. Questa invece diminuisce sempre di più considerando i testi di oltre 1000 parole, come invece rappresentato nella figura (5.3). Per quanto riguarda la percentuale di parole lunghe e corte sono state considerate le prime come superiori agli 11 caratteri mentre le seconde come inferiori a 5 caratteri, partendo dalle considerazioni espresse nell'articolo "*Fake News Detection on Social Media: A Data Mining Perspective*" [Shu et al., 2017].

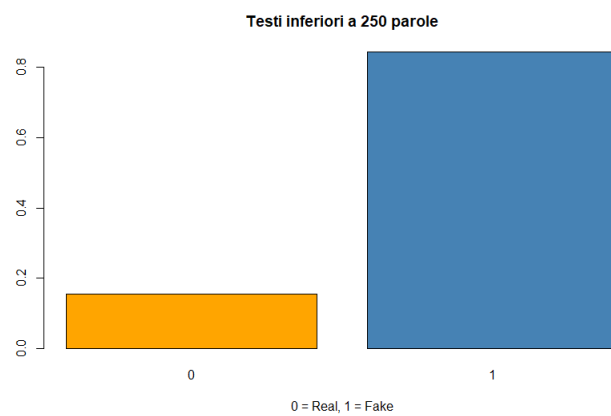


Figura 5.2: Frequenza di fake e real news nei testi con meno di 250 parole

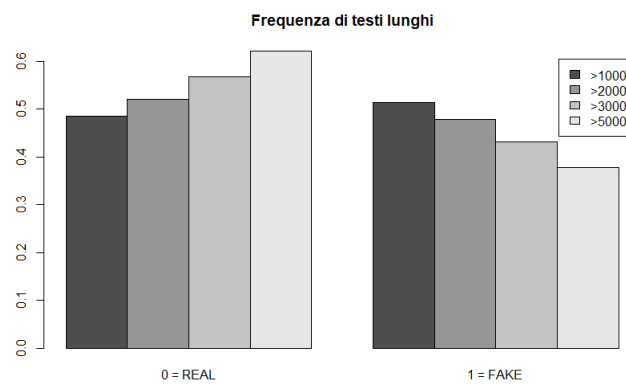


Figura 5.3: Frequenza di 0 (real) e 1 (fake) nei testi con più di 1000, 2000, 3000 e 5000 parole

Dalla rappresentazione delle frequenze di documenti con oltre il 20% di parole superiore agli 11 caratteri figura(5.4) si nota che la frequenza di parole “lunghe” è nettamente superiore nelle fake news.

- Numero di parole medio per frase
- Numero di segni di punteggiatura
- Dato che le fake news non fanno riferimento a fatti accaduti realmente, si suppone che non possano essere presenti numerose citazioni, quindi si è contata anche la presenza di discorsi diretti, contando il numero di “” presenti nei testi. Dai dati infatti vediamo che la frequenza di notizie reali fra i documenti che hanno almeno 1 citazione è decisamente maggiore, come illustrato nella figura (5.5).

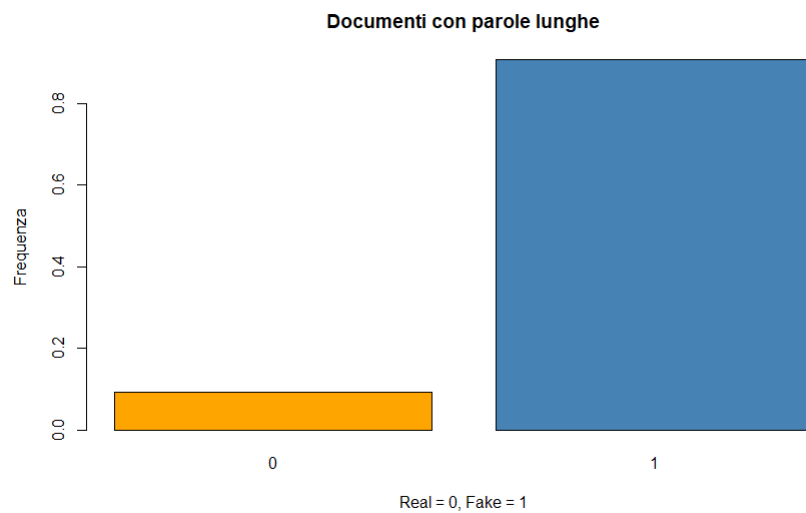


Figura 5.4: Frequenza real e fake news nei testi con più del 20% di parole con più di 11 caratteri

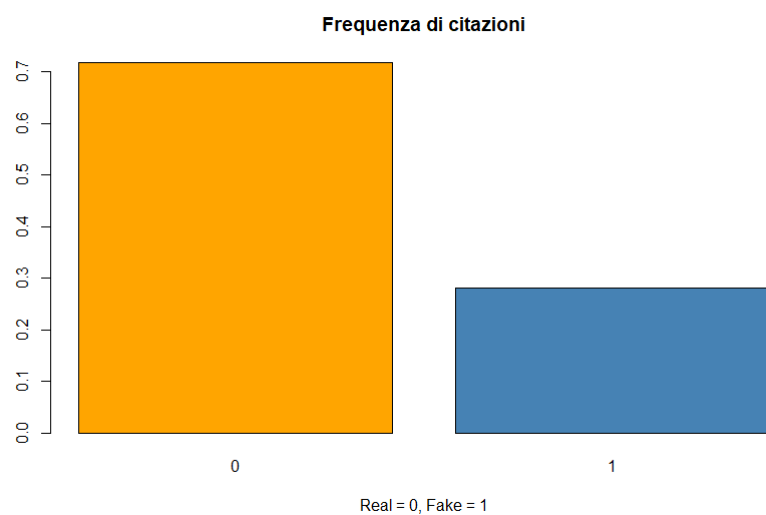


Figura 5.5: Frequenza di real e fake news nei testi con più di 1 citazione

5.2 Bag of words

Dopo l'analisi della struttura sintattica dei documenti, si è svolta un'analisi delle parole più frequenti nelle due tipologie di news considerate. I risultati osservati sono stati ottenuti dopo aver processato i testi: quindi sono state tolte le stop-words e i segni di punteggiatura e si sono uniti i bigrammi più frequenti. Si può notare (nelle figure 5.7;5.8) che ci

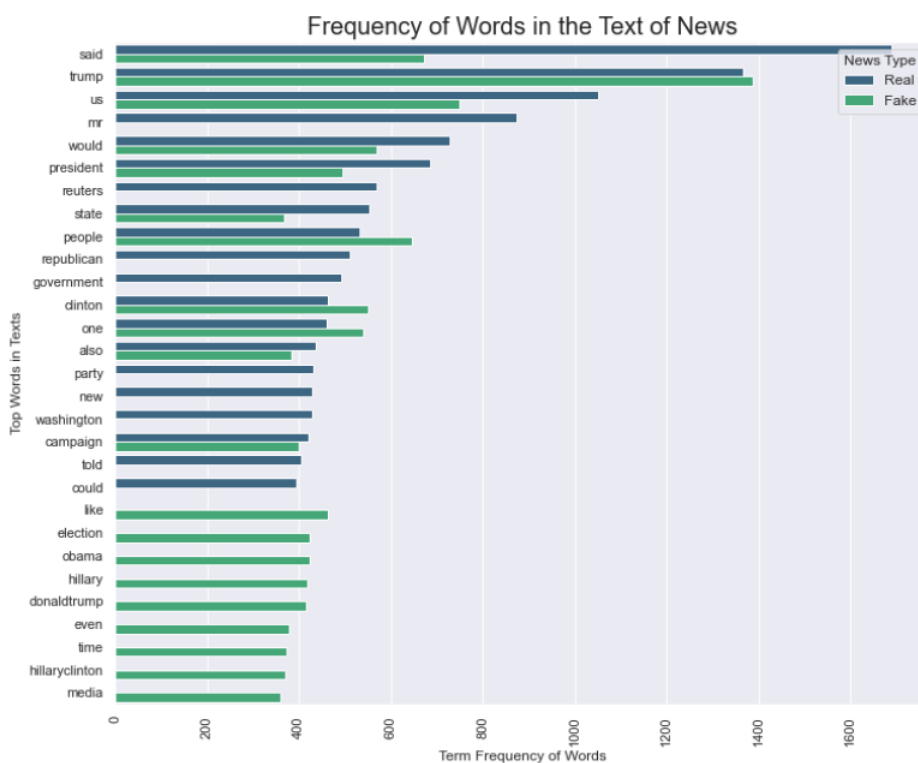


Figura 5.7: Frequenza delle parole più frequenti

sono delle parole con frequenza molto alta sia nelle notizie reali che in quelle false. Non si esclude che il peso delle parole più frequenti delle



Figura 5.8: WordCloud di fake e real news

fake news calcolato in ogni testo, potrebbe essere una caratteristica importante da estrarre per la classificazione. Anzitutto si procede con la creazione della *bag-of-words* di riferimento: questa è costituita dalle 20 parole più frequenti nei documenti falsi. Per ottenere i pesi di rilevanza di ogni elemento della *bag-of-words* nei documenti, si è considerato lo schema di pesatura TF-IDF (*term frequency-inverse document frequency*): come già ampiamente spiegato nel capitolo 3, questa quantità non indica solo il numero di volte che un termine appare nel testo, ma anche l'importanza che ha all'interno dell'intera collezione. In particolare, il TF-IDF prevede che il peso c_{ij} venga calcolato nel modo seguente:

$$c_{ij} = f_{ij} \log \frac{N}{n_j} \quad (5.1)$$

dove f_{ij} è la frequenza della j -esima parola nell' i -esimo documento e n_j è il numero di documenti in cui appare j fra tutti gli N del data set di train.

Dall'analisi delle componenti principali per le 20 parole più frequenti nelle fake news (figura 5.9), si vede che le osservazioni vengono ben distinte tra fake e real lungo l'asse della seconda componente principale, descritta soprattutto da parole come *“said”*, *“president”*, *“hillary”* e *“U.S.”*; al contrario termini come, ad esempio, *“trump”*, *“campaign”*, *“election”* o *“clinton”*, molto frequenti in entrambe le categorie, hanno correlazione quasi nulla con la seconda componente. Tuttavia, la varianza cumulata spiegata dalle due componenti è solo del 20% circa, quindi troppo bassa per poter avere una visualizzazione accurata dei dati originari sul piano bidimensionale.



Figura 5.9: Analisi delle componenti principali della bag-of-words

5.3 Text Clustering

Si è applicata una procedura di clustering dei testi utilizzando un metodo non supervisionato ovvero la **LDA**. Si tratta di un modello probabilistico che si basa sull'assunzione che ogni testo sia considerabile come una mistura di argomenti latenti, dove ogni argomento è caratterizzato da una distribuzione di parole. Il processo di generazione dei topic prevede il campionamento dell'argomento per l' i -esimo documento da θ_i dove $\theta_i \sim Dir(\alpha)$ e si campiona, condizionatamente al topic selezionato, la k -esima parola. Nel train set si sono individuati 8 topic che si possono riassumere con i seguenti titoli: *politica estera statunitense, notizie di attualità, campagna elettorale, rapporti USA-Russia, partito repubblicano, notizie di cronaca, argomenti generali sociologici*. Per ottenere la classificazione anche dei testi nel test set si è utilizzata la probabilità a posteriori:

$$Pr(d_{ij}|t_j, x) \tag{5.2}$$

che definisce la probabilità che l' i -esimo documento del test set riguardi il j -esimo topic, dati i j argomenti trovati nel train set e le x parole del dizionario utilizzato per creare il modello iniziale.

Si è ottenuta l'analisi delle componenti principali della figura(5.10). Dall'analisi risulta che la discriminazione più evidente tra real e fake news avviene lungo l'asse della prima componente, la quale risulta positivamente correlata con gli argomenti di politica estera e rapporti con la Russia. Le osservazioni con punteggi più alti in corrispondenza di queste variabili sono soprattutto notizie vere, mentre quelle con punteggi maggiori in materia di campagna elettorale di Donald Trump sono quasi esclusivamente fake.

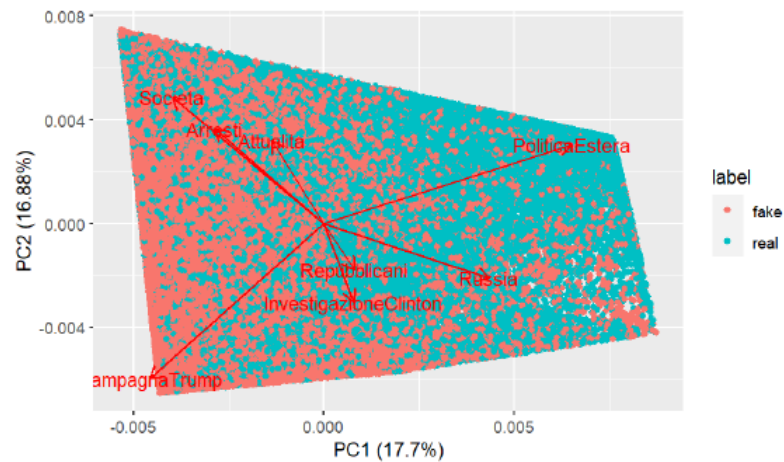


Figura 5.10: Acp dei topic latenti

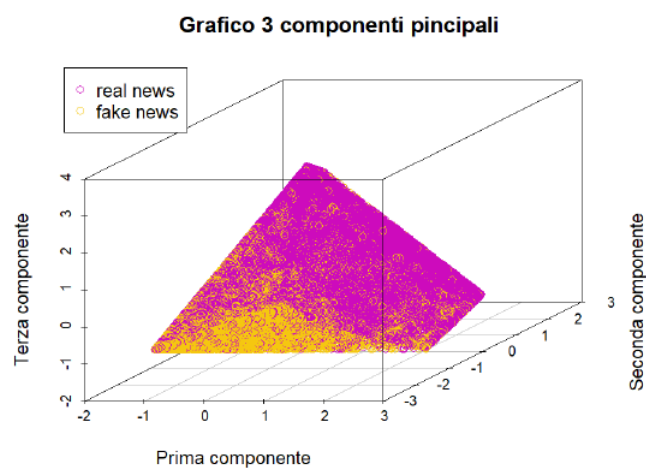


Figura 5.11: Acp topic latenti con terza componente

Tuttavia, le prime due componenti principali spiegano circa solo il 35% della varianza cumulata. Infatti abbiamo maggiore distinzione se prendiamo in considerazione anche la terza componente, la quale porta a una varianza spiegata cumulata del 50% (figura 5.11).

5.4 Rappresentazione generale

In generale, le variabili rilevate portano ad una buona distinzione di notizia vera o falsa, anche analizzandole a gruppi come nei paragrafi precedenti.

Per avere una rappresentazione grafica complessiva della somiglianza tra documenti, si è utilizzato il metodo t-SNE (*Stochastic Neighbour Embedding*), il quale modella i punti in modo che oggetti vicini nello spazio originale di 44 dimensioni risultino vicini nello spazio bidimensionale (e viceversa per gli oggetti lontani), cercando di preservarne la struttura di partenza.

I dati sono stati standardizzati in modo da essere confrontabili. Dalla figura (5.12) vediamo che c'è una buona distinzione delle due categorie rispetto alla seconda coordinata; di conseguenza le osservazioni della categoria “real” avranno valori simili rispetto a questa. La visibile distinzione delle fake e real news fa intendere che le variabili selezionate siano dei buoni indicatori delle due categorie: questa base di partenza assicura che i modelli adottati nel prossimo paragrafo discriminino efficacemente le osservazioni e giungano a buone previsioni.

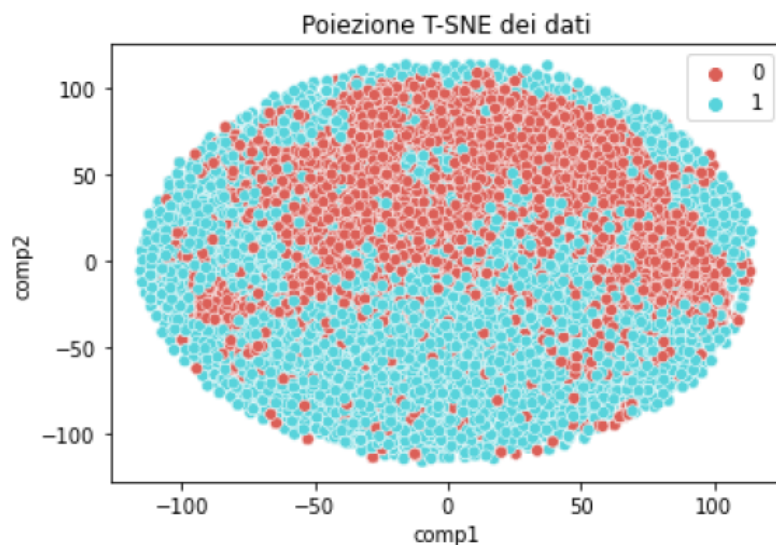


Figura 5.12: Proiezione t-SNE dei dati

5.5 Analisi delle emozioni primarie

Un altro aspetto interessante da tenere in considerazione nell'analisi, sono le emozioni espresse nei testi. Infatti, le fake news vengono scritte appositamente per catturare l'attenzione del lettore e suscitargli un'emozione forte così da favorire la condivisione dell'articolo: per questi motivi ci si aspetta che le fake news contengano una percentuale di emozioni forti alta.

Lo psicologo Robert Plutchik nel libro *Emotion: a Psychoevolutionary Synthesis* [Plutchik, 1980] ha esposto la sua teoria secondo la quale ogni animale prova 8 emozioni primarie: gioia, accettazione, paura, sorpresa, tristezza, disgusto, rabbia e aspettativa. L'idea di Plutchik è che le emozioni possano essere disposte in una “ruota” e combinandone quelle primarie, si ottengono emozioni più complesse, con lo stesso funzionamento della ruota dei colori: per esempio, la delizia è il risultato della combinazione di sorpresa e gioia. Da questa teoria è stata elaborata la *ruota delle emozioni* (Figura 5.13)

Si possono utilizzare dei dizionari specifici per predire la percentuale di ogni emozione espressa nel testo. [Mohammad and Turney, 2013]

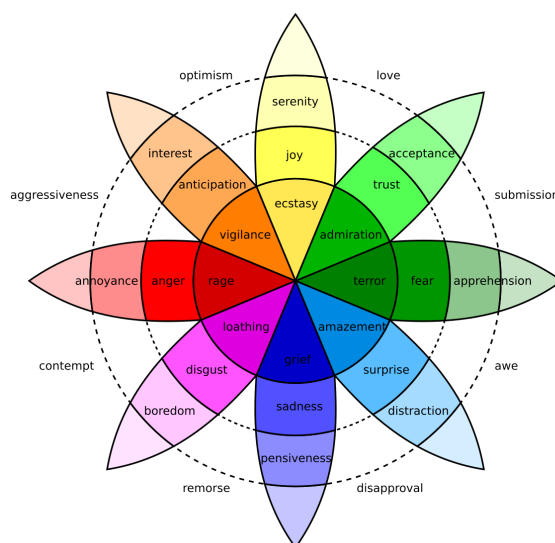


Figura 5.13: Ruota delle emozioni di Plutchik

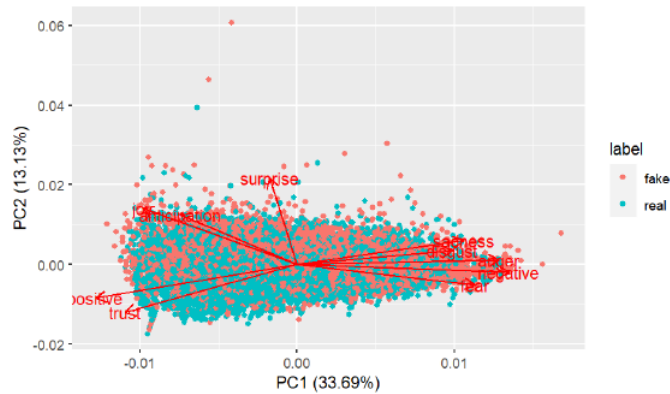


Figura 5.14: Analisi componenti principali delle emozioni primarie

All'interno di questi particolari dizionari troviamo i concetti e i legami logici e non le definizioni delle parole. Recenti aggiornamenti hanno aggiunto anche la classificazione in emozioni negative e emozioni positive, categorie più generiche per poter descrivere anche le parole non allocate nelle 8 emozioni primarie.

Utilizzando i dizionari citati precedentemente, si è ottenuto per ogni documento della collezione la percentuale delle 10 possibili emozioni. Nella figura (5.14) viene rappresentato il risultato dell'analisi delle componenti principali per una visualizzazione in dimensioni ridotte delle osservazioni. La distinzione tra *fake news* e *real news* risulta più chiara rispetto alla prima componente principale, che è correlata positivamente alle emozioni negative come tristezza, rabbia, paura e disgusto: le variabili con punteggi più alti rispetto alla prima componente sono principalmente *fake news*. Al contrario, emozioni positive come fiducia o gioia sono correlate negativamente alla prima componente principale, facendo intuire che osservazioni con punteggi alti per queste variabili sono soprattutto notizie reali.

5.6 Criteri di valutazione

Per procedere all'applicazione dei modelli statistici al *fake news detection* verranno utilizzate le caratteristiche estratte in precedenza dai documenti. Come risposta viene considerata l'etichetta "FAKE" o "REAL" dei documenti, codificata come una variabile dicotomica pari a 1 quando la news è falsa e 0 viceversa.

Le variabili concomitanti sono in totale 44: come già spiegato in precedenza, per ogni testo è stata registrata la sua lunghezza, la percentuale di parole "corte", la percentuale di parole "lunghe", il numero medio di parole nel testo, il numero di citazioni, il numero di segni di punteggiatura, il punteggio TF-IDF per ognuna delle 20 parole più frequenti nelle fake news, la percentuale di parole riferite alle 10 emozioni primarie e infine la percentuale di parole riferite agli 8 topic individuati.

Per valutare l'efficacia dei modelli utilizzati per la classificazione delle fake news, è stata utilizzata principalmente la misura di accuratezza, la quale viene calcolata utilizzando i seguenti indici:

- **VP (Vero Positivo)**: quando una fake news viene classificata correttamente come tale;
- **VN (Vero Negativo)**: quando una real news viene classificata correttamente come tale;
- **FP (Falso Positivo)**: quando una notizia predetta come fake in realtà è una notizia vera;
- **FN (Falso Negativo)**: quando una notizia classificata come reale è in realtà una fake news.

L'accuracy viene calcolata con la seguente formula:

$$\text{Accuracy} = \frac{|VP| + |VN|}{|VP| + |VN| + |FP| + |FN|} \quad (5.3)$$

Un'altra misura di efficacia (introdotta nel capitolo 4) che verrà utilizzata per il confronto tra modelli è il punteggio **F1**, il quale risulta particolarmente efficace rispetto all'accuratezza quando ci sono numerosità molto diverse tra categorie. In questa analisi il numero di notizie reali e false è abbastanza simile, tuttavia è appropriato valutare i modelli con più misure.

Il punteggio F1 è la media armonica di precisione e la recall:

$$\text{Precision} = \frac{|VP|}{|VP| + |FP|} \quad (5.4)$$

$$\text{Recall} = \frac{|VP|}{|VP| + |FN|} \quad (5.5)$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Come strumento grafico verrà utilizzata principalmente la curva ROC che tiene conto degli indici elencati in precedenza: infatti, al variare della soglia di separazione tra le due classi, indica i valori di sensibilità (altro modo per indicare la recall eq.(5.5)) e specificità eq.(5.6).

$$\text{Specificità} = \frac{|VN|}{|VN| + |FP|} \quad (5.6)$$

5.7 Modello di regressione logistica

Data la variabile risposta dicotomica, lo strumento più semplice e intuitivo da utilizzare per la classificazione è il modello di regressione logistica, il quale assume che le osservazioni provengano dalle variabili casuali Y_i con

$$Y_i \sim \text{Bin}(1, \pi_i), \quad Y_1, \dots, Y_n \text{ indipendenti.}$$

Considerando un esempio teorico, la specificazione della probabilità di successo per l' i -esima osservazione diventa dunque:

$$\pi_i = g^{-1}(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}),$$

con $g(\cdot)$ funzione di legame.

Per facilità interpretativa si considererà $g(\cdot) = \text{logit}(\cdot)$.

Come soglia di separazione per la classificazione nelle due categorie, si considera $\hat{y}_i = 0.5$, allocando quindi l' i -esima osservazione al gruppo “FAKE” se il valore predetto è superiore a questa soglia; se inferiore, viene allocato al gruppo “REAL”.

5.8 Analisi discriminante

L'analisi discriminante è una procedura che, al contrario della regressione logistica, è pensata appositamente per la classificazione in problemi di regressione con risposta categoriale.

Il punto di partenza è pensare alla distribuzione della variabile casuale p -dimensionale \mathbf{X} come una mistura delle K distribuzioni di probabilità delle K sub-popolazioni (nel caso presente, $K = 2$). La densità complessiva è quindi:

$$p(x) = \sum_{k=1}^K \pi_k p_k(x) \tag{5.7}$$

con $\hat{\pi}_k = \frac{n_k}{n}$, dove n_k è la numerosità della k -esima sub-popolazione mentre n è la numerosità totale.

Per ogni soggetto con valori di \mathbf{X} noti e pari a x_0 , l'analisi discriminante punta a trovare il valore di k in cui allocare l' i -esima osservazione che massimizza la funzione **discriminante**:

$$d_k(x_0) = \log \pi_k + \log p_k(x_0) \quad (5.8)$$

Per poter applicare questa procedura è però necessario conoscere p_k , stimato con due possibili approcci parametrici:

- **Analisi discriminante lineare (LDA)**: $p_k(x)$ è funzione di densità normale multipla $\mathcal{N}_p(\mu_k, \Sigma_k)$, per cui risulta

$$p_k(x) = \frac{1}{(2\pi)^{p/2} \det(\Sigma_k)^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu_k)' \Sigma_k^{-1} (x - \mu_k) \right\}. \quad (5.9)$$

Viene inoltre assunta omoschedasticità tra gruppi, quindi

$$\hat{\Sigma}_k = \hat{\Sigma} = \frac{1}{n - K} \sum_{k=1}^K \sum_{i:y_i=k} (x_i - \hat{\mu}_k)' (x_i - \hat{\mu}_k),$$

uguale in ogni sub-popolazione.

- **Analisi discriminante quadratica (QDA)**: rimane l'ipotesi distributiva normale per le $p_k(x)$, ma cade l'assunzione di omoschedasticità fra le sub-popolazioni; di conseguenza bisogna stimare un $\hat{\Sigma}_k$ per ogni k .

Albero di classificazione

Gli alberi di classificazione applicano a risposte qualitative l'idea alla base degli alberi di regressione: questi mirano ad approssimare una curva di regressione effettuando un partizionamento nello spazio delle p variabili esplicative, in modo da ottenerne una rappresentazione parsimoniosa tramite una funzione a gradini definibile come:

$$\hat{f}(x) = \sum_{j=1}^J \hat{c}_j I(x \in R_j) \quad (5.10)$$

dove R_j sono le regioni individuate dopo aver effettuato il partizionamento e \hat{c}_j sono le medie di ogni j -esimo gruppo.

La procedura prevede due fasi:

1. **Fase di crescita:** vengono esplorate tutte le possibili suddivisioni per poi effettuare quelle che portano al guadagno maggiore (nel caso degli alberi di regressione, inteso come riduzione della devianza all'interno delle regioni);
2. **Fase di potatura:** mira ad ottimizzare il numero di foglie presenti introducendo nella funzione obiettivo un parametro di penalizzazione della complessità dell'albero.

Nel caso in esame la risposta è dicotomica; di conseguenza si mira ad approssimare

$$p(x) = P\{y = 1 \mid x\},$$

cioè la probabilità che l' i -esimo documento con caratteristiche x sia una fake news.

tramite una funzione a gradini del tipo

$$\hat{p}(x) = \sum_{j=1}^J P_j I(x \in R_j), \quad (5.11)$$

dove P_j rappresenta la probabilità che $y = 1$ nella j -esima regione.

Effettuando un nuovo partizionamento, si crea una nuova foglia con associato valore di $\hat{p}(x)$: la regione R_j verrà allocata alla classe delle notizie “real” se $\hat{p}(x) \leq 0.5$, mentre sarà classificata come fake news se maggiore della soglia di separazione fissata.

Negli alberi di classificazione, non si userà la somma delle devianze all’interno delle regioni come criterio di accostamento per effettuare nuovi tagli, ma un indice di impurità D , il quale corrisponde alla media delle entropie Q delle foglie pesate con le loro numerosità:

$$D = 2n \sum_{j=1}^J \frac{n_j}{n} Q(\hat{P}_j) \quad \text{con} \quad Q(\hat{P}_j) = - \sum_{k=0,1} P_{jk} \log P_{jk} \quad (5.12)$$

Un’alternativa alla misura di entropia per quantificare l’impurità della regione è l’indice di Gini:

$$Q(\hat{P}_j) = \sum_{k=0,1} P_{jk} (1 - P_{jk}) \quad (5.13)$$

La fase di potatura rimane uguale sia in alberi di regressione che di classificazione: si definisce la funzione obiettivo

$$C_\alpha(J) = \sum_{j=1}^J D_j + \alpha J, \quad (5.14)$$

dove α è il parametro di penalizzazione della dimensione J dell’albero.

L’obiettivo della fase di potatura è quello di minimizzare C_α : dunque, dopo aver terminato la fase di crescita ed essere giunti a $J = n$, si incrementa il valore di α che porta all’esclusione sequenziale di quelle foglie la cui eliminazione comporta il minor incremento di $\sum_j D_j$, ottenendo così l’albero ottimale.

Foresta casuale

Gli alberi di classificazione sono dei metodi molto instabili, in quanto i risultati possono cambiare notevolmente a seconda dell'insieme dei dati di train presi in considerazione.

Una possibile soluzione per migliorare la classificazione è rappresentata dalle *foreste casuali*: queste infatti sono combinazioni di alberi applicati a una selezione casuale di F variabili esplicative.

L'idea alla base della fase di crescita degli alberi rimane uguale a quella spiegata nel paragrafo precedente ma, in questo caso, a ogni nuovo nodo non vengono prese in considerazione tutte le possibili suddivisioni delle variabili esplicative per poi scegliere il “cut-point” che porta al maggior guadagno; infatti, vengono esplorate solo F variabili scelte casualmente. Inoltre, non è presente la fase di potatura in quanto il numero ottimale di foglie viene selezionato tramite la combinazione dei diversi alberi. Dunque, sono presenti due parametri di regolazione da ottimizzare: il numero di F variabili da selezionare per ogni nuova suddivisione, selezionando quella che porta ad un errore di classificazione inferiore, e il numero B di alberi che costituiscano la foresta. Per ottimizzare ulteriormente i risultati ottenuti, è possibile creare ogni albero della foresta su n campioni bootstrap invece che sull'intero dataset: quindi, per ogni B -esima iterazione, vengono campionati n sottoinsiemi dei dati su cui effettuare le suddivisioni sulle K variabili.

Per ogni nuovo nodo, viene poi effettuata la media dei risultati ottenuti su ogni n -esimo campione bootstrap, così da ottenere risultati più robusti.

5.9 Gradient boosting

Un'alternativa alla Foresta Casuale per aumentare l'efficacia predittiva degli alberi di classificazione è l'algoritmo *Gradient Boosting*, il quale mira all'ottimizzazione della selezione delle variabili.

Alla base della procedura c'è il concetto di *boosting*: questa è una strategia che prevede l'adattamento di più modelli in modo sequenziale, i quali utilizzano lo stesso insieme di dati ma con probabilità di entrata diversa a ogni iterazione, a seconda di come sono state classificate le osservazioni al passo precedente (verrà assegnato un peso maggiore a quelle che portano a un errore di predizione maggiore).

In primo luogo si definisce la funzione di perdita considerata, la quale deve essere differenziabile: in questo lavoro si è presa in considerazione la log-verosimiglianza $L(y, F_i)$ della binomiale, dove F_i sono i valori previsti dall' i -esimo albero.

La procedura inizia calcolando il valore della funzione di perdita al primo albero F_1 adattato: da qui, per tutte le M iterazioni, il modello al passo $m + 1$ verrà adattato cercando di ridurre la somma di tutte le m funzioni di perdita delle m iterazioni precedenti.

Il nuovo albero risulta:

$$F_{m+1} = F_m - \frac{\partial L(y, F_m(x))}{\partial F_m(x)} \gamma_{m+1}, \quad (5.15)$$

dove γ_{m+1} è il valore che porta all'ottimizzazione della perdita, in quanto viene calcolato come:

$$\hat{\gamma}_{m+1} = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, F_m + \gamma F_{m+1}). \quad (5.16)$$

5.10 Scelta degli iperparametri

Come nell'albero di classificazione bisogna regolare il numero massimo di profondità per poter arrivare alle capacità predittive migliori, anche nella Foresta Casuale e nel *Gradient Boosting* devono essere ottimizzati dei parametri che controllano la precisione del modello: questi vengono chiamati *iperparametri*. Nel caso della foresta casuale bisogna regolare: il numero B di alberi presenti nella foresta, il numero di K variabili da considerare per la creazione di un nuovo nodo, l'indice di impurità (calcolabile con l'entropia (5.14) o l'indice di Gini (5.13)), e l'utilizzo o meno dei campioni bootstrap per la costruzione degli alberi.

Per il *Gradient Boosting* invece si sono ricercati i valori ottimi per M , definito come numero totale di iterazioni, e per il parametro γ di apprendimento, il quale minimizza la funzione di perdita nella creazione del nuovo albero.

Per ottenere gli iperparametri ottimali si è effettuata una *ricerca a griglia*: questa, una volta definiti i possibili valori assumibili da ogni parametro, effettua una ricerca esaustiva esplorando tutte le possibili combinazioni, cercando quella che restituisce il tasso di accuratezza maggiore.

Data la complessità computazionale di tutto il processo, si è voluto introdurre un ulteriore partizionamento dei dati: quindi sono stati presi 21 000 documenti dal training set per formare il validation set. In questo modo l'aggiustamento degli iperparametri avviene su un dataset di train di dimensioni inferiori rispetto a quello iniziale (ora contenente 32 517 testi) e le valutazioni vengono effettuate sul validation set. Il dataset di test viene utilizzato per confermare i risultati ottenuti sul validation set.

Per la foresta casuale si indagano 352 possibili combinazioni di parametri:

- B viene ricercato in un intervallo tra 1000 e 3000, di lunghezza 11;
- K assume valori in un intervallo di 8 elementi da 1 a 20, il quale comprende i valori di default \sqrt{p} o $\log(p)$, dove p è il numero delle variabili totali (44);
- l'indice di impurità può essere "entropy" o "gini";
- l'utilizzo o meno del bootstrap regolato dalla presenza dell'input `bootstrap=True`.

Invece, per il *Gradient Boosting*, si sono esplorati 220 possibili modelli con il valore M che varia in un intervallo di 11 elementi da 1000 a 3000, mentre il tasso di apprendimento può assumere 20 diversi valori tra 0 e 1.

Gli iperparametri considerati si riferiscono agli input richiesti dalle funzioni `RandomForestClassifier` e `GradientBoostingClassifier` della libreria `sklearn.ensemble`.

5.11 Risultati

Regressione Logistica

Il primo passo nella fase di applicazione dei modelli è stato adattare il modello additivo di regressione logistica con tutte le variabili concomitanti a disposizione. La formulazione matriciale diventa dunque:

$$\text{logit}(\pi) = X\beta \quad (5.17)$$

con X matrice a rango pieno di dimensioni $n \times p$, dove n è il numero di osservazioni nel training set, pari a 53 517, e p è il numero di variabili esplicative compresa l'intercetta, quindi 45.

Impostata la soglia di separazione delle due classi a 0.5, il primo modello adattato porta alla seguente matrice di confusione:

Tabella 5.1: Matrice di confusione regressione logistica

| Predetti/Osservati | Real | Fake |
|--------------------|------|------|
| Real | 7435 | 1043 |
| Fake | 1327 | 8035 |

con un tasso di accuratezza del 87.02%.

Successivamente, si sono applicate diverse procedure di selezione automatica delle variabili per selezionare il modello preferenziale rispetto ai dati considerati:

- **selezione passo-a-passo:** questa procedura prevede di modificare il modello iniziale togliendo (*backward selection*) o aggiungendo (*forward selection*) le variabili che portano ad un abbassamento dell'AIC (Akaike Information Criterion), fermandosi quando non si possono ottenere altri miglioramenti.

Applicando questa procedura automatica, si è selezionato un sottoinsieme di 41 variabili, escludendo solo i punteggi TF-IDF dei vocaboli "*time*" e "*one*", la percentuale di parole riferite all'emozione "*joy*", e la percentuale di parole inerenti al topic denominato "*Politica Estera*".

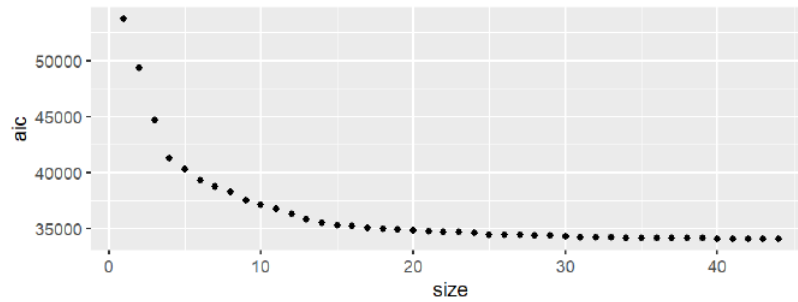


Figura 5.15: Aic

Il tasso di accuratezza ottenuto è pari a 86.99%, quindi leggermente inferiore al modello iniziale.

- **best subset selection:** questo algoritmo mira a selezionare il migliore sottoinsieme di variabili tra tutte le combinazioni possibili, senza cercare un percorso tra esse.

Nella figura (5.15) viene rappresentato come varia l'indice AIC a seconda delle dimensioni del sottogruppo considerato. Questa procedura porta a selezionare il modello con 42 variabili, escludendo anche qui i punteggi riferiti a *"time"* e *"one"*, e in aggiunta gli scores riferiti all'argomento denominato con *"Russia"*.

Il tasso di accuratezza risulta pari a 86.95%.

I tre modelli raggiungono tassi di accuratezza molto simili, con un leggero vantaggio per il modello completo. Tuttavia, il modello preferibile tra i tre risulta quello selezionato con il metodo *passo-a-passo*, in quanto più parsimonioso:

Tabella 5.2: Confronto tra modelli

| Modello | AIC | Accuratezza |
|-------------------------|----------|-------------|
| Modello completo | 34076.89 | 0.8706 |
| Selezione passo-a-passo | 34069.59 | 0.8705 |
| Best subset selection | 34074.15 | 0.8704 |

Analisi discriminante

In questa fase dell'applicazione dei metodi, si sono adottati l'analisi discriminante sia lineare che quadratica. Le due procedure giungono a risultati molto diversi, portando alle seguenti matrici di confusione:

Tabella 5.3: Matrice di confusione LDA

| Predetti/Osservati | Real | Fake |
|---------------------------|-------------|-------------|
| Real | 7295 | 1467 |
| Fake | 914 | 8164 |

Tabella 5.4: Matrice di confusione QDA

| Predetti/Osservati | Real | Fake |
|---------------------------|-------------|-------------|
| Real | 8026 | 736 |
| Fake | 3360 | 5718 |

Notiamo che la tabella 5.3 giunge a risultati simili a quelli della regressione logistica, classificando bene quindi i veri positivi e i veri negativi, con un tasso di accuratezza del 86.67%.

L'assunto di eteroschedasticità invece porta a stimare un numero molto elevato di falsi positivi (notizie reali classificate come false), causando infatti una forte diminuzione dell'accuratezza, che risulta pari a 77.04%.

Albero di classificazione

Per arrivare all'albero di classificazione ottimale, non è sufficiente applicare l'algoritmo di crescita, in quanto si giungerebbe a problemi di *overfitting* e a un albero troppo complesso.

Per ottimizzare la procedura di potatura si è voluto regolare il parametro di complessità α , dalla (5.16). Infatti, si sono eseguiti diversi esperimenti in cui si è fatto variare il parametro α in tutti i suoi possibili valori, confrontando i risultati di accuratezza ottenuti nel test set.

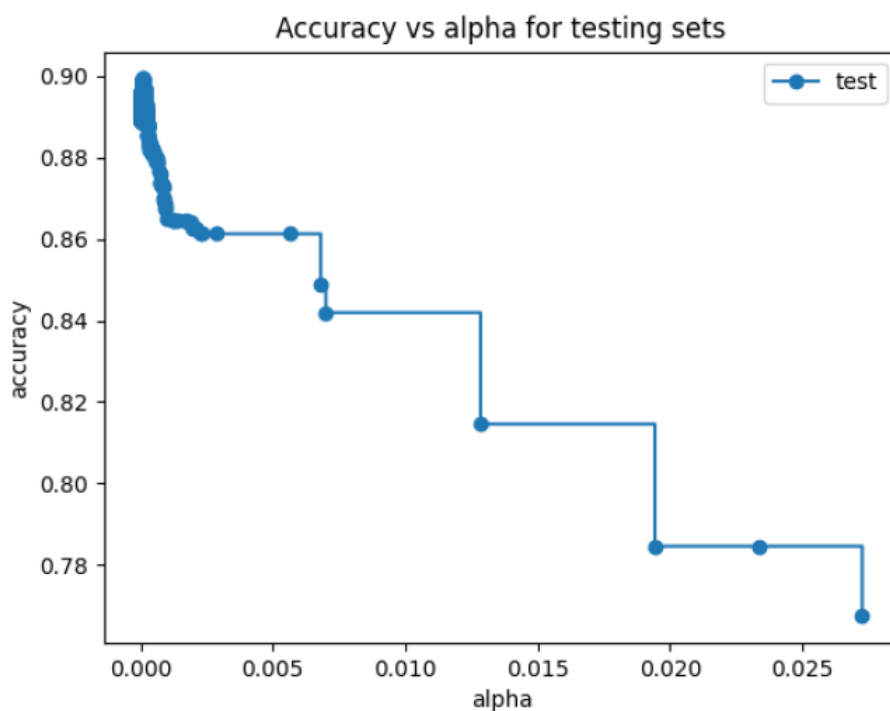


Figura 5.16: Confronto accuracy con parametro complessità

Dalla figura (5.16) notiamo che all'aumentare del parametro di complessità diminuisce l'accuratezza, in quanto si penalizza sempre di più un maggior numero di foglie, portando ad alberi molto potati.

Dunque, si seleziona α dove lo score del test set è maggiore: con questo valore, pari a circa 0.00014, la massima profondità raggiunta dell'albero è 15.

I risultati della figura (5.16) sono stati ottenuti con devianza calcolata tramite l'indice di Gini, in quanto parametro di default; tuttavia, per ottenere l'albero che arriva alle previsioni migliori, oltre al parametro di profondità, si desidera modificare anche l'indice di impurità di default con l'entropia.

Per fare ciò si sono adattati 15 alberi, ognuno con una profondità massima consentita tra 5 e 20 (dato che la massima accuratezza per Gini viene raggiunta con 15), registrando i tassi di accuratezza ottenuti così da poter confrontare i due metodi: nel grafico (5.17) i risultati.

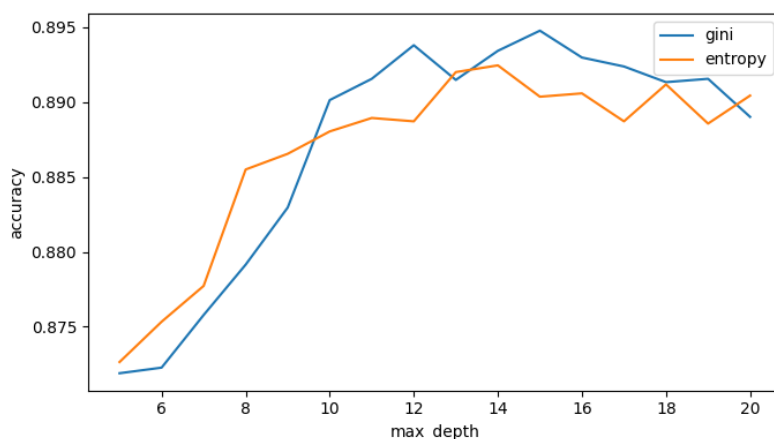


Figura 5.17: Confronto tra accuratezza con entropia e indice di Gini al variare della profondità dell'albero

L'albero ottimale si ottiene dunque utilizzando l'entropia come indice di impurità in corrispondenza di un albero di profondità pari a 14.

Impostando questi parametri si ottiene un tasso di accuratezza dell'89.50% e la seguente matrice di confusione:

Tabella 5.5: Matrice di confusione Albero di classificazione

| Predetti/Osservati | Real | Fake |
|--------------------|------|------|
| Real | 7682 | 1080 |
| Fake | 793 | 8285 |

Foresta casuale

Sono state ricercate tutte le possibili combinazioni degli iperparametri. La configurazione ottima è ottenuta per 2400 alberi presenti nella foresta, costruiti su campioni bootstrap considerando 20 variabili alla costruzione di ogni nuovo nodo. L'indice di impurità utilizzato è quello di entropia.

Adattando la foresta casuale sul data set di train con questa configurazione di iperparametri, si ottiene un tasso di accuratezza del 91.79% e la matrice di confusione (5.6)

Tabella 5.6: Matrice di confusione albero di classificazione

| Predetti/Osservati | Real | Fake |
|--------------------|------|------|
| Real | 7905 | 857 |
| Fake | 608 | 8470 |

Gradient boosting

Dopo aver definito gli intervalli di ricerca delle variabili si sono calcolate le prestazioni di tutte le possibili 220 combinazioni di parametri, registrando per ognuna il tasso di accuratezza.

Come classificatore debole si è mantenuto l'albero ottimale ottenuto in precedenza, quindi con una profondità massima di 14 split e con l'entropia come indice di impurità: fissati questi parametri, si è giunti alla configurazione ottimale con un numero M di iterazioni pari a 1600 e un tasso di apprendimento pari a 0.4.

Il modello adattato con i parametri ottimali raggiunge un tasso di accuratezza del 92.51% con la matrice di confusione (5.7)

Tabella 5.7: Matrice di confusione gradient boosting

| Predetti/Osservati | Real | Fake |
|--------------------|------|------|
| Real | 8038 | 724 |
| Fake | 613 | 8465 |

5.12 Confronto dei risultati

Come mostrato nella Tabella (5.8), i tassi di accuratezza e i punteggi F1 registrati per ogni metodo utilizzato raggiungono punteggi simili tra loro e quindi si può affermare che garantiscano dei risultati robusti.

Il modello che giunge ai punteggi migliori è il gradient boosting, seguito dalla foresta casuale. Anche i metodi semplici sono risultati idonei alla classificazione: la logistica ha una quantità di minimi quadrati ben maggiore, mentre la discriminante lineare porta a risultati migliori rispetto a quella quadratica, che ha classificato male un gran numero di notizie reali come fake news. In generale, la quantità di minimi quadrati osservata nella classificazione logistica è maggiore, e i modelli più complessi hanno discriminato meglio le notizie reali dalle fake. Tuttavia la quantità da minimizzare in un problema di fake news detection sono i falsi negativi, quindi quelle news classificate come notizie reali ma

Tabella 5.8: Confronto dei risultati

| Modello | Accuratezza | F1 score |
|----------------------------------|--------------------|-----------------|
| Modello di regressione lineare | 0.8706 | 0.8741 |
| Best subset selection | 0.8705 | 0.8740 |
| Selezione passo-a-passo | 0.8704 | 0.8739 |
| Analisi discriminante lineare | 0.8667 | 0.8727 |
| Analisi discriminante quadratica | 0.7704 | 0.7363 |
| Albero di classificazione | 0.8950 | 0.8984 |
| Foresta casuale | 0.9179 | 0.9204 |
| Gradient boosting | 0.9251 | 0.9268 |

in realtà sono fake news. Per esempio l'albero di classificazione giunge a un tasso di accuratezza molto elevato, ma il numero di falsi negativi risulta oltre i 1000. Dunque, il gradient boosting risulta preferibile non solo per l'alto numero di osservazioni classificate correttamente, ma anche per avere il tasso minore di FN.

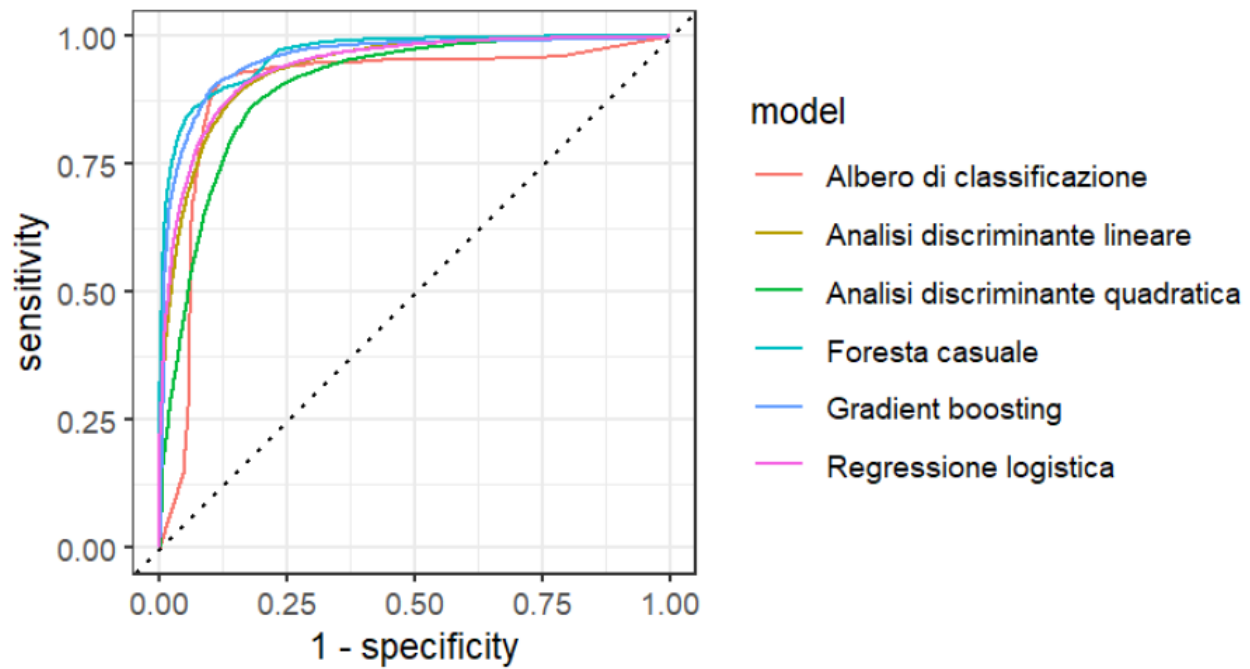


Figura 5.18: Curve ROC

5.13 Conclusioni e approfondimenti finali

Tutti i metodi considerati hanno portato a buone capacità predittive, si può dunque giungere alla conclusione che le caratteristiche estratte dai documenti sono risultate dei buoni parametri per discriminare una notizia falsa da una vera.

In particolare, è possibile utilizzare l’output dell’albero di classificazione, della foresta casuale e del gradient boosting per ottenere il punteggio di importanza di ogni variabile; dato che il modello che ottiene previsioni migliori è il gradient boosting verrà presa di riferimento la tabella (5.9), ma notiamo che la (5.10) e la (5.11) giungono a risultati simili.

Tabella 5.9: Gradi di importanza delle prime 10 variabili per il gradient boosting

| Nomi variabili | Importanza |
|------------------------------------|------------|
| Punteggio TF-IDF “said” | 0.3241 |
| Numero di citazioni | 0.1204 |
| Topic “Politica Estera” | 0.0825 |
| Topic “Repubblicani” | 0.0630 |
| Topic “CampagnaTrump” | 0.0445 |
| Topic “Società” | 0.0283 |
| Numero medio di parole per frase | 0.0229 |
| Numero segni di punteggiatura | 0.0220 |
| Lunghezza testo | 0.0198 |
| Topic “Russia” | 0.0170 |
| Percentuale di emozione “positive” | 0.0147 |
| Punteggio TF-IDF “hillary” | 0.0139 |

Notiamo che la variabile più importante è il punteggio TF-IDF di ogni documento per la parola “*said*”: si pensi al fatto che in un testo in lingua inglese questa parola può essere messa prima di un discorso diretto o indiretto, anticipando dunque il riferimento a una fonte ufficiale. La presenza di citazioni infatti è il miglior modo per distinguere una notizia vera da una falsa, ipotesi confermata dal fatto che anche la variabile denominata “*quotes*” che conta i “ ” presenti in concomitanza di discorsi diretti, si è rivelata con il punteggio di importanza maggiore.

Se guardiamo ai coefficienti del modello di regressione logistica adatto essi confermano la massima verosimiglianza sia del coefficiente “said” che “quotes” sono negativi, e ciò vuol dire che la probabilità che il documento sia una notizia vera aumenta nei documenti che hanno punteggi maggiori in corrispondenza di entrambe le variabili.

Nelle fake news notiamo invece molte parole per frase con una scarsa presenza di segni di punteggiatura, un numero elevato di parole con meno di 6 caratteri e una lunghezza media dei documenti più elevata: i testi con questa struttura sintattica risultano più semplici e chiari, portando dunque messaggi che colpiscono maggiormente il lettore in quanto più intuitivi.

Un altro aspetto significativo da considerare per distinguere la veridicità di un documento è l’emozione espressa dalle parole utilizzate: infatti, alla nona posizione per punteggio di importanza c’è “*positive*”. Questa variabile esprime la percentuale di parole riferite a emozioni positive presenti nel testo. Il coefficiente stimato nel modello di regressione logistica è negativo, quindi come le citazioni, contribuiscono negativamente alla probabilità che l’*i*-esima osservazione sia una fake news. L’effetto contrario invece è prodotto dall’emozione “*rabbia*”, la quale è la seconda emozione più importante.

Per quanto riguarda le variabili rilevate nella text clustering (Par. 5.3), dai punteggi di importanza si nota che rivestono un ruolo importante nella discriminazione delle notizie vere e false: infatti, i documenti con un alto numero di parole riferite ai topic denominati come politica estera americana, campagna elettorale di Donald Trump e rapporti degli Stati Uniti con la Russia, sono soprattutto fake news; al contrario invece il topic “*Società*” sembra entrare negativamente nel modello, indicando maggiore probabilità di avere una notizia reale per i documenti che ne hanno un punteggio alto.

I risultati a cui si è giunti non si possono generalizzare a tutti i problemi di fake news detection, soprattutto per i discorsi riguardanti i topic, in quanto variabili rilevate esclusivamente nei dati in questione; tuttavia, l'elevato numero di documenti di genere così vario presenti nel dataset porta a un abbassamento della distorsione di un solo insieme di dati e quindi a risultati più robusti.

La minaccia rappresentata dal diffondersi di fake news nel web non è un fenomeno ignorabile e tutti gli utenti di social network o siti d'informazione online devono prestare attenzione a non contribuire all'allargamento del fenomeno. I metodi automatici per far fronte alla questione, come si è analizzato nel presente lavoro, possono portare a risultati molto efficaci, rappresentando dunque una base affidabile; tuttavia, affidarsi a siti ufficiali o verificare una notizia da più fonti è il modo più semplice per assicurarsi una buona qualità delle informazioni recepite.

Tabella 5.10: Gradi di importanza delle prime 10 variabili per foresta casuale

| Nomi variabili | Importanza |
|------------------------------------|-------------------|
| Punteggio TF-IDF “said” | 0.2265 |
| Numero di citazioni | 0.1321 |
| Topic “Politica Estera” | 0.1232 |
| Topic “Repubblicani” | 0.0682 |
| Topic “Campagna Trump” | 0.0474 |
| Numero medio di parole per frase | 0.0307 |
| Numero di segni di punteggiatura | 0.0264 |
| Lunghezza testo | 0.0251 |
| Topic “Società” | 0.0228 |
| Percentuale di emozione “positive” | 0.0159 |
| Punteggio TF-IDF “hillary” | 0.0153 |

Tabella 5.11: Gradi di importanza delle prime 10 variabili per albero di classificazione

| Nomi variabili | Importanza |
|----------------------------------|-------------------|
| Punteggio TF-IDF “said” | 0.2902 |
| Topic “Politica Estera” | 0.1598 |
| Numero di citazioni | 0.0956 |
| Topic “Repubblicani” | 0.0767 |
| Numero medio di parole per frase | 0.0326 |
| Topic “Società” | 0.0299 |
| Lunghezza testo | 0.0246 |
| Topic “Campagna Trump” | 0.0192 |
| Percentuale di parole “corte” | 0.0124 |
| Topic “Russia” | 0.0118 |

Bert

Un modello che ha rivoluzionato il campo dell'elaborazione automatica del linguaggio naturale è BERT (Bidirectional Encoder Representations from Transformers), introdotto da Google nel 2018. La sua peculiarità consiste nell'impiego dell'architettura Transformer, e in particolare dell'encoder bidirezionale, che permette di rappresentare le parole di un testo considerando contemporaneamente sia il contesto a sinistra che quello a destra. Questo approccio consente di superare i limiti dei modelli precedenti, spesso unidirezionali, e di ottenere rappresentazioni semantiche molto più ricche ed efficaci. BERT viene pre-addestrato su grandi quantità di testo attraverso due compiti principali: il *Masked Language Modeling*, in cui alcune parole vengono oscurate e il modello deve predirle sulla base del contesto, e il *Next Sentence Prediction*, in cui il modello deve stabilire se due frasi sono consecutive in un testo. Grazie a questo pre-training, BERT può essere successivamente *fine-tuned* su compiti specifici come classificazione del testo, analisi del sentiment, question answering. È importante sottolineare che, mentre nei modelli tradizionali di text mining (like Bag-of-Words o TF-IDF) la rimozione delle stopwords era una prassi comune per ridurre la dimensionalità e migliorare l'efficienza computazionale, nei modelli moderni basati su *self-attention* non è più necessario eliminare manualmente tali parole. Infatti, come spiegato da [Vaswani et al., 2017] nel paper "Attention is All You Need", l'architettura Transformer è in grado di attribuire automaticamente un diverso peso (*attention weight*) ad ogni parola della sequenza, in funzione del contesto. Questo implica che anche termini ad alta frequenza e apparentemente poco informativi, come articoli, preposizioni o ausiliari, possano risultare determinanti per la corretta interpretazione semantica della frase. Pertanto, nei moderni modelli di linguaggio pre-addestrati (ad esempio BERT e le sue varianti), la rimozione preventiva delle stopwords non solo è superflua, ma rischierebbe di compromettere la qualità della rappresentazione contestuale.

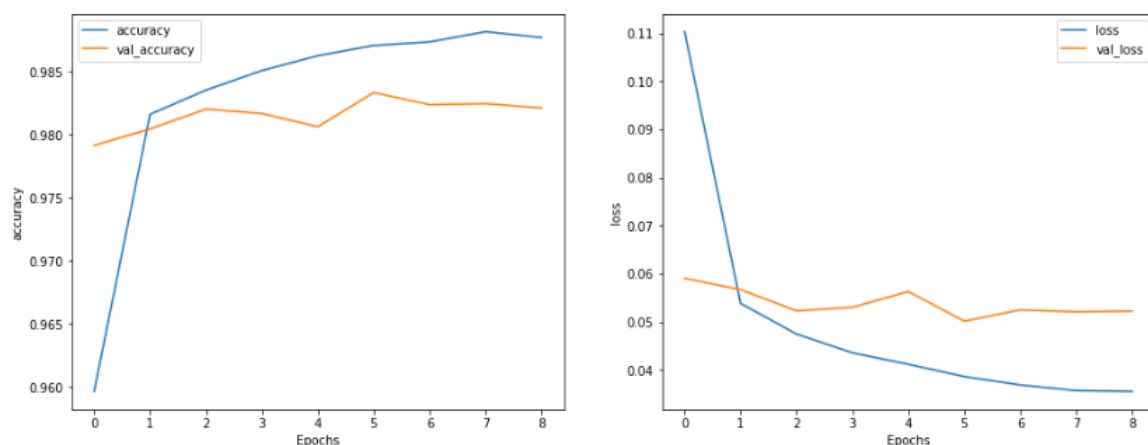


Figura 5.19: Caption

Il modello ha mostrato un'elevata capacità di generalizzazione. Dalla (Figura 5.19) notiamo una rapida riduzione della funzione di perdita già nelle prime epoche, seguita da un andamento stabile e costantemente decrescente. Parallelamente, l'accuratezza del training cresce fino a superare il 98%, mentre l'accuratezza di validazione si mantiene su valori molto alti (circa 98%), con uno scarto minimo rispetto alla curva di training. Questo comportamento suggerisce che il modello non è affetto da overfitting marcato, riuscendo a mantenere prestazioni simili anche su dati non visti in fase di addestramento.

La matrice di confusione (Figura 5.20) conferma le prestazioni osservate. Su un totale di oltre 14.000 esempi di test, il modello classifica correttamente 13.082 istanze, commettendo solo 96 falsi positivi (articoli falsi etichettati come veri) e 130 falsi negativi (articoli veri classificati come falsi). Ciò si traduce in valori elevati di *precision* e *recall* per entrambe le classi, con una distribuzione degli errori bilanciata e quantitativamente trascurabile rispetto al volume complessivo di campioni. Nel complesso, i risultati dimostrano l'efficacia di BERT nell'attività di

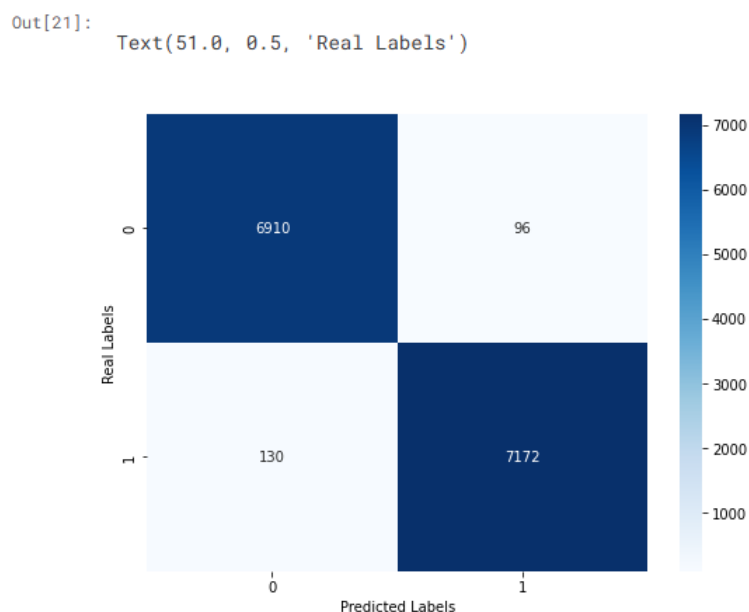


Figura 5.20: Caption

classificazione delle notizie. Le metriche ottenute e la bassa incidenza di errori confermano l'idoneità del modello per applicazioni reali di rilevamento automatico della disinformazione.

Codici Python

Codici R

Bibliografia

- [Dörre et al., 1999] Dörre, J., Gerstl, P., and Seiffert, R. (1999). Text mining: Finding nuggets in mountains of textual data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 398–401, San Diego, CA. ACM Press.
- [Feinerer et al., 2008] Feinerer, I., Hornik, K., and Meyer, D. (2008). Text mining infrastructure in R. *Journal of Statistical Software*, 25(5):1–54.
- [Lane et al., 2019] Lane, H., Howard, C., and Hapke, H. (2019). *Natural Language Processing in Action: Understanding, Analyzing, and Generating Text with Python*. Manning Publications, Shelter Island, NY.
- [Lenci, 2018] Lenci, A. (2018). Distributional models of word meaning. *Annual Review of Linguistics*, 4:151–171.
- [Manning and Schütze, 2002] Manning, C. D. and Schütze, H. (2002). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.
- [Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.
- [Mikolov and Le, 2014] Mikolov, T. and Le, Q. V. (2014). Distributed representations of sentences and documents.

-
- [Miner et al., 2012] Miner, G., IV, J. E., and Hill, T. (2012). *Practical Text Mining and Statistical Analysis for Non-Structured Text Data Applications*. Academic Press, Waltham, MA.
- [Mohammad and Turney, 2013] Mohammad, S. M. and Turney, P. D. (2013). Nrc emotion lexicon. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval-2013)*, pages 321–327, Atlanta, Georgia, USA. National Research Council Canada, Association for Computational Linguistics.
- [Plutchik, 1980] Plutchik, R. (1980). *Emotion: A Psychoevolutionary Synthesis*. Harper & Row, New York.
- [Puri, 2025] Puri, P. (2025). Ai vs human content detection - 1000+ records (2025). <https://www.kaggle.com/datasets/pratyushpuri/ai-vs-human-content-detection-1000-record-in-2025>.
- [Research, 2020] Research, E. (2020). *The Seven Practice Areas of Text Analytics*. Elder Research. Figura 2.1: Venn diagram of text mining and related fields.
- [Salton et al., 1994] Salton, G., Buckley, C., and Singhal, A. (1994). Pivoted document length normalization. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, Heidelberg, Germany. Springer.
- [Shu et al., 2017] Shu, K., Sliva, A., Wang, S., Tang, J., and Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.
- [Sutton and McCallum, 2012] Sutton, C. and McCallum, A. (2012). An introduction to conditional random fields for relational learning. In Getoor, L. and Taskar, B., editors, *Introduction to Statistical Relational Learning*, pages 93–128. MIT Press.
- [Ted, 2017] Ted, K. (2017). *Text Mining in Practice with R*. Wiley.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, , and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, volume 30.