UNIVERSITÀ DEGLI STUDI DI
MILANO-BICOCCA

TEXT MINING AND SEARCH

FINAL PROJECT

Classificazione di notizie Real o Fake

Authors:
Davide Porcellini - 816586
d.porcellini2@campus.unimib.it

Simone D'Amico - 850369
s.damico4@campus.unimib.it

1 INTRODUZIONE

Al giorno d'oggi, il mondo dell'informazione sta cambiando rispetto alle sue origini, i media tradizionali quali carta stampata, tv e radio stanno lentamente cedendo il passo all'avvento di internet. In un mondo sempre più interconnesso le notizie girano ad un'elevata velocità e senza alcun tipo di controllo, queste condizioni sollevano alcuni problemi. Come riconoscere se una notizia sia vera o falsa? Come posso riconoscere l'attendibilità di una fonte? Come faccio ad essere sicuro che chi è all'altro lato dello schermo sia in buonafede? In che misura esiste il vero e il falso in una notizia? Sono due elementi che possono coesistere? Queste domande sono accentuate se si pensa al fatto che ormai internet ed i social network sono disponibili alla stragrande maggioranza della popolazione, anche a quelle fasce della popolazione magari molto giovane o magari poco istruita che potrebbero riscontrare delle difficoltà maggiori nel riuscire a rispondere a queste domande quando si trovano davanti a delle notizie. Lo scopo del nostro progetto è di riuscire a determinare, grazie a delle tecniche di text mining e dei modelli se, dato il testo di una notizia, questa sia vera oppure falsa. Per fare ciò, oltre ad aver addestrato dei modelli di classificazione, abbiamo realizzato un'interfaccia grafica che permette all'utente di inserire una nuova notizia e ottenere una sintetica rappresentazione del testo e se il modello migliore che abbiamo ottenuto la classifica come *True* o *Fake*.

2 PREPROCESSING

Il Dataset che abbiamo utilizzato riguarda delle notizie in lingua inglese trovate online e comprende due categorie di notizie, notizie catalogate come *True* quindi vere, attendibili e notizie Fake ovvero notizie inventate o mistificate da parte dell'autore. In principio abbiamo raccolto i dati da un unico dataset di Kaggle[1] ma ci siamo accorti che, probabilmente dovuto al fatto che gli articoli provenivano dagli stessi siti, bastava utilizzare poche parole per raggiungere un'accuracy molto vicina al 100%. Abbiamo quindi deciso di integrare, sempre da Kaggle, un secondo dataset[2], in modo da ottenere una maggiore varietà di fonti e "complicare" il lavoro dei classificatori per renderli più efficaci su notizie di provenienza diversa. Inizialmente i dati che abbiamo raccolto contavano più di 50000 notizie diverse ma per esigenze computazionali abbiamo dovuto ridurre queste dimensioni. Dopo una breve operazione di pulizia dei dati in cui abbiamo eliminato le notizie il cui testo era vuoto o quelle che risultavano essere dei duplicati, il Dataset finale si compone di 10000 notizie uniche ottenute tramite campionamento dei due dataset e mantenendo una buona distribuzione della variabile target dicotomica, le cui classi sono 'True' e 'Fake', vicina al 50%. Le altre variabili che abbiamo selezionato sono due: il testo originale dell'articolo ed il risultato del testo originale dopo avergli applicato delle tecniche di preprocessing per prepararlo all'analisi.
Le attività di preprocessing applicate alle varie notizie sono state:

- Identificazione della lingua delle notizie per separare i testi inglesi da quelli di altre lingue, in particolare la libreria Python utilizzata rilevava la lingua inglese anche di testi che contenevano caratteri non solo di quella lingua, questo ha reso necessari un ulteriore passaggio di pulizia.
- Eliminazione di caratteri non appartenenti all'alfabeto inglese dai testi, quindi derivanti da altre lingue come il cinese o il cirillico. Dopo l'individuazione della lingua come inglese, in alcuni testi erano presenti caratteri di altre lingue che sono stati eliminati con l'uso di espressioni regolari.
- Eliminazione dei numeri, della punteggiatura e altri segni particolari come le emoji presenti in alcuni testi.
- Eliminazione delle *stop word* inglesi come gli articoli o le congiunzioni.
- Lemmatizzazione dei vari token delle parole con utilizzo di *POS* per aumentarne l'efficacia.

Il dataset finale contiene un campione totale di 10.000 righe provenienti, in egual misura, dai due dataset originari dopo la pulizia. Si mostrano adesso alcune statistiche descrittive del dataset finale (rif. Figura 1), in particolare la distribuzione dei valori della variabile target all'interno dei dati. In figura 2 si mostrano alcuni esempi di wordcloud ottenute dai testi preprocessati di notizie catalogate sia come *True* (rif. Figura 2a) che come *Fake* (rif. Figura 2b) e si mostra infine la distribuzione del numero di parole utilizzate negli articoli, sempre dopo aver attuato le tecniche di pulizia, in relazione alle due modalità del target (rif. Figura 3).

[1] https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset
[2] https://www.kaggle.com/saratchendra/fake-news

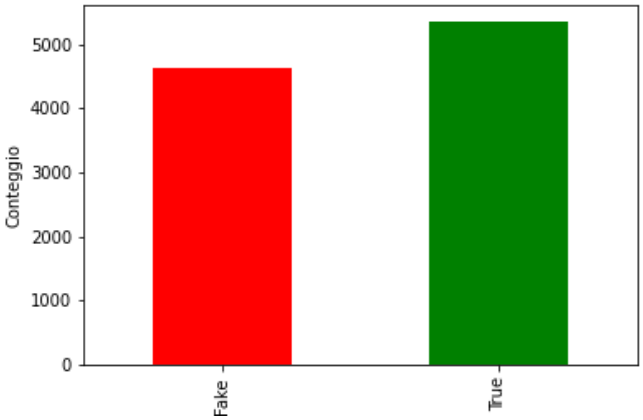


Figure 1: Distribuzione della variabile target

(a) Word cloud notizia vera

(b) Word cloud notizia falsa

Figure 2: Word cloud

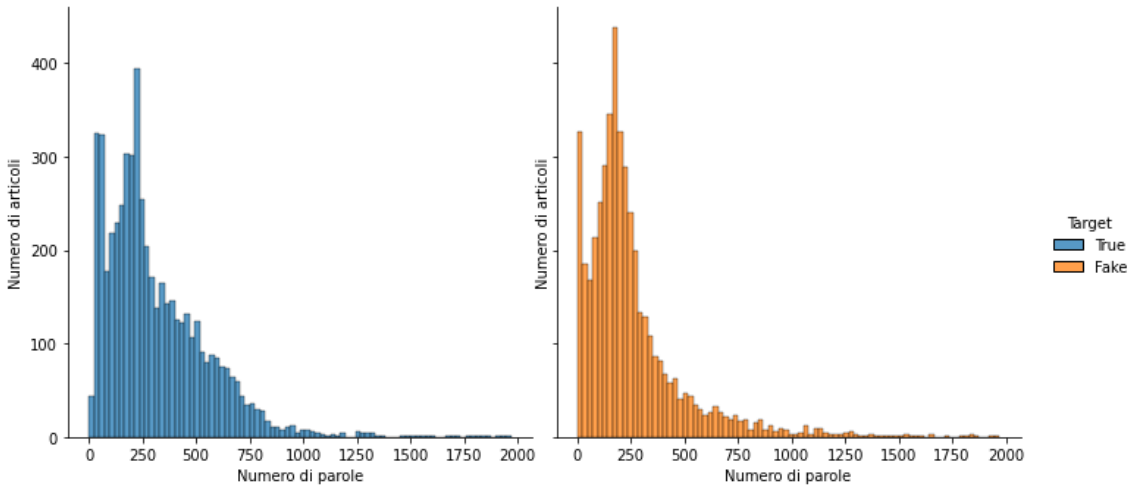


Figure 3: Distribuzione numero parole per articolo rispetto al target

Dal grafico (rif. Figura 3) notiamo che le distribuzione delle notizie *Fake*, per quanto riguarda il numero di parole utilizzate, sembra essere orientata verso un valore minore, tra le 0 e le 250 parole, mentre la distribuzione delle

notizie *True* risulta essere leggermente più omogenea e presenta valori più elevati nel numero di articoli che contengono almeno 250 parole rispetto alle notizie *Fake*. Per una rappresentazione ottimale del grafico sono stati presi in considerazione solamente gli articoli che contenevano meno di 2000 parole.
Dal dataset si sono poi ottenuti il train set con il 67% delle osservazioni iniziali e il test set con il rimanente 33%.

3 RAPPRESENTAZIONE DEL TESTO

Il testo delle notizie è stato rappresentato in tre modi diversi: tramite il conteggio delle parole, con l'uso del *Tf-Idf* e utilizzando i *Word Embeddings*. Nel primo caso, per ogni parola del corpus si è calcolato il numero di occorrenze nei vari testi. Si è ottenuta una matrice che aveva tante colonne quante sono le parole presenti, in cui le righe rappresentavano i documenti e nella cella (d,t) è presente il numero di occorrenze del termine t nel documento d.
La funzione *Tf-Idf* (*term frequency–inverse document frequency*) è una funzione utilizzata per misurare l'importanza di un termine rispetto ad un documento o ad una collezione di documenti. La funzione è composta da due fattori: il primo fattore della funzione ($Tf_{t,d}$) è il numero di volte in cui il termine t si presenta nel documento d. Questo valore può essere normalizzato tramite le occorrenze massime nel documento:

$$Tf_{t,d} = \frac{tf_{t,d}}{\max_{t_i \in d} tf_{t_i,d}} \tag{1}$$

Il secondo fattore della funzione (*idf*) valuta l'importanza del termine in tutta la collezione, cioè il numero di documenti che contengono il termine t:

$$Idf_t = log\left(\frac{|\mathcal{D}|}{df_t}\right) \tag{2}$$

dove $|\mathcal{D}|$ è la cardinalità della collezione di documenti e df_t è la frequenza del termine tra tutti i documenti, cioè il numero di documenti che contengono il termine t. Il Tf-Idf è dato da: $Tf - Idf_{t,d} = Tf_{t,d} \times Idf_t$.
Parametro importante per il calcolo sia della frequenza del termine sia del suo Tf-Idf è la dimensione degli n-grammi considerati che può influire sulle performance dei modelli. Per capire quale dimensione di n-grammi scegliere, si è eseguita una grid search e per entrambe le rappresentazioni si sono considerati sei diverse misure di n-grammi: i solo unigrammi, i solo bigrammi, i soli trigrammi, unigrammi e bigrammi, bigrammi e trigrammi e infine tutti e tre insieme. Si sono utilizzate queste sei rappresentazioni per addestrare i diversi modelli.
Un secondo approccio considerato consisteva nell'utilizzo di modelli di Word Embeddings per rappresentare le varie parole, in particolare il modello utilizzato è stato Doc2Vec che permette di rappresentare interi testi in forma vettoriale. Si sono utilizzate due strategie per l'uso dei Word Embeddings cioè si è usato un modello già addestrato da cui inferire i vettori dei nostri test e, in alternativa, si è addestrato direttamente il modello sulle notizie del test set. Il modello già trainato è disponibile su github[3], è un modello addestrato partendo da 25M di documenti e 9B di token [1] ottenuti da una serie di articoli della *Associated Press English* dal 2009 al 2015. L'addestramento è avvenuto tramite l'algoritmo DBOW, l'equivalente dell'approccio Skip-gram in Word2Vec e i vettori sono formati da 300 componenti. Il modello addestrato ad hoc ha le stesse caratteristiche del modello già trainato, stesso algoritmo e dimensioni dei vettori e si è scelto di fare l'addestramento su 50 epoche.

4 MODELLI DI CLASSIFICAZIONE

Abbiamo addestrato diversi modelli per poter fare un confronto ed ottenere il miglior modello possibile. Si è usata la libreria Python sklearn che mette a disposizione una serie di modelli, in particolare abbiamo utilizzato:

- Logistic Regression (LR): modello di regressione non lineare utilizzato quando la variabile dipendente è di tipo dicotomico. Stabilisce la probabilità con cui un'osservazione può generare uno o l'altro valore della variabile dipendente, si adatta particolarmente bene al nostro scopo di prevedere uno tra i due valori della variabile target.

[3] https://github.com/jhlau/doc2vec

- Support Vector Machine (SVM): algoritmo che rappresenta le osservazioni del dataset in uno spazio *n*-dimensionale. La soluzione consiste nel trovare gli iperpiani che separino al meglio gli elementi che appartengono alle classi diverse.
- Random Forest (RF): modello con il quale si definiscono una serie di decision tree. La classe predetta è quella maggiormente predetta dai vari alberi decisionali. In particolare la Random Forest utilizzata contiene 100 alberi.
- Multi-Layer Perceptron (MLP): tipo di rete neurale, per la nostra classificazione si è definita un'architettura a quattro livelli, oltre al livello di input e quello di output sono presenti altri due livelli nascosti, ognuno formato da 10 neuroni. La funzione di attivazione è la funzione *relu*, la funzione di unità lineare rettificata che restituisce $f(x) = max(0,x)$. Come *solver* si è usato *Adam*, un ottimizzatore basato su gradiente stocastico che risulta essere più robusto su set di dati relativamente grandi come nel nostro caso.

Per ognuno di questi modelli abbiamo utilizzato come dati di addestramento sia la rappresentazione tramite la frequenza delle parole che quella tramite il Tf-Idf per i quali abbiamo adottato il medesimo approccio:

- tecnica Grid Search per trovare il numero ideale di *n*-grammi da utilizzare, ognuno dei precedenti modelli è stato addestrato sulle sei diverse lunghezze degli *n*-grammi
- confronto tra modelli in cui viene prima utilizzato il testo preprocessato e successivamente gli stessi modelli in cui sono state rimosse le parole più frequenti e quelle meno frequenti;

Gli stessi modelli sono stati utilizzati anche con i *Word Embeddings* generati sia dal modello già addestrato e sia dal modello ad hoc.
Per la valutazione si è usata la metrica dell'*accuracy* utilizzando i valori predetti sui testi nel test set e confrontandoli con il loro reale valore per la variabile target, l'accuracy è definita come:

$$\alpha = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Dove TP rappresenta i veri positivi, TN i veri negativi, FP i falsi positivi e FN i falsi negativi.

4.1 RAPPRESENTAZIONE CON FREQUENZA E TF-IDF

Per individuare il modello migliore nelle due rappresentazioni si è eseguita una grid search sulle varie rappresentazioni, sia con la frequenza sia con il Tf-Idf, create con una diversa dimensione degli *n*-grammi: i soli unigrammi (1,1), i soli bigrammi (2,2), i soli trigrammi (3,3), unigrammi e bigrammi insieme(1,2), bigrammi e trigrammi insieme (2,3) e tutti e tre gli *n*-grammi (1,3). Nelle tabelle 1 e 2 sono mostrate le *accuracy* ottenute.

Table 1: Accuracy per i modelli basati sulla frequenza

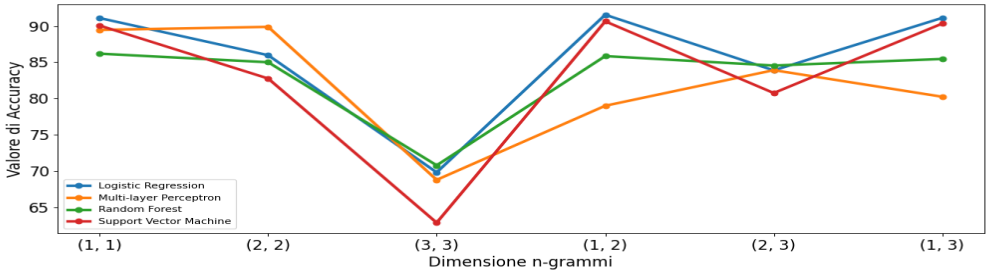
n-grammi	LR	MLP	RF	SVM
(1,1)	91.09%	89.45%	86.18%	90.06%
(1,2)	**91.55%**	79.00%	85.85%	90.64%
(1,3)	91.12%	80.24%	85.45%	90.33%
(2,2)	85.97%	89.88%	85.00%	82.76%
(2,3)	83.89%	83.91%	84.55%	80.76%
(3,3)	69.82%	68.79%	70.79%	62.88%

Table 2: Accuracy per i modelli basati sul Tf-Idf

n-grammi	LR	MLP	RF	SVM
(1,1)	90.12%	91.60%	86.39%	**92.06%**
(1,2)	88.24%	90.52%	86.21%	91.12%
(1,3)	86.82%	88.39%	86.39%	89.67%
(2,2)	85.30%	87.09%	85.36%	89.21%
(2,3)	81.73%	86.70%	84.79%	88.03%
(3,3)	70.94%	83.70%	72.06%	80.33%

Si è visto che c'è una significativa diminuzione dell'*accuracy* se si usano i soli trigrammi, in figura 4 si mostra l'andamento dell'*accuracy* all'aumentare della dimensione degli *n*-grammi usati.

(a) Andamento dell'accuracy all'aumentare della dimensione degli n-grammi nella rappresentazione del conteggio delle parole.



(b) Andamento dell'accuracy all'aumentare della dimensione degli n-grammi nella rappresentazione Tf-Idf.

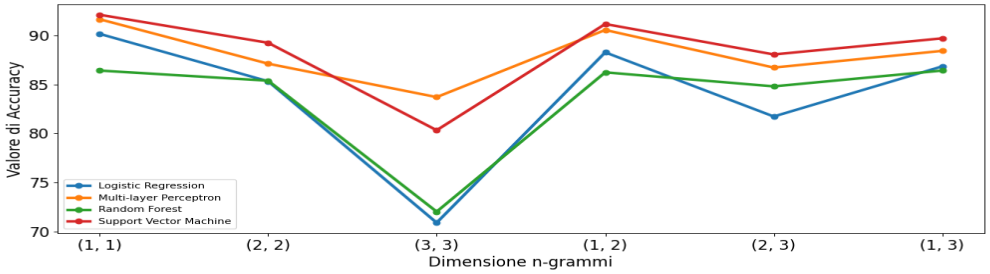


Figure 4: Andamento dell'accuracy nelle due rappresentazioni

4.1.1 FILTRAGGIO DELLE PAROLE

I modelli migliori, in termini di *accuracy*, risultano essere la regressione logistica con unigrammi e bigrammi nel caso della frequenza e il SVM con i soli unigrammi nel caso del Tf-Idf. Ci si è concentrati su questi due modelli e si è cercato di capire se, filtrando le parole in base alla loro frequenza o al loro valore di Tf-Idf, si potesse migliorare l'accuracy dei modelli. Le parole più o meno frequenti nei testi sono state individuate tramite l'utilizzo del range interquartile.

$$IQR = Q3 - Q1 \tag{4}$$

$$LowerBound = Q1 - 1.5 * IQR \tag{5}$$

$$UpperBound = Q3 + 1.5 * IQR \tag{6}$$

Sono state rimosse le parole con una frequenza minore del *Lower Buond* e maggiore del *Upper Buond*. Successivamente sono state calcolate nuovamente le rappresentazioni testuali considerando solo le parole all'interno dell'intervallo definito, poi si sono addestrati i due modelli migliori per queste due nuove rappresentazioni. Per la rappresentazione con la frequenza, il nuovo modello di regressione logistica raggiunge un accuracy del 68.15% mentre il modello SVM nella rappresentazione Tf-Idf ha un'accuracy del 70.85%. La valutazione dei modelli è quindi molto inferiore e si è deciso di non considerare queste varianti nella scelta del modello migliore.

4.2 MODELLI CON IL DOC2VEC

I quattro diversi modelli di classificazione sono stati usati anche con le rappresentazioni Doc2Vec del testo. Nella tabella 3 sono mostrate le varie *accuracy*, il modello migliore risulta essere il Multi-Layer Perceptron applicato al modello addestrato direttamente sul train set.

Table 3: Accuracy per i modelli basati sul Tf-Idf

Modello Doc2Vec	LR	SVM	RF	MLP
Doc2Vec ad hoc	85.88%	85.88%	82.79%	**88.18%**
Doc2Vec pre-train	83.60%	83.91%	80.33%	81.24%

5 INTERFACCIA UTENTE INTERATTIVA

Un ulteriore obiettivo del progetto era quello di fornire all'utente uno strumento interattivo che permettesse di classificare delle notizie di interesse per l'utente stesso. A tale scopo si è implementata anche una piccola interfaccia grafica, tramite *widgets* di *Jupyter Notebook*; l'utente può inserire, nell'apposita area di testo, la notizia che vuole classificare e può selezionare diverse opzioni per l'output che rendono la sua interazione con il sistema più utile e coinvolgente. Il modello usato per la classificazione è quello migliore tra tutte le varianti analizzare, cioè il SVM con rappresentazione Tf-Idf. In figura 5 si mostra il risultato dalla valutazione di una notizia *Fake* con il

Figure 5: Valutazione di una notizia e output dell'interfaccia.

relativo output. Il sistema fornisce diverse funzionalità tra cui la possibilità di avere una *soft classification*, cioè assegnare una probabilità della notizia di essere *True* o *Fake*, in questo modo l'utente può distinguere tra diverse notizie con un diverso grado di verità o falsità. Altra funzionalità fornita da questa interfaccia c'è un analisi di *Explainable AI*, una disciplina che definisce metodi e tecniche che rendono maggiormente comprensibili gli algoritmi black box, tipici del machine learning e della AI, agli esserei umani. Tra i vari strumenti di EX-AI, si utilizza *lime*, un algoritmo di interpretabilità locale, in particolare si spiega il risultato di un algoritmo di machine learning applicato ad una specifica osservazione considerando le sue osservazioni più vicine. Nel nostro caso specifico, viene fornita una spiegazione di come parole presenti in una notizia influiscano sul risultato della sua classificazione come *True* o *Fake*.

6 CONCLUSIONI

6.1 Miglior modello

Il modello migliore che abbiamo ottenuto, ovvero quello che risulta più affidabile nel predire se una nuova notizia che noi gli chiederemo di catalogare sia *True* o *Fake* risulta essere:

- per il metodo della rappresentazione tramite la frequenza la Regressione Logistica utilizzando unigrammi e bigrammi con un'*accuracy* del **91.55%**;
- per il metodo della rappresentazione tramite il Tf-Idf è SVM utilizzando unigrammi con un'*accuracy* del **92.06%**;
- utilizzando il metodo del Word embedding il modello migliore risulta essere MLP con un'*accuracy* pari a **88.18%**.

Per il nostro studio non abbiamo motivo di preferire un tipo di rappresentazione rispetto ad un altro quindi consideriamo il modello SVM il nostro miglior modello poiché è quello che presenta un valore puntuale di *accuracy* maggiore.

6.2 Sviluppi futuri

Per mancanza di mezzi e per esigenze computazionali abbiamo allenato il nostro modello Multi-Layer Perceptron con dei parametri che possono sicuramente essere migliorati, come ad esempio l'architettura che può essere espansa con ulteriori hidden layer o un numero maggiore di neuroni; questo potrebbe provocare un aumento della precisione di classificazione di questi due modelli. Disponendo di mezzi più potenti potremmo anche non rinunciare a tutte le notizie che siamo stati costretti a scartare non avendo la possibilità di processare un così grande quantitativo di dati che ci avrebbe consentito di avere un train set più ampio su cui addestrare i nostri modelli, soprattutto i modelli di Word Embeddings. Un altro possibile sviluppo futuro riguarda l'implementare la capacità di riconoscere testi di lingue differenti all'inglese in modo da poter rendere utilizzabile questo strumento al maggior numero di persone possibile. Provare diversi modelli di classificazioni o diverse tecniche di filtraggio delle parole poco frequenti o troppo frequenti per favorire il lavoro dei modelli.

REFERENCES

[1] Lau, J. H., & Baldwin, T. (2016). *An empirical evaluation of doc2vec with practical insights into document embedding generation.* arXiv.org.