Università degli studi di Milano-Bicocca

Text Mining & Search

Amazon Reviews Classification

Authors:
Licciardello Matteo - 799368 - m.licciardello@campus.unimib.it
Mariani Paolo - 800307 - p.mariani20@campus.unimib.it

Settembre 2020

Abstract

L'estrazione di informazioni a partire da dati testuali rappresenta una delle tecniche più utili per comprendere ciò che un autore vuole comunicare ad altri individui; oggi, queste operazioni possono essere svolte tramite il supporto della tecnologia, la quale permette l'analisi di grandi quantità di dati in tempi brevi per estrarre informazioni utili tramite l'utilizzo di algoritmi complessi. Il caso preso in esame consiste in un compito di *Text Classification* applicato ad un insieme di recensioni riguardanti prodotti venduti sull'e-commerce statunitense di Amazon, al fine di prevedere la categoria di appartenenza di ciascuna recensione.

1 Introduzione

L'Amazon e-commerce è un marketplace digitale in cui avvengono miliardi di scambi di prodotti, ogni giorno in tutto il globo, che agevola l'acquisto di prodotti da parte degli utenti in maniera comoda e veloce. Tale piattaforma è fornita dall'azienda statunitense *Amazon.com, Inc.*[1], fondata nel 1994 sotto il nome di *Cadabra* da Jeffrey Bezos e, ad oggi, rappresenta un colosso multinazionale impegnato negli ambiti di e-commerce, servizi di *cloud computing*, *artificial intelligence* ed *entertainment*. L'azienda fa grande utilizzo dell'applicazione della tecnologia ai dati, a partire dallo *storaging* dei prodotti dell'e-commerce fino ad arrivare all'analisi delle interazioni nelle piattaforme offerte. Nello specifico, con diverse finalità, svolge attività provenienti dal mondo *Text Mining* come *Text Clustering*[2], *Item Recommendation*[3], *Sentiment Analysis*[4] e *Text Classification*[5]. Queste attività sono finalizzate a creare un valore per le parti che interagiscono nella piattaforma e per l'azienda stessa, nello specifico implementando una politica incentrata sui bisogni del cliente. Ad esempio, le attività precedentemente elencate, possono essere utili per individuare prodotti che condividono le stesse caratteristiche ed abilitare una raccomandazione a specifici utenti, o per comprendere quali siano le preferenze dell'utente e il suo grado di soddisfazione per ogni acquisto. Le potenzialità offerte da queste tecniche basate sul *Natural Language Processing* [6] sono innumerevoli: in particolare, nel progetto proposto, ci si concentra sul task di Text Classification svolto a partire dalle recensioni scritte dagli utenti statunitensi sulla piattaforma e-commerce, finalizzato a determinare per ogni recensione quale sia la macro-categoria predeterminata di appartenenza in funzione del tipo di prodotto associato. Tale compito di *Knowledge Organization* risulta essere estremamente utile in un contesto di assegnazione di struttura alla conoscenza, laddove tale conoscenza sia rappresentata da testo non

strutturato. Solo alcune delle classi disponibili dalla fonte dati considerata sono selezionate:

- *Office Products*, ovvero prodotti da ufficio come arredamento, cancelleria e prodotti tecnologici ad uso dedicato all'ufficio
- *Automotive*, che tratta di prodotti dedicati al settore automobilistico per la cura dei veicoli, accessori e gadget
- *Patio, Lawn and Garden*, relativa a tutti i prodotti dedicati al giardino ed all'arredamento per esterni
- *Toys and Games*, inerente a giocattoli e giochi da tavolo

Il task di classificazione viene effettuato sfruttando il servizio gratuito Google Colab, che però possiede dei vincoli di utilizzo pari a 12 ore consecutive. Considerando i vincoli di risorse computazionali si assume che non sia possibile raggiungere la massima efficacia nell'applicazione degli algoritmi ma che si dovrà puntare su algoritmi sufficientemente efficienti per completare la computazione nei tempi previsti. Nello svolgimento del compito presentato l'obiettivo del gruppo è quello di valutare la bontà di classificazione dei modelli rispetto a diverse forme di rappresentazione delle recensioni testuali (TF-IDF, Doc2Vec, SVD applicata alla matrice TF-IDF), con particolare attenzione all'efficienza raggiunta in termini di tempo e risorse utilizzate. Verrà inoltre fornito un commento dei risultati ottenuti ed effettuata la scelta di un classificatore testuale che il gruppo ritiene sufficientemente performante ed applicabile ad un contesto di utilizzo reale.

2 Dataset

La fonte dati da cui sono state prelevate le recensioni utilizzate nel task di classificazione è il dataset Amazon Reviews, composto da quasi 35 milioni di recensioni che gli utenti statunitensi hanno rilasciato, per svariati prodotti, a partire dal 1995. Le categorie considerate sono le già citate *Office Products*, *Automotive*, *Patio, Lawn and Garden*, *Toys and Games*.
Ogni recensione mette a disposizione le seguenti 10 features:

- *asin*: ID prodotto recensito
- *reviewerID*: ID dell'utente che crea la recensione
- *reviewerName*: nome dell'utente

- *reviewTime*: data (DD MM, YYYY) in cui è stata scritta la recensione
- *unixReviewTime*: data di pubblicazione in formato Unix
- *verified*: variabile binaria che indica se Amazon ha verificato l'acquisto del prodotto per l'utente reviewerID
- *overall*: valutazione del prodotto, da 1 a 5 stelle
- *reviewText*: testo della recensione
- *summary*: sintesi della recensione, piccolo titolo della recensione
- *style*: dizionario utile per riconoscere la versione del prodotto (colore, dimensioni, versione, ...)

L'ammontare di recensioni percepito all'interno di queste classi selezionate è pari a 4.909.367, distribuite in maniera sbilanciata, come si può osservare nell'immagine 1:

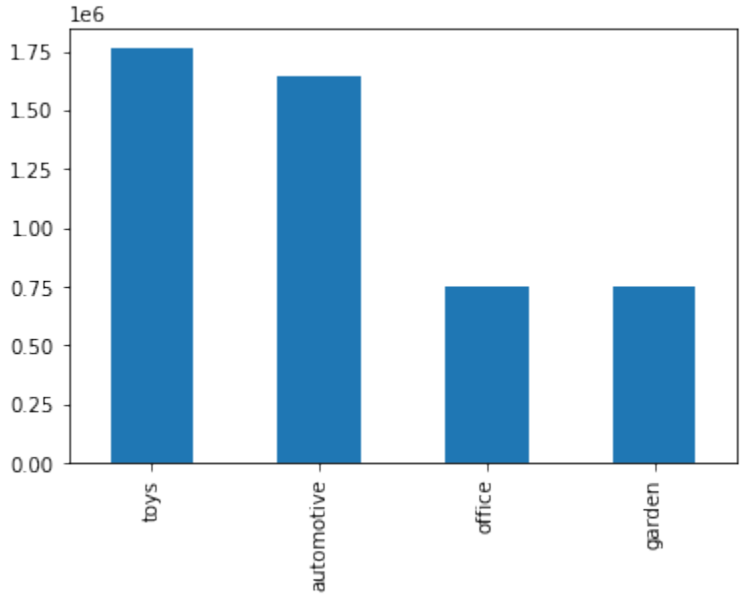

Figure 1: distribuzione delle recensioni rispetto alle classi

Si osserva uno sbilanciamento in favore delle classi *toys* e *automotive* che rappresentano la maggior parte delle recensioni, mentre *office* e *garden* possiedono un numero più ridotto e simile di elementi. La procedura di ottenimento dei file avviene

mediante il download di un file in formato *JSON* per ogni classe-categoria Amazon e la seguente trasformazione del file in una struttura *Dataframe Pandas* [7] per ognuno di essi, da cui si è proceduto ad unire tutti i dataframe intermedi per poter creare un'unica struttura dati denominata "*collection*".

3 Preprocessing

Il dataset, come presentato nella sezione 2, contiene alcuni attributi non informativi per lo scopo del progetto ed una dimensione troppo elevata per poter essere processata con le risorse a disposizione: tale considerazione porterà all'eliminazione (svolta in seguito) di una buona parte di recensioni e features ritenute non utili. In primo luogo è stata effettuata un'analisi esplorativa del dataset la quale ha portato a rivelare la presenza di numerose recensioni contenenti link o immagini che, a loro volta, comportano la presenza di tag *HTML*: tali elementi influiscono enormemente sulla qualità del risultato dell'attività di elaborazione del testo e di creazione di una rappresentazione, per questo motivo è stato rimosso ogni elemento di disturbo dalle 11849 recensioni in cui questi sono stati individuati.

3.1 Riduzione del Dataset

Come già esaminato, il numero delle recensioni è troppo elevato per poter garantire l'ottenimento di risultati in tempi adeguati. Il processo di *sampling* delle recensioni è avvenuto in diversi momenti procedendo, ad ogni tentativo di riduzione del dataset, ad effettuare un'attività di preprocessing e conseguente classificazione per valutare la fattibilità dello svolgimento del compito. Inizialmente sono state eliminate tutte le recensioni duplicate o senza alcun testo, in seguito il numero di recensioni è stato ridotto selezionando per ogni prodotto la recensione più lunga: si considera tale recensione come la più informativa, in quanto contiene un numero di termini più ampio e generalmente più adatto a generare un vocabolario sufficientemente vario. Una seconda motivazione è legata al fatto che molte recensioni non sono sufficientemente informative poiché risultano essere troppo brevi e contenenti termini troppo frequenti nel dominio delle recensioni (es: "Very good item, recommended"). In tale situazione la distribuzione delle classi per la raccolta risulta essere rappresentata dalla figura 2, con un numero di osservazioni pari a 219092.

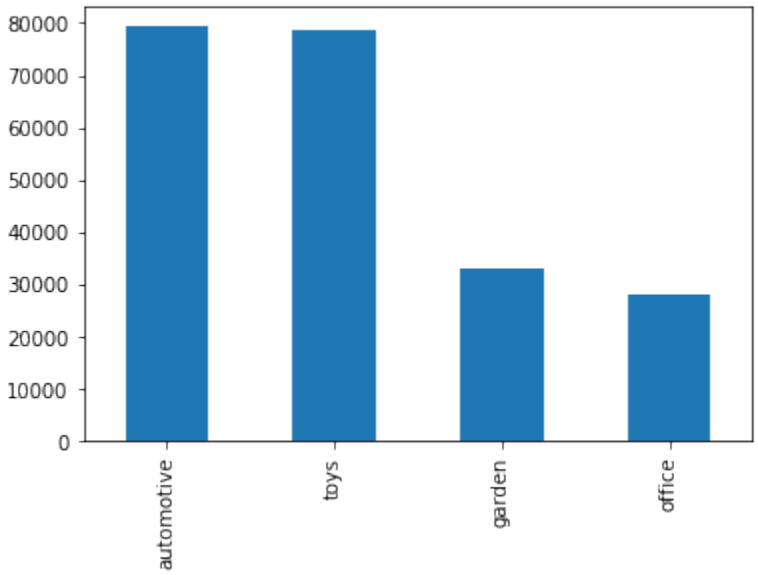

Figure 2: distribuzione delle recensioni rispetto alle classi, dopo il sampling

Come si evince dal barplot in Figura 2, le classi sono ancora affette da un forte sbilanciamento: è quindi stato imposto un bilanciamento delle classi che ha provveduto a selezionare casualmente per ogni classe un numero di recensioni pari alla numerosità della classe meno frequente, in questo caso *office*. In totale si ottiene una collezione di 111860 recensioni, equamente ripartite rispetto alle classi. Le colonne ritenute non informative o utili sono state rimosse, nell'elenco di features mantenute osserviamo: *reviewText* (già elaborato per l'eliminazione dei tag e link), *summary*, *target*, *asin* e *charlen* (lunghezza complessiva della frase prima del preprocessing).

3.2 Preprocessing del testo

Per poter ottenere una rappresentazione formale di una recensione si effettuano delle operazioni di preprocessing del testo che garantiscono l'estrazione di unità informazionali fondamentali, ovvero le parole. Su queste vengono poi applicate alcune fasi complementari di pulizia che permettono di ottenere una base ideale per generare una rappresentazione vettoriale di features di una recensione, secondo il modello *Bag of Words* (BOW)[8], che permette ad un documento di poter essere elaborato da un qualsiasi sistema automatico. Le operazioni di seguito elencate permettono di ottenere la base da cui creare una rappresentazione matriciale della collection di recensioni, nella quale i documenti sono rappresentati sulle colonne e le parole

sulle righe, con gli elementi della matrice che rappresentano le singole occorrenze della parola nel documento espresse secondo uno specifico schema di pesatura delle occorrenze:

- Tokenization
- Normalization
- Lemmatization o Stemming
- Stopwords removal

3.2.1 Tokenization

È il primo fondamentale passo di elaborazione dei documenti testuali: attraverso questa attività è possibile individuare, all'interno della recensione espressa in linguaggio naturale, l'insieme di unità dense di significato, ovvero le parole o *token* (sequenze di caratteri) che compongono il documento. Si procede a svolgere tale attività sfruttando dei meccanismi basati su espressioni regolari e metodi statistici. In tale fase solitamente vengono svolte anche alcune operazioni fondamentali di riconoscimento della lingua scritta, non necessarie in quanto l'intera collezione è in lingua inglese. In seguito, si è svolta l'eliminazione della punteggiatura, dei caratteri speciali e degli spazi tra le parole, oltre che la rimozione di tutte le cifre che non risultano informative. Il risultato è una semplice lista di token.

3.2.2 Normalization

Tale fase dell'elaborazione del testo consiste nell'applicare un insieme di tecniche che permettono di rappresentare token scritti in formati diversi ma con il medesimo significato in una unica forma comune, evitando che si presenti lo stesso concetto con più forme differenti. Il vantaggio che si ottiene è legato alla riduzione del numero di token duplicati che compongono il dizionario, e quindi la riduzione della dimensionalità della matrice che rappresentazione le recensioni. In questa fase è stata applicata una riduzione di tutti i termini al formato minuscolo e la sostituzione delle parole in forma contratta ("you'll", "I'd", ...) con l'equivalente in forma estesa ("you will", "I would", ...). Per quest'ultima operazione è stata utilizzata la libreria *contractions* [9] che prevede un dizionario di contrazioni e l'equivalente in forma estesa sufficientemente ampio. In questo elaborato non viene presentato il risultato ottenuto dall'applicazione del processo di *spell-checking*, il cui obiettivo è correggere

gli errori di battitura commessi dall'utente, poiché un numero considerevole di token errati sono stati ignorati o convertiti con parole non corrispondenti al termine originale.

3.2.3 Lemmatization

Procedura di ottenimento di un lemma o forma base morfologica della parola da cui si origina il token, che si ottiene a seguito di una analisi morfologica completa. Permette di esprimere token rappresentativi di parole diverse ma derivati dallo stesso lemma con un unica forma comune. Tale analisi ha coinvolto anche l'utilizzo dell'attività di POS Tagging [10] per poter identificare la classe lessicale che ciascun token assume all'interno della frase considerata. Dopo aver applicato il metodo di riconoscimento del tag, specifico per ogni token, grazie all'utilizzo della libreria NLTK [11] e al database WordNet [12] è stato possibile dedurre il lemma. L'applicazione di questa tecnica contribuisce a ridurre la dimensione e la sparsità della matrice.

3.2.4 Stemming

Procedura di estrazione degli *stem*, ovvero delle radici delle parole, a partire dal formato token. Come per il processo di Lemmatization, l'obiettivo finale è la riduzione del numero di token da trattare che consente una riduzione della dimensione e della sparsità della matrice. Questa operazione è stata eseguita attraverso lo *Snowball Stemmer*[13], implementato dalla libreria NLTK. Questo tipo di procedura è alternativa alla procedura di Lemmatization descritta nella Sezione 3.2.3, la motivazione legata alla scelta di possedere due diverse versioni è dovuta alla volontà di effettuare una comparazione dell'efficacia della classificazione sui due risultati ottenuti da processi alternativi.

3.2.5 Stopwords removal

Attività di rimozione di termini chiamati *stopwords* corrispondenti a parole che non portano alcuna informazione aggiuntiva per la rappresentazione testuale, come i termini rappresentativi di articoli, pronomi, congiunzioni, ecc. Si tratta di termini molto frequenti la cui rimozione aiuta a ridurre la sparsità della matrice e la relativa dimensionalità, questi vengono eliminati attraverso una funzione messa a disposizione dalla libreria NLTK per l'individuazione delle stopwords che è stata arricchita, a sua volta, in base ad altri termini non trattati che invece erano contenuti nel testo. Allo stesso modo si procede ad eliminare tutti i termini che vengono dal gruppo considerati stopwords in quanto troppo frequenti o troppo poco frequenti nella collezione

di recensioni, cioè quei termini che causano sparsità nella matrice delle recensioni. Il motivo di tale eliminazione è la volontà di ottenere una rappresentazione dei vettori-recensioni il più compatta possibile, espressa attraverso l'utilizzo di token dotati di grande abilità discriminatoria. Entrambe le rappresentazioni, ottenute con processi di lemmatization e stemming, vengono sottoposti all'eliminazione di stopwords presenti nella lista di *stopword-removal*, se ne crea successivamente una versione ulteriormente filtrata dalle parole troppo frequenti o troppo poco frequenti selezionando solo i token che complessivamente compaiono un numero di volte compreso in [3, 9999].

4 Rappresentazione

A seguito dell'attività di preprocessing del testo si ottiene una rappresentazione intermedia per ogni recensione della collection in due versioni: una ottenuta dal procedimento di lemmatization, l'altra tramite quello di stemming, nelle due versioni indicate nella Sezione 3.2.5. L'attività di preprocessing mantiene le 111860 recensioni originali, producendo però un numero diverso di token considerati. Le dimensioni di ogni collection sono le seguenti:

- Lemmatization standard, 77440 token;
- Lemmatization con filtro aggiuntivo, 34393 token;
- Stemming standard, 60183 token;
- Stemming con filtro aggiuntivo, 27467 token;

Vengono così generati differenti modelli di rappresentazione delle recensioni secondo diversi approcci. Il risultato consiste in una collection strutturata i cui elementi, rappresentanti delle recensioni, sono computabili da un sistema automatizzato.
Le rappresentazioni considerate ai fini del progetto vengono descritte di seguito.

4.1 TF-IDF Matrix

Rappresentazione della collection di documenti basata sulla matrice vocabolario-documenti, dove il vocabolario è composto dall'insieme di tutti i termini che compaiono nella collection (ottenuti come stem o lemma) ed i documenti sono le recensioni. Ogni token del vocabolario viene identificato attraverso una specifica riga della matrice, mentre una recensione è rappresentata da un vettore verticale della matrice

contenente un insieme di pesi $w_{i,j}$, rappresentativi del valore di importanza, determinato dall'abilità discriminatoria del termine corrispondente, calcolato per mezzo di TF-IDF (Formula 1):

$$TF - IDF = \frac{tf_{i,j}}{max(tf_{i,j})} \cdot log\left(\frac{N}{df_i}\right) \tag{1}$$

Si osserva che ogni peso TF-IDF viene ottenuto da due componenti:

- *Term Frequency tf*, che indica il numero di occorrenze del termine i nella recensione j
- *Document Frequency df* che rappresenta l'inverso dell'informatività di un documento, poiché più un termine compare in un numero elevato di documenti meno risulta essere discriminante ed utile.

La formula prevede il prodotto tra il primo termine, che viene normalizzato in funzione della frequenza massima complessiva individuata nel documento $\frac{tf}{max(tf_j)}$, ed il logaritmo del rapporto tra la numerosità della collection N ed il valore di df per il termine j, $log\left(\frac{N}{df_i}\right)$.
La procedura di calcolo della matrice è stata svolta con l'utilizzo della funzione *tfidfvectorizer* della libreria *Sci-Kit Learn* [14], utilizzando i parametri standard che non effettuano, di conseguenza, alcun filtro dei token poiché già svolto manualmente in precedenza. Un problema rilevante nella rappresentazione ottenuta è quello della sparsità della matrice: esso si verifica quando un token compare in un numero estremamente ridotto di documenti, decretando la presenza del valore 0 nell'incrocio riga-colonna per tutti i documenti in cui non compare. Si deduce che la rappresentazione della matrice risulta essere inefficiente e caratterizzata da una dimensione più elevata. Le dimensioni delle rappresentazioni ottenute sono le seguenti:

- Lemmatization standard, 77440 token x 111860 recensioni;
- Lemmatization con filtro aggiuntivo, 34393 token x 111860 recensioni;
- Stemming standard, 60183 token x 111860 recensioni;
- Stemming con filtro aggiuntivo, 27467 token x 111860 recensioni;

4.2 SVD applicata a TF-IDF Matrix

Per risolvere il problema della sparsità della rappresentazione TF-IDF, viene utilizzato il metodo di fattorizzazione della matrice SVD in base agli autovalori ed autovettori della matrice. Tale metodo di *Word Embedding*, denominato *Singular Value Decomposition* [15], permette di scomporre la matrice TF-IDF (indicata con X) come prodotto di tre differenti matrici:

- U: matrice composta dagli autovettori ortonormali alla matrice XX^T
- S: matrice in cui sulla diagonale principale compaiono si singular values, numeri reali non negativi.
- V^T: matrice composta dagli autovettori ortonormali alla matrice X^TX

Nella matrice S si osserva la presenza dei *singular values* sulla diagonale principale, valori che sommati tra di loro permettono di rappresentare il valore di varianza della matrice TF-IDF originale. Selezionando una porzione n di tali valori, in funzione della varianza che si vuole preservare della matrice originale, si procede a troncare i singular values e le righe e colonne corrispondenti. Tramite un prodotto matriciale, che prevede la selezione delle prime n colonne della matrice U e delle prime n righe della matrice V^T si ottiene la versione ridotta della matrice TF-IDF nella quale i vettori riga, denominati *embedding*, possiedono una dimensione compatta (solitamente fino a 100 elementi circa) e densa (cioè perlopiù priva di zeri), composta da un numero ridotto di n features. Il vantaggio di tale rappresentazione è la diminuzione della sparsità della matrice, che può comunque verificarsi, ed una dimensione ridotta che garantisce un apprendimento del classificatore più efficiente, oltre che ad occupare uno spazio in memoria ridotto. Nonostante tali vantaggi, il procedimento del calcolo della matrice dev'essere effettuato ogni volta che cambiano i documenti della collection, con costi quadratici. Il vocabolario, pertanto, risulta essere complesso da mantenere. Tramite questa tecnica di riduzione della matrice TF-IDF si selezionano i valori singolari della matrice fino al numero 100, troncando i rimanenti. Tale valore, che coincide con il valore di default, viene individuato a seguito di diversi tentativi di classificazione; rappresenta un tradeoff che permette di ottenere buone prestazioni e di impiegare un tempo ragionevole per la procedura di Cross Validation. Le dimensioni delle rappresentazioni ottenute sono le seguenti:

- Lemmatization standard, 100 features x 111860 recensioni;
- Lemmatization con filtro aggiuntivo, 100 features x 111860 recensioni;

- Stemming standard, 100 features x 111860 recensioni;
- Stemming con filtro aggiuntivo, 100 features x 111860 recensioni;

4.3 Doc2Vec

Si tratta di un algoritmo di Word Embedding, di tipo predittivo, il cui scopo è produrre una rappresentazione numerica di ciascun documento indipendentemente dalla sua lunghezza. L'algoritmo è derivato dalla tecnica Word2Vec [16]: in questa rappresentazione, i token di un testo vengono rappresentati da un vettore identificativo che li colloca in uno spazio multidimensionale, prodotto utilizzando come base i termini di un insieme di documenti. La rappresentazione Word2Vec può essere generata tramite due diverse architetture:

- Continuous Bag-of-Words (CBOW), architettura che considera una finestra di *context-word* intorno ad una *center-word* per prevedere quest'ultima;
- Skip-gram, architettura in grado di generare la rappresentazione di una *center-word*, tramite la previsione delle *context-word* relative ad una parola target, massimizzando la probabilità delle parole di contesto.

Nella rappresentazione Doc2Vec [17], invece, si vuole rappresentare ogni documento come un vettore di features. Per poter adattare il procedimento di creazione della rappresentazione Word2Vec a quello Doc2Vec, è necessario aggiungere all'architettura una componente chiamata *paragraph matrix* che fornisce informazioni specifiche ed univoche per ogni documento. Esistono due differenti architetture dedicate, simili e derivate da partire quelle di Word2Vec appena descritte:

- *Distributed Memory*, derivato da *Continuous Bag-of-Words*
- *Distributed Bag Of Words (DBOW)*, derivato da *Skip-gram*

Nello specifico la versione utilizzata per creare la rappresentazione Doc2Vec è quella DBOW, che permette di determinare a priori la dimensione dei vettori associati ad ogni documento, in questo caso viene determinata dal gruppo di lavoro per una dimensione finale pari a 300. Si può osservare la struttura semplificata nell'immagine 3. Questa architettura garantisce tempi ridotti per la fase di addestramento del modello e per la creazione della rappresentazione, a differenza di Distributed Memory che invece sfrutta una rappresentazione più complessa e non idonea ai nostri scopi. Il tempo impiegato per produrre un risultato, comprese le operazioni di creazione del vocabolario, risulta essere estremamente oneroso e superiore all'ora con la potenza

computazionale a disposizione. Per tale motivo non si considera l'applicazione del modello Distributed Memory, che impiega un tempo ancora maggiore, né l'utilizzo di una dimensione finale superiore a 300. Tale condizione rappresenta un vincolo che permette di considerare l'applicazione della tecnica solamente per le versioni della collection ottenute filtrando anche i token troppo frequenti e troppo poco frequenti.

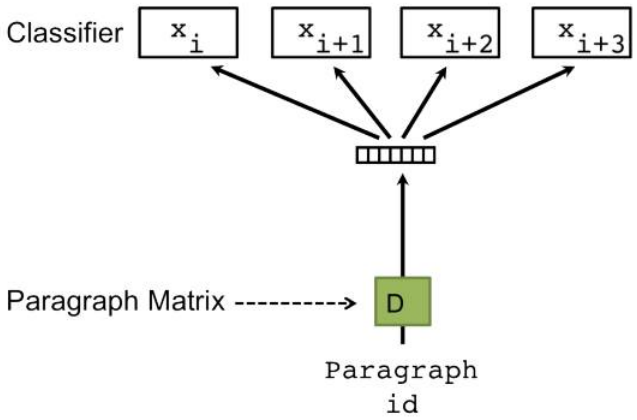

Figure 3: Struttura DBOW

Le dimensioni delle rappresentazioni ottenute sono le seguenti:

- Lemmatization standard, 300 features x 111860 recensioni;
- Lemmatization con filtro aggiuntivo, 300 features x 111860 recensioni;
- Stemming standard, 300 features x 111860 recensioni;
- Stemming con filtro aggiuntivo, 300 features x 111860 recensioni;

5 Classificatori

Il problema che si è deciso di affrontare riguarda l'assegnazione di una classe di appartenenza a ciascuna recensione tramite algoritmi di apprendimento automatico: in questo caso, il problema da affrontare consiste in una classificazione non-binaria a singola etichetta (*SLMC*), ovvero una tipologia di classificazione in cui la variabile target di ciascuna istanza è descritta solamente da un valore che è, a sua volta, parte di un insieme definito e limitato di almeno tre valori (nel caso in esame le

classi corrispondono a 4). Durante la fase di pianificazione del progetto si è cercato, sia dalla letteratura sia attingendo a conoscenze pregresse, un algoritmo di *machine learning* supervisionato che meglio si adattasse al problema proposto: la selezione dell'algoritmo da utilizzare è pesantemente influenzata sia dalle sue capacità di classificazione, quindi dalla sua efficacia nel prevedere la classe target, sia dalla sua capacità di produrre previsioni in tempi rapidi, ovvero dalla sua efficienza, in quanto è necessario rimanere nei vincoli già precedentemente descritti nel Capitolo 1.

5.1 Random Forest

Random Forest è un algoritmo di apprendimento basato su un insieme di *decision tree*. Si tratta, dunque, di un modello *ensemble*: ciò significa che vengono combinati più modelli dove ciascun classificatore fornisce una valutazione rispetto all'assegnazione di un'etichetta per ciascuna istanza e, in base alla modalità di *voting* definita, viene decretata l'etichetta finale. Nel caso del Random Forest l'esito della classificazione viene stabilito tramite la modalità *Max Voting*, cioè si attribuisce all'osservazione la classe con il maggior numero di voti.

5.2 XGBoost

XGBoost è un modello di classificazione che implementa un *Gradient Boosted Decision Tree*. Questo classificatore prevede di realizzare un determinato numero di alberi decisionali in sequenza, in modo tale che ogni *decision tree* complementi quelli precedentemente costruiti.

5.3 Support Vector Machine

SVM è un algoritmo di Machine Learning supervisionato che rappresenta i record del dataset in uno spazio n-dimensionale. Trovare la soluzione significa trovare gli iper-piani che separino al meglio i diversi elementi, in altre parole si vogliono isolare i record in base al valore della loro variabile target. Il classificatore implementato è denominato *Support Vector Classifier* ed è stato prelevato dalla libreria *sklearn* messa a disposizione da *Scikit-Learn*.

5.4 K-Nearest Neighbors

K-Nearest Neighbor è un modello di apprendimento supervisionato, che utilizza il concetto di *neighbors* per assegnare l'etichetta finale ad un'istanza. L'idea generale è quella di rappresentare le osservazioni sotto forma di punti in uno spazio

n-dimensionale (considerando n features), e, fornita una *distance function*, si procede a calcolare la distanza con le k osservazioni circostanti per ogni osservazione. Si procede quindi ad assegnare ad ogni istanza i la classe di appartenenza in funzione della classe più rappresentata nello spazio comprendente i k neighbors considerati.

6 Risultati

Considerati gli algoritmi presentati nella Sezione 5, è stata svolta l'attività di apprendimento supervisionato sfruttando l'intera collection di recensioni ottenuta a seguito delle operazioni descritte nella Sezione 3. Nello specifico, il processo di apprendimento e valutazione degli algoritmi è stato implementato attraverso una procedura di *5-fold Cross Validation*: tale meccanismo di valutazione separa la collection in 5 fold diversi secondo un sampling stratificato. Per 5 diverse iterazioni un algoritmo viene allenato su osservazioni di cui si possiedono le etichette contenute in 4 dei 5 fold, poi a seguito della procedura di classificazione, in cui si procede ad assegnare la classe corrispondente ad ogni osservazione, viene valutato il fold rimanente. Ad ogni iterazione i fold usati per l'apprendimento e per la valutazione cambiano, arrivando a fine procedura dopo aver implementato tutte le combinazioni; questa tecnica permette di ottenere una valutazione esaustiva degli algoritmi che non è influenzata dalla porzione dei dati usata per allenare e per valutare l'algoritmo.
La valutazione dell'attività di Text Classification avviene attraverso due diversi criteri di valutazione: efficacia ed efficienza. L'efficacia viene valutata utilizzando delle misure apposite che agiscono sull'insieme di classi assegnate dal classificatore confrontandole con quelle reali. Nel caso in esame è stata utilizzata la metrica *Accuracy* (Formula 2) per ogni iterazione della 5-fold CV: questa valuta la porzione di scelte corrette effettuate dall'algoritmo nella porzione di dati di test rispetto al totale dei dati classificati dal classificatore; se ne può osservare la formula di seguito.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2}$$

L'Accuracy ottenuta per l'intero processo di Cross Validation viene calcolata come il valore medio, a cui si aggiunge un'informazione aggiuntiva relativa all'intervallo di confidenza che è estremamente utile per poter decretare il miglior classificatore quando sono comparati.

La valutazione dell'efficienza del modello invece viene stabilita in funzione del tempo che ogni classificatore impiega per poter svolgere l'intera fase di Cross Validation, influenzato enormemente dalla dimensione della rappresentazione finale della

collection di recensioni. Si ricorda inoltre che sia l'efficacia sia l'efficienza che ogni classificatore consegue sono estremamente influenzate dal suo funzionamento interno. A tale scopo si decide di valutare ogni combinazione di algoritmo, rappresentazione delle recensioni e attività di processing testuale (Lemmatization vs Stemming - che sono identificate con *Lemma* e *Stem* nelle tabelle) precedentemente proposti.

6.1 Classificazione con TF-IDF

Si osservano i risultati della classificazione ottenuti sulla rappresentazione TF-IDF presentati nella Tabella 1:

Classificatore	**Preprocessing**	**Accuracy media**	**Intervallo di confidenza**	**Tempo**
Random Forest	Lemma	0.8523	[0.8497, 0.8549]	40 m 56 s
–	Lemma filtro	0.8627	[0.8610, 0.8644]	32 m 20 s
–	Stem	0.8538	[0.8496, 0.8580]	37 m 11 s
–	Stem filtro	0.8638	[0.8619, 0.8657]	30 m 48 s
XGBoost	Lemma	0.8495	[0.8473, 0.8517]	39 m 37 s
–	Lemma filtro	0.8167	[0.8137, 0.8197]	25 m 55 s
–	Stem	0.8536	[0.8515, 0.8557]	41 m 2 s
–	Stem filtro	0.8171	[0.8152, 0.8204]	25 m 7 s
SVM	Lemma	0.9242	[0.9226, 0.9258]	12 s
–	Lemma filtro	0.9146	[0.9137, 0.9155]	9 s
–	Stem	0.9245	[0.9226, 0.9264]	12 s
–	Stem filtro	0.9141	[0.9127, 0.9155]	9 s
KNN	Lemma	0.8612	[0.8567, 0.8637]	8 m 57 s
–	Lemma filtro	0.8342	[0.8305, 0.8379]	5 m
–	Stem	0.8631	[0.8607, 0.8655]	9 m 51 s
–	Stem filtro	0.8145	[0.8064, 0.8226]	5 m 14 s

Table 1: Tabella dei risultati con rappresentazione TF-IDF

Si osserva, dai risultati in Tabella 1, che vi è un netto vantaggio, in termini di accuratezza ed efficienza, per il modello SVM: il tempo impiegato è compreso tra i 9 e i 12 secondi per l'esecuzione della Cross Validation, a seconda della versione di rappresentazione considerata. Nello specifico, attraverso l'utilizzo degli intervalli di confidenza, è possibile decretare che le versioni non filtrate, "Lemma" e "Stem" (evidenziate in giallo), sono le migliori, senza però poter affermare che uno sia migliore dell'altro. Osservando nel complesso i modelli si osserva un'enorme disparità nei

tempi di esecuzione, accuratezze inferiori e una differenza sostanziale del tempo impiegato per concludere la procedura tra classificatori basati su rappresentazioni complete e filtrate (a favore delle ultime).

6.2 Classificazione con SVD

I risultati ottenuti per la rappresentazione generata con l'applicazione di SVD sono i seguenti contenuti nella Tabella 2:

Classificatore	Preprocessing	Accuracy media	Intervallo di confidenza	Tempo
Random Forest	Lemma	0.8502	[0.8476, 0.8528]	13 m 36 s
–	Lemma filtro	0.8587	[0.8558, 0.8616]	15 m 33 s
–	Stem	0.8496	[0.8467, 0.8525]	14 m 56 s
–	Stem filtro	0.8593	[0.8573, 0.8613]	14 m 46 s
XGBoost	Lemma	0.8602	[0.8581, 0.8623]	49 m 31 s
–	Lemma filtro	0.8591	[0.8566, 0.8616]	44 m 54 s
–	Stem	0.8595	[0.8573, 0.8617]	52 m 47 s
–	Stem filtro	0.8603	[0.8592, 0.8614]	52 m 38 s
SVM	Lemma	0.8696	[0.8674, 0.8718]	37 s
–	Lemma filtro	0.8586	[0.8571, 0.8601]	48 s
–	Stem	0.8684	[0.8663, 0.8705]	40 s
–	Stem filtro	0.8598	[0.8582, 0.8614]	47 s
KNN	Lemma	0.8268	[0.8243, 0.8293]	57 m 49 s
–	Lemma filtro	0.8097	[0.8059, 0.8135]	48 m 19 s
–	Stem	0.8285	[0.8261, 0.8309]	64 m 15 s
–	Stem filtro	0.8127	[0.8113, 0.8141]	61 m 40 s

Table 2: Tabella dei risultati con rappresentazione SVD

La Tabella 2 che include i risultati della procedura SVD offre una chiave di lettura molto simile alla Tabella 1 prodotta in precedenza per TF-IDF: in particolare, anche in questo caso si evidenzia come l'algoritmo SVM risulta essere molto migliore degli altri sotto il profilo dell'efficienza e, contemporaneamente, si verifica che le esecuzioni "Stem" e "Lemma" ottengono un'accuratezza maggiore ma tempi di esecuzione più elevati rispetto alle controparti filtrate. Si osserva che le prestazioni in termini di efficacia siano molto simili tra tutti gli algoritmi, con il solo KNN che ottiene risultati leggermente peggiori; anche in questo caso l'algoritmo migliore risulta essere SVM ma senza possibilità di distinguere tra le varie accuratezze ottenute da tale algoritmo.

Infine, effettuando un confronto con i risultati ottenuti dalla matrice TF-IDF, si osserva che i migliori classificatori addestrati e valutati sulla rappresentazione ridotta tramite SVD risultano avere prestazioni inferiori. Il modello Random Forest risulta ridurre il tempo di esecuzione, mentre si osserva un incremento sostanziale del tempo impiegato dagli altri algoritmi.

6.3 Classificazione con Doc2Vec

Per motivi legati alle risorse di tempo e calcolo disponibili, si procede a considerare solo le versioni della rappresentazione Doc2Vec in cui i termini sono filtrati in base alle considerazioni del gruppo di lavoro, come già anticipato. Un'altra considerazione viene effettuata in relazione agli algoritmi presentati: solo Random Forest, SVM e KNN hanno concluso l'esecuzione nei tempi concessi dalla macchina virtuale, XGBoost è stato escluso in quanto le risorse computazionali non sono state sufficienti per poter svolgere il compito. I risultati vengono visualizzati in Tabella 3:

Classificatore	**Preprocessing**	**Accuracy media**	**Intervallo di confidenza**	**Tempo**
Random Forest	Lemma filtro	0.5140	[0.5124, 0.5155]	16 m 11 s
–	Stem filtro	0.5082	[0.5061, 0.5103]	17 m 43 s
SVM	Lemma filtro	0.4989	[0.4961, 0.5106]	34 m 16 s
–	Stem filtro	0.4969	[0.4932, 0.5006]	50 m 50 s
KNN	Lemma filtro	0.5011	[0.4986, 0.5036]	105 m 31 s
–	Stem filtro	0.5032	[0.5, 0.5064]	132 m 8 s

Table 3: Tabella dei risultati con rappresentazione Doc2Vec

I risultati ottenuti da Doc2Vec, riportati in Tabella 3, risultano essere ampiamente peggiori rispetto a quelli restituiti dalle precedenti esecuzioni, da tutti i punti di vista: tutti gli algoritmi vanno incontro ad un netto peggioramento dei tempi di completamento del task, inoltre la valutazione dell'accuratezza dei vari classificatori non supera in nessun caso la soglia del 50%, il che rende praticamente incompatibile ed invalutabile l'utilizzo di questa tecnica con i dati, il preprocessing ed i classificatori utilizzati.

7 Conclusioni

Il progetto si è incentrato sul compito di Text Classification, con particolare riguardo alla sperimentazione di differenti rappresentazioni, ed il relativo confronto delle

prestazioni. Dal punto di vista complessivo, è stato individuato un modello di classificazione che soddisfa ampiamente i requisiti del gruppo legati alla volontà di trovare un algoritmo efficiente ed efficace: SVM allenato sulle rappresentazioni TF-IDF, sia in versione Stem che Lemma. Questo classificatore raggiunge infatti una accuracy complessiva del 92.4%, un risultato estremamente soddisfacente se confrontato con i modelli alternativi, distanziati di almeno 6 punti percentuali. Per quanto riguarda l'efficienza di addestramento e valutazione si osserva il sorprendente tempo di esecuzione dell'algoritmo di soli 12 secondi. Si decreta che senza ombra di dubbio tale soluzione sia la migliore proposta, nonché quella che verrebbe applicata in un contesto di utilizzo reale. In generale, si osserva che le rappresentazioni ottenute con SVD e Doc2Vec risultano essere particolarmente deludenti: la dimensione ridotta delle recensioni ottenuta tramite il procedimento di fattorizzazione delle matrici con SVD non garantisce un miglioramento rispetto alla versione TF-IDF, anzi, in alcuni casi le prestazioni sono inferiori ed i tempi di esecuzione della Cross Validation più alti; si presume che tale problematica possa essere affrontata e risolta tramite l'utilizzo di una dimensione più elevata di features rappresentative della recensione, garantendo quindi una varianza catturata dalla matrice originale superiore, a condizione che però le risorse computazionali e di tempo vengano incrementate. La rappresentazione Doc2Vec invece, come già evidenziato, non permette neanche lontanamente di essere considerata per un'applicazione del sistema in un contesto reale: il risultato estremamente insoddisfacente ha portato a considerare che per impiegare in maniera efficiente tale rappresentazione, con una collection di dati dalle dimensioni elevate come quella considerata, sarebbe necessario disporre di risorse computazionali ben più generose di quelle attuali. Uno sviluppo futuro che si propone è proprio quello legato alla riproduzione dell'intero esperimento con mezzi adeguati, poiché le risorse garantite da Google Colab non sono disponibili per un tempo sufficiente ad ottimizzare i parametri di SVD e Doc2Vec con tecniche adeguate. In relazione all'interesse del gruppo di valutare delle differenze tra le rappresentazioni basate sull'approccio Lemmatization e Stemming, si sancisce che nel contesto di lavoro considerato non è possibile individuare un approccio evidentemente migliore dell'altro. Si propone infine un ultimo sviluppo implementabile: implementare un modello di classificazione basato sull'algoritmo di Rete Neurale, con lo scopo di garantire un apprendimento dei parttern presenti nelle rappresentazioni in maniera più approfondita rispetto ai modelli già implementati.

References

[1] Wikipedia. Amazon.com, inc. Amazon Wiki.

[2] F. Beil, M. Ester, and X. Xu, "Frequent term-based text clustering," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 436–442.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[4] R. K. Bakshi, N. Kaur, R. Kaur, and G. Kaur, "Opinion mining and sentiment analysis," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2016, pp. 452–455.

[5] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining text data*. Springer, 2012, pp. 163–222.

[6] K. Chowdhary, "Natural language processing," in *Fundamentals of Artificial Intelligence*. Springer, 2020, pp. 603–609.

[7] W. McKinney *et al.*, "pandas: a foundational python library for data analysis and statistics," *Python for High Performance and Scientific Computing*, vol. 14, no. 9, 2011.

[8] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 43–52, 2010.

[9] PyPI. Contractions. Contractions.

[10] E. Brill, "A simple rule-based part of speech tagger," PENNSYLVANIA UNIV PHILADELPHIA DEPT OF COMPUTER AND INFORMATION SCIENCE, Tech. Rep., 1992.

[11] E. Loper and S. Bird, "Nltk: the natural language toolkit," *arXiv preprint cs/0205028*, 2002.

[12] C. Fellbaum, "Wordnet," *The encyclopedia of applied linguistics*, 2012.

[13] M. F. Porter, "Snowball: A language for stemming algorithms," 2001.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[15] V. Klema and A. Laub, "The singular value decomposition: Its computation and some applications," *IEEE Transactions on automatic control*, vol. 25, no. 2, pp. 164–176, 1980.

[16] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[17] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, 2014, pp. 1188–1196.