RELAZIONE PROGETTO NATURAL LANGUAGE PROCESSING

Classificazione Automatica di Articoli Giornalistici del “The Guardian“

Studenti: Daniele Careri, matricola n°0771184

Anno Accademico: 2025/2026

---

1. SINTESI DEGLI OBIETTIVI

L’obiettivo di questo progetto è lo sviluppo e il confronto di sistemi di classificazione automatica di testi, nello specifico applicati ad articoli giornalistici estratti dal “The Guardian”.

Il progetto mira a:

1. Costruire un corpus bilanciato estraendo articoli dalla piattaforma *The Guardian*.
2. Effettuare una pipeline completa di preprocessing (tokenizzazione, lemmatizzazione e normalizzazione).
3. Addestrare modelli predittivi capaci di assegnare una delle seguenti categorie ad un articolo mai visto: *World, Sport, Food, Music, Film, Art & Design*.
4. Confrontare le performance di due diverse tecniche di vettorizzazione del testo (**Bag of Words** e **TF-IDF**) e due tecniche di classificazione (**Multinomial Naive Bayes** e **Multinomial Logistic Regression**), analizzando i risultati tramite metriche standard.

---

2. RISORSE UTILIZZATE

Per lo sviluppo del progetto sono state utilizzate risorse open-source e dataset costruiti manualmente.

- **Sorgente Dati:** La creazione del Corpus è avvenuta estraendo articoli dal “The Guardian” usando l’api ufficiale “**The Guardian Open Platform API**”. La scelta è ricaduta su questa fonte per l'alta qualità editoriale del testo e la sua natura open source, inoltre a ciò anche la disponibilità di metadati affidabili come autore, data, categoria, ecc…

- **Librerie Python:**
  - theguardian-api-python / requests: Utilizzate per l'interazione con l'API del The Guardian e il reperimento dei dati. Requests gestisce le chiamate HTTP, mentre il wrapper specifico facilita il download degli articoli in formato JSON.

  - spaCy (usando i modelli en_core_web_sm / en_core_web_trf): Selezionata rispetto a nltk per la migliore accuratezza nelle operazioni di pre-processing (tokenizzazione, POS tagging e lemmatizzazione).

  - regex: Usata per effettuare una pulizia del testo personalizzata, prima della fase di pre-processing.

  - pandas & numpy : Utilizzato per la manipolazione dei dati e la gestione di Data Frame e array multidimensionali.

  - scikit-learn: Impiegata per la fase di Feature Extraction (vettorizzazione del testo), l'implementazione dei classificatori (Naive Bayes, Logistic Regression) e le metriche di valutazione delle performance.

  - matplotlib : Utilizzata per la generazione di grafici e plot necessari per l'analisi dei risultati ottenuti dai modelli.

  - pickle & joblib: Utilizzate per il salvataggio dei dataset pre-processati e dei modelli addestrati.

  - pyarrow & scipy: Dipendenze necessarie per il funzionamento di pandas e numpy.

  - pyTorch: Usato per diminuire i tempi di calcolo, ottimizzando sia le fasi di training che le operazioni di preprocessing, sfruttando l'accelerazione hardware su GPU.

3. SVILUPPO DEL PROGETTO

Lo sviluppo è stato articolato in tre fasi principali: Data Collection, Preprocessing e Training.

Data Collection: Ho implementato uno script per interrogare l'API del Guardian, in particolare scaricando **10.000 articoli** per ciascuna delle seguenti categorie: world, sport, food, music, film, art and design. Per gestire i limiti di richieste dell'API e i timeout, ho implementato un meccanismo di sleep tra le chiamate e un salvataggio incrementale. Il corpus finale conta **60.000 articoli** ognuno con i seguenti metadati: Article Title, Section (category), Authors, Key Word, Text, Language, Number of Word, Url, Date, Article Id. È possibile scaricare il corpus al seguente link.

Es. di dataframe:

Article Title	Section	Authors	Key Word	Text
Likeness of restored angel to Giorgia Meloni triggers investigations in Rome	World news	['Ashifa Kassam']	['Italy' 'Giorgia Meloni' 'Christianity' 'Religion' 'Art' 'Art and design' 'Europe' 'World news']	Italy's culture minister and the diocese of Rome have launched investing…

Language	Number of Word	Url	Date	Article Id
en	692	https://www.theguardian.com/p/x49z93	2026-02-01	cTRKD2YZ

Preprocessing: Ogni articolo è stato processato utilizzando la pipeline di spaCy e regex. Per ogni testo i passi di pre-processing sono:

1. **Pulizia del testo** attraverso l'uso di regex, in particolare:
    a. sono stati sostituiti gli indirizzi http con un token _URL_.
    b. sono state sostituite le email user@example.com con _EMAIL_.
    c. sono stati rimossi spazi avanzati.
2. **Tokenizzazione**.
3. Applicato il **POS Tagging** per filtrare punteggiatura e simboli non rilevanti e migliorare la tokenizzazione.
4. **Lemmatizzazione** per ridurre la dimensionalità del vocabolario (es. "running" -> "run").
5. Eliminazione delle **stopword**.
6. Sostituito i lemmi dei seguenti tag con gli stessi tag (**normalizzazione**/**astrazione**):
    a. `["MONEY", "PERCENT", "QUANTITY", "ORDINAL", "CARDINAL"] -> TAG.lower()`
    b. `["DATE", "TIME"] -> "_TIME_"`
    c. `["PERSON", "ORG", "GPE", "EVENT", "FAC", "WORK_OF_ART"] -> TAG.lower()`

7. Il risultato è stato salvato in un DataFrame Pandas contenente il testo originale, il testo ricostruito con i lemmi, la lista dei token e la lista dei lemmi, per permettere la comparazione tra testo grezzo e pre-processato.

Es. di dataframe pre-processato:

…	Cleaned Text	Token
…	“__ordinal__ …”	[[“first”, “__ordinal__”, “ADJ”, “ORDINAL”],...]

Feature Extraction e Training: Il dataset è stato diviso in Training Set (70%) e Test Set (30%). Sono state generate due matrici di feature basate sia sui lemmi sia sui token:

1. Bag of Words (BoW): Utilizzando CountVectorizer, che conta le occorrenze delle parole.
2. TF-IDF: Utilizzando TfidfVectorizer, che pesa le parole in base alla loro frequenza nel documento e la loro rarità nel corpus (TermFrequency−InverseDocumentFrequency).

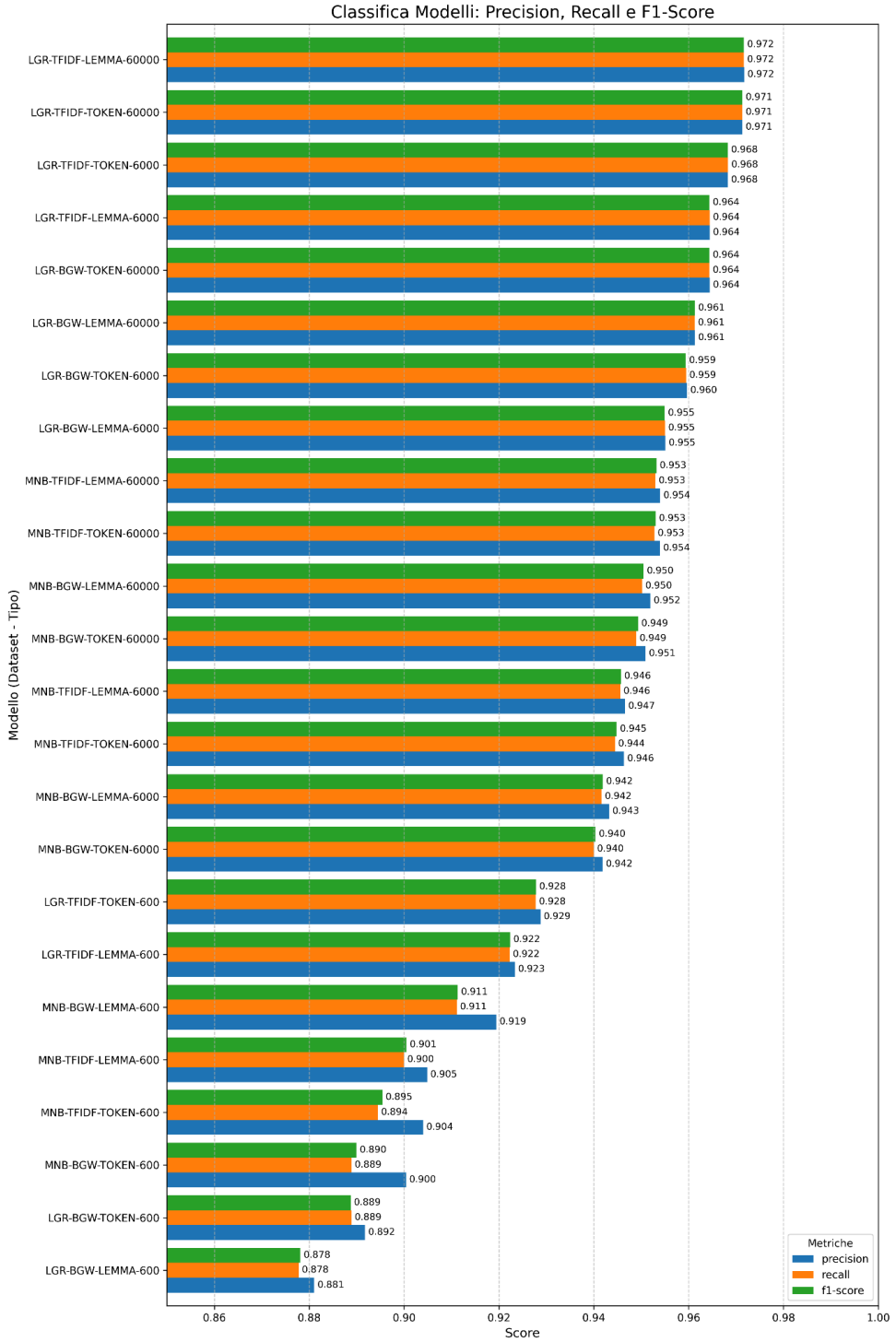
Su queste vettorizzazioni sono stati addestrati 24 modelli usando 600, 6.000, 60.000 token/lemmi:

- Naive Bayes su BoW Lemma (600, 6.000, 60.000)
- Naive Bayes su BoW/ Token (600, 6.000, 60.000)
- Naive Bayes su TF-IDF Lemma (600, 6.000, 60.000)
- Naive Bayes su TF-IDF Token (600, 6.000, 60.000)
- Logistic Regression su BoW Lemma (600, 6.000, 60.000)
- Logistic Regression su BoW Token (600, 6.000, 60.000)
- Logistic Regression su TF-IDF Lemma (600, 6.000, 60.000)
- Logistic Regression su BoW Token (600, 6.000, 60.000)

4. ANALISI E CONCLUSIONI

I modelli sono stati valutati sul Test Set (articoli mai visti in fase di addestramento). Per la valutazione, mi sono focalizzati sul **Macro-Average** di Precision, Recall e F1-Score, per garantire che tutte le classi abbiano lo stesso peso nella valutazione finale.

I risultati sono riportati nella seguente figura:



Analisi delle Prestazioni e "Best Performer":

Osservando le performance, emerge una parità tra i primi tre modelli, i quali ottengono punteggi di accuratezza (accuracy) molto elevati e quasi identici. La configurazione migliore risulta essere LGR-TFIDF-LEMMA-60000 (Logistic Regression, TF-IDF, Lemmatization, 60.000 feature).

Confronto tra Logistic Regression e Naive Bayes:

I dati mostrano un chiaro dominio della Logistic Regression (LGR), che occupa tutte le prime 8 posizioni della classifica. Questo conferma che, in presenza di dataset bilanciati e vocabolari sufficientemente ampi, la Logistic Regression tende a superare il Naive Bayes. Inoltre il Naive Bayes, pur essendo efficiente, soffre maggiormente l'assunzione di indipendenza tra le feature rispetto alla capacità della LGR di gestire relazioni più complesse anche quando sono presenti molte feature.

Impatto del Numero di Feature (Dimensionalità):

L'analisi del numero di mostra che l'uso di sole 600 feature è chiaramente insufficiente poiché il modello va in underfitting, ciò causato dal vocabolario che è troppo limitato per cogliere le differenze necessarie a distinguere le categorie di articoli. Si osserva invece che passare da 600 a 6.000 feature garantisce un incremento enorme delle prestazioni, mentre passare da 6.000 a 60.000 feature offre un incremento minore. L'aggiunta di ulteriori 54.000 parole porta solo un miglioramento marginale, confermando che le 6.000 parole più frequenti contengono la maggior parte dell'informazione.

Rappresentazione del Testo “TF-IDF vs Bag of Words”:

Il confronto tra le tecniche di vettorizzazione conferma la superiorità del TF-IDF rispetto al BOW; infatti notiamo che il modello LGR-TFIDF-600 supera di molto il LGR-BOW-600. Il *Bag of Words* (conteggio semplice) è una tecnica più semplice rispetto al TF-IDF che riesce a ridurre il rumore delle parole comuni (stopwords o parole frequenti non informative) e a valorizzare le parole chiave distintive, compensando la scarsità di feature.

Preprocessing “Lemmatizzazione vs Tokenizzazione”:

L'impatto della normalizzazione del testo risulta marginale in termini di punteggio infatti il divario tra modelli con "Tokenizzazione" e "Lemmatizzazione" è quasi inesistente. Effettuare una lemmatizzazione comporta un costo computazionale aggiuntivo che, in questo contesto, non produce un beneficio prestazionale. Nonostante il punteggio simile, la lemmatizzazione può comunque essere utile per migliorare la generalizzazione del modello, gestendo meglio varianti di nomi, organizzazioni, numeri e date che potrebbero non apparire nel training set.

Miglioramenti Futuri: Per migliorare ulteriormente il modello, si potrebbe integrare l'uso di n-grams (bigrammi e trigrammi) o utilizzare i Word Embeddings (es. Word2Vec o BERT) per catturare meglio il contesto semantico rispetto a BoW o TF-IDF.