Text Mining and Search

Supervised, unsupervised learning e topic modelling su copioni cinematografici

Pietro Carmine Valenti, Mario Alessandro Napoli, Paolo Lindia
CdLM in Data Science, Università degli studi di Milano-Bicocca

Abstract

In un contesto in cui l'informazione diventa non solo un mezzo per studiare i fenomeni e prevederne i possibili andamenti, ma addirittura un prezioso asset che le aziende acquisiscono e vendono come una risorsa fisica, è assolutamente naturale che si cerchi di estrapolare dati dalle più disparate fonti. L'idea di *Text Mining* si colloca esattamente in tale contesto: riuscire a ricavare dati strutturati, organizzabili in schemi e comprensibili ad una macchina da forme testuali ricche non solo di complessità sintattica, ma soprattutto di realtà semantica. Ad oggi non esistono ancora computer in grado di catturare fedelmente il senso semantico dei testi ed interpretarlo in maniera simile all'uomo; tuttavia grazie allo sviluppo di algoritmi di Machine Learning e allo sviluppo di tecniche di *Features Extraction* sempre più avanzate, si è oggi in grado di rispondere a numerosi task che nascono dalla realtà testuale. Tale studio si colloca in tale ambito e si prefigge l'obiettivo di applicare lo stato dell'arte del *Text Mining*, in termini di estrazione di dati e analisi, a copioni di film.

Keywords: Text Mining, Supervised Learning, Unsupervised Learning, Topic Modelling, Features Engineering, Movies.

Indice

1 Introduzione **2**

2 Dati **3**
2.1 Web Scraping 3
2.2 TMDB api interrogation 4
2.3 Data Enrichment 5

3 Pre-Processing e Features Extraction **6**

4 Supervised Learning **8**

5 Unsupervised Learning **11**

6 Topic Modelling **14**

7 Conclusioni e possibili sviluppi **15**

Introduzione

L'industria cinematografica è tra le più importanti nell'ambito dell'intrattenimento, sia dal punto di vista economico, sia dal punto di vista della capacità di raggiungere le persone: una porzione significativa di popolazione mondiale occupa parte del suo tempo libero a guardare film. Una caratteristica fondamentale che rende i film così amati è la possibilità di unire insieme capacità attoriali, abilità nella scrittura di testi, gusto estetico e fotografico e utilizzo di strumentazioni tecnologiche per migliorare la qualità visiva; musica, teatro, radio e qualsivoglia altra industria dell'intrattenimento, non si collocano in un contesto in cui tale moltitudine di caratteristiche trovano unione. Se da un lato il cinema ha una forte e varia componente artistica, dall'altro è un settore economico in cui trovano impiego artisti, tecnici e imprenditori; è proprio grazie a questo secondo aspetto che l'analisi dei dati trova in tale settore terreno fertile per diverse applicazioni, che siano descrittive, di mercato o predittive.

Nonostante alcune branche dell'analisi dei dati siano impiegate in maniera diffusa dall'industria del cinema, tecniche più avanzate come il *Machine Learning* (ML), e di conseguenza il *Text Mining* (TM), non hanno ancora trovato uno spazio applicativo in ambito business, seppur mantenendo un notevole interesse dal punto di vista accademico e di ricerca. Se da un lato l'analisi predittiva sugli incassi o l'analisi descrittiva sui migliori orari in cui proiettare il film (magari in base al genere o all'età del pubblico di riferimento) hanno evidenti applicazioni di business, lo studio testuale automatizzato di un copione rischia di scontrarsi e/o contrapporsi con la sfera artistica intrinseca del cinema. Dopotutto quasi nessun artista vorrebbe farsi consigliare da un algoritmo il miglior ordine in cui dire delle battute, o quali termini siano più adeguati al tipo di film a cui prendono parte.

E' stato necessario introdurre lo studio utilizzando tali considerazioni non per semplice retorica, ma per chiarire subito la natura dell'analisi. Non si ha la pretesa che le tecniche impiegate abbiano un reale impiego e/o portino un vantaggio al settore di riferimento. Il progetto ha uno scopo puramente teorico e utilizza il mondo del cinema come base di dati di riferimento per applicare tecniche di TM e di *Features Engineering*.

La base di dati di partenza è stata costruita utilizzando informazioni reperibili online (approfondimento nella Sez.2) e integrando i copioni di 28 000 film con le relative informazioni generali; tali informazioni sono poi alla base degli studi e delle tecniche di TM impiegate. Il progetto ha poi lo scopo di applicare 3 tipologie di analisi:

- **Supervised Learning**: utilizzo di features estratte dal testo grezzo al fine di classificare i copioni in base al genere (Sez.4).
- **Unsupervised Learning**: utilizzo di features estratte dal testo grezzo per segmentare la base informativa in base ad una certa metrica di similarità (Sez.5).
- **Topic Modelling**: individuazione di strutture semantiche nascoste nel testo (Sez.6).

Dati

E' necessario precisare che la base di dati non viene pensata inizialmente per tale studio, ma è stata costruita per un precedente progetto. Il sito `https://www.springfieldspringfield.co.uk/`, utilizzato per l'ottenimento dei copioni, purtroppo non è più reperibile. Vengono di conseguenza descritti per completezza tutti i passaggi che hanno portato alla costruzione della base di dati, non essendo tuttavia strettamente legati allo studio in esame.

2.1 Web Scraping

Essendo buona parte del progetto basata su analisi testuale, il criterio iniziale di selezione dei film è basato sulla disponibilità di reperimento in rete dei copioni degli stessi. Per fare ciò abbiamo utilizzato il sito *Springfield!Spingfield!* (`https://www.springfieldspringfield.co.uk/`), il quale rende disponibili gratuitamente i copioni di circa 30 mila film a partire dal 1910. Il primo passo per costruire la base di dati è consistito nella raccolta di tutti i titoli, gli anni di produzione e i copioni dei film contenuti in tale sito; per fare ciò abbiamo utilizzato tecniche automatizzate di *web scraping* attraverso l'esplorazione della struttura HTML del sito stesso.

Lo strumento utilizzato per lo sviluppo di un codice capace di "rastrellare" tutti i dati necessari al progetto è il software Python; per questa prima parte è stato fondamentale l'utilizzo della libreria *BeautifulSoup*, la quale risulta tra le più efficienti per effettuare *scraping* dal HTML dei siti internet. Dopo uno studio approfondito del sito contenente i copioni e dopo aver verificato la struttura divisa per lettere dell'alfabeto, con ciascun film inserito nella pagina corrispondente all'iniziale del titolo inglese, è stato possibile determinare un path comune a tutte le pagine del sito: *"https://www.springfieldspringfield.co.uk/movie_scripts.php?order=&page="*. Tale url rappresenta la base comune a tutte le pagine, più due elementi che variano in base alla posizione in cui ci si trova; al comando order= viene sostituita la lettera dell'alfabeto che rappresenta la sezione del sito in cui ci si trova (con 0 rappresentante i simboli e i numeri), mentre page= indica la pagina in cui ci si trova all'interno della sezione. Definita tale struttura è stato possibile ricavare tutte le informazioni necessarie allo sviluppo del codice Python direttamente dal HTML: la radice comune degli indirizzi url, gli indicatori delle diverse sezioni (lettere dell'alfabeto latino più lo zero) e il numero di pagine per ciascuna sezione del sito.

Tenendo conto di tali elementi è stato sviluppato un ciclo che iterasse esternamente sulle sezioni, e internamente sulle pagine; ad ogni iterazione viene creato un indirizzo url di una pagina associata ad un determinato film, dal quale è possibile ricavare il titolo, l'anno di uscita e il copione integrale. Con la giusta combinazione di funzioni della libreria *BeautifulSoup* è stato possibile estrarre tali informazioni per un totale di più di 28 mila film.

Durante questa fase le principali irregolarità riscontrate nei dati si riferiscono prettamente ad errori del sito stesso: la prima problematica riscontrata riguarda la presenza di alcuni indirizzi di pagine alle quali non è associato alcun copione (*missing values*), mentre la seconda fa riferimento ad alcuni film per il quale sono presenti più pagine replicate (valutati utilizzando congiuntamente titolo e anno per evitare di eliminare possibili *remake*). Tali irregolarità hanno

trovato una semplice soluzione attraverso l'inserimento di apposite condizioni all'interno del ciclo, capaci di sopperire al problema filtrando sia i valori mancanti, sia i dati replicati.

Qui di seguito viene riassunto in maniera sintetica il funzionamento dell'algoritmo utilizzato per il *Web Scraping*:

1. Viene definito il path comune a tutte le sezioni del sito.
2. Viene sviluppato un ciclo che iteri su una lista comprendente lo zero e le lettere dell'alfabeto latino (rappresentanti le sezioni del sito).
3. Attraverso le apposite funzioni di *BeautifulSoup* si accede all'indirizzo url iniziale di ciascuna sezione del sito e si ricava il numero di pagine contenute in ciascuna di esse.
4. Viene sviluppato un ciclo interno che iteri sul numero di pagine contenute in ciascuna sezione, e che quindi consenta di muoversi all'interno di esse.
5. All'interno di ciascuna sezione e pagine è possibile estrarre le informazioni desiderate (titolo, anno, copione) direttamente dal codice HTML. Per ovviare alle problematiche precedentemente descritte vengono applicati alcuni filtri.
6. Per alleggerire il carico di lavoro e diminuire il rischio di perdita di informazioni, dovuta di interruzioni dell'algoritmo, al termine dello *scraping* di ciascuna sezione viene creato un *DataFrame* (libreria *pandas*) contenente il titolo, l'anno di produzione, l'indirizzo url e il copione per ciascun film appartenente alla sezione appena esplorata; la tabella viene poi esportata in formato CSV e salvata automaticamente in locale, prima di "rastrellare" la nuova sezione.

2.2 TMDB api interrogation

Ottenuti tutti i dati preliminari, come descritto nella precedente sezione, è stata ritenuta necessaria un'integrazione con informazioni riguardanti le caratteristiche generali di ciascun film; risulta chiaro come i soli dati riguardanti i copioni non siano sufficienti allo svolgimento di un'analisi approfondita del tema alla base del progetto. Al fine di ottenere tali informazioni aggiuntive è stato selezionato il sito online *The Movie Database* (TMDb, `https://www.themoviedb.org/`), un sito che mette a disposizione una base di dati che raccoglie informazioni su un numero considerevole di film. Il vantaggio garantito dalla selezione di tale sito risiede sia nella vastità della banca dati, sia nella disponibilità di un'apposita API per l'interrogazione della stessa (documentata all'indirizzo `https://developers.themoviedb.org/3/`), la quale viene inoltre implementata in Python attraverso la libreria *tmdbsimple*.

I dati ritenuti più utili per l'arricchimento della base di dati sono: budget, incasso, luogo di produzione, regista (nome, genere), cast (attore, genere, provenienza, popolarità e personaggio interpretato), casa di produzione cinematografica (nome e provenienza), numero di voti e voto medio da parte degli utenti.

L'interrogazione dell'API si basa su funzioni che richiedono la conoscenza di un ID interno, univoco per ciascuna osservazione, ricavabile mediante l'utilizzo del titolo di ciascun film ottenuto in precedenza; poiché la funzione per ottenere l'ID univoco accetta come parametro opzionale anche l'anno, abbiamo utilizzato tale informazione per separare eventuali *remake* o film omonimi. Ottenuto tale identificativo è stato possibile estrarre le informazioni precedentemente elencate attraverso due funzioni dell'API:

1. La funzione Movies, la quale accetta come parametro l'ID univoco e restituisce un file JSON (*document-based*), dal quale sono ricavabili le informazioni circa il budget, l'incasso, il luogo di produzione, il regista (id, nome), la composizione del cast (attore, id,

personaggio interpretato), la casa di produzione cinematografica (nome e provenienza), il numero di voti e il voto medio da parte degli utenti

2. La funzione People consente invece di ricavare le informazioni circa il genere, la provenienza e la popolarità sia degli attori, sia dei registi; il parametro richiesto per tale interrogazione è l'id univoco della persona.

2.3 Data Enrichment

Ottenute tutte le informazioni descritte nelle precedenti sezioni, è stato necessario definire uno schema per l'arricchimento dei dati ottenuti mediante *web scraping* con i dati ottenuti mediante API *interrogation*. Come chiave d'integrazione tra le due collezioni di dati è stato utilizzato l'identificativo univoco (ID) del sito TMDb, il quale è stato prima associato a ciascun titolo dei copioni ottenuti tramite *scraping* e successivamente utilizzato per iterrogare TMDb. Al fine di ottenere un *Database* robusto sono stati inseriti due filtri durante il processo di arricchimento dei dati:

1. Non sono stati presi in considerazione i film precedenti il 1960, poiché durante la fase preliminare di esplorazione dei dati non è stata riscontrata una presenza sufficiente a giustificare l'inserimento nel *Database*; inoltre, poiché il cinema ha subito numerosi cambiamenti negli anni, al fine della nostra analisi tali film avrebbero potuto generare effetti distorsivi.

2. Sono stati scartati i film per i quali non si è ottenuta una risposta dall'API di TMDb: per alcuni film infatti non è stato possibile ricavare l'identificativo (ID), attraverso l'impiego di titolo e anno estratti tramite *scraping*, necessario a interrogare la banca dati al fine di ottenere le informazioni aggiuntive.

In seguito all'inserimento di tali condizioni si è generata una riduzione delle osservazioni da 28 a 21 mila: tale diminuzione della numerosità del campione iniziale non ha destato particolari preoccupazioni dal momento che tali film non avrebbero ricevuto alcuna integrazione con le informazioni generali, ottenute da TMDb.

Per la struttura finale del *Database* si è optato per una forma *document-based*, organizzando le informazioni dei vari film in file JSON. Tale decisione deriva sia dalla tipologia dei dati, sia dalla flessibilità garantita dal modello documentale; non avendo la necessità di una definizione a priori dello schema (in quanto *schema-free*), i modelli NoSQL sono ideali per la costruzione di *Database* che richiedono operazioni di inserimento, aggiornamento o cancellazione di attributi e/o campi nel tempo; essendo la produzione di film un fenomeno continuo, tale condizione risulta estremamente utile. In secondo luogo la presenza di numerosi campi annidati, come le informazioni sul cast, sul regista, sulle case di produzione e sul copione (approfondite successivamente nella Capitolo 3), confermano nuovamente la necessità di una struttura con schema annidato, garantita dal modello *document-based*.

Come struttura finale del *Database* si è deciso per una suddivisione in 27 file JSON (uno per ogni lettera dell'alfabeto latino più lo 0), ognuno dei quali raccoglie tutte le informazioni riguardanti i film il cui titolo (in lingua inglese) inizia con la lettera/simbolo rappresentante del file (0.json, A.json, ..., Z.json). La prima motivazione che ha portato a decidere per una suddivisione simile riguarda l'onerosità di raccolta e di memorizzazione delle informazioni. Avendo utilizzato fonti dati online si è riscontrata una limitazione dovuta alla velocità di risposta e al rischio di interruzione dell'esecuzione; di conseguenza un sistema di memorizzazione progressivo ha consentito di smorzare tali problematiche. La seconda motivazione è invece legata a fattori di potenza computazionale: la suddivisione della struttura ha consentito consentito di mettere in atto misure di *horizontal scaling*, e quindi suddividere il carico di lavoro su diverse macchine,

in particolare con l'ausilio di 5 dispositivi. L'ultima motivazione è legata infine ai processi successivi alla memorizzazione, in particolare alla necessità di interrogare il *Database* nel tempo; la suddivisione consente di ottenere risposte senza la necessità di analizzare l'intera base di dati, ma prendendo in esame solo le sezioni che sono d'interesse.

Pre-Processing e Features Extraction

Quando si parla di *Text Mining* la base informativa da cui si attinge è composta generalmente di dati non strutturati, testi carichi di significato semantico non comprensibile alla maggior parte dei computer. Affinché tali testi siano utilizzabili come input da un qualsiasi algoritmo di *Machine Learning* è necessario estrarre dal documento grezzo dati strutturati, come vettori, matrici, etc. Tale procedimento viene definito proprio *Features Engineering* o *Extraction*; lo scopo di tale procedura consiste proprio nell'analizzare i dati testuali, individuare gli elementi caratteristici, come ad esempio termini ricorrenti, estrarli e convertirli in dati strutturati (preferibilmente numerici). L'estrazione di variabili non è tuttavia un processo che può essere applicato semplicemente sui dati grezzi, poiché in tal caso si otterrebbero risultati poco interessanti: in qualsiasi tipo di testo minimamente complesso le parole più ricorrenti sono quelle menu utili, come ad esempio articoli, congiunzioni, etc.

Il primo passo per lo sviluppo di uno studio di *Text Mining* consiste nella suddivisione del testo in liste o vettori di elementi più semplici chiamati lessemi; tale procedimento viene definito *tokenization*. Successivamente si procede con l'eliminazione degli elementi meno interessanti: le *stopwords*. Per entrambe le procedure esistono molte soluzioni implementate da diversi programmi e piattaforme; essendo la parte computazionale del progetto sviluppata interamente in ambiente *Python*, si è optato per l'utilizzo della libreria *nltk*, la quale fornisce dizionari in lingue diverse e soluzioni *ad hoc* per la rimozione delle *stopwords*. Ottenuti dunque vettori di *tokens* è stato necessario controllare che la lingua fosse unica: avere testi in lingua differente avrebbe sicuramente generato molteplici problemi allo studio. Identificati tali testi è stato sufficiente tradurli in maniera automatizzata.

Finita la fase di *pre-processing* è stato possibile passare alla scelta delle *features* da estrarre. Di seguito vengono descritte una ad una.

La prima tipologia, nonché la più semplice e meno efficiente per la soluzione del problema, è la **Term-Frequency** (TF). La TF fornisce indicazioni circa la frequenza di comparsa di ciascun termine (o nel caso specifico token) all'interno di un testo in linguaggio naturale. La TF si può presentare sia in forma di frequenza assoluta (*word count*), sia in forma di frequenza relativa. Si è scelto di utilizzare la seconda opzione poiché i copioni dei film hanno lunghezza differente (a causa della durata o del numero/ampiezza delle battute), di conseguenza un conteggio assoluto sarebbe risultato distorto. Chiaramente la sua semplicità porta con sé alcune criticità: sapere che un determinato testo presenta un *token* o parola in una frequenza significativa non è necessariamente un elemento interessante. Tale procedura non tiene infatti conto della frequenza dello stesso termine nel resto della collezione (*corpus*): poiché tutti gli obiettivi descritti nell'introduzione prevedono l'identificazione di elementi discriminanti tra documenti (per il *clustering* la distanza, per la classificazione le differenze tra *labels*), tale misura non risulta efficiente.

Una seconda metrica d'interesse, strettamente legata alla prima ma capace di rispondere in parte alla criticità appena descritta, é la **Term-Frequency Inverse-Document-Frequency** (TFIDF). La TFIDF ha la caratteristica di pesare la TF di ciascun documento per una matrice di frequenza riferita all'intera collezione (DF). Come conseguenza la TFIDF di ciascun token aumenterà con la frequenza dello stesso all'interno del documento, e diminuirà con l'aumentare dello stesso nell'intera collezione. Una TFIDF particolarmente elevata indicherà un termine in proporzione molto più frequente in un documento rispetto al resto della collezione. Tale caratteristica rende la TFIDF abbastanza adeguata a problemi che implicano la discriminazione tra documenti. Se da un lato ha il vantaggio di essere computazionalmente poco onerosa e più efficiente della semplice TF, anche la TFIDF porta con sé alcune mancanze. La principale è l'incapacità del metodo di catturare qualsiasi elemento afferente alla sfera semantica, ma si limita a descrivere la realtà lessicale; inoltre la TFIDF non fornisce informazioni circa la posizione dei singoli *token* all'interno del testo. Un ultimo punto da evidenziare è che un termine (o collezione) molto ricorrente non necessariamente fornisce informazioni interessanti per descrivere un testo. Poiché, per quanto possibile, lo studio cerca di tenere conto della realtà semantica (estremamente importante quando si analizzano dialoghi), neanche tale metrica risulta la più adatta.

L'ultima tipologia di *features* afferisce al mondo del **Word Embedding** (WE). La principale caratteristica che rende tali tecniche particolarmente interessanti per il *Natural Language Processing* (NLP) è la capacità di catturare sia informazioni sintattiche, sia informazioni semantiche. Per fare ciò viene generato un apposito spazio vettoriale in cui le parole più simili, convertite in vettori, risultano più vicine. Di conseguenza quando alcuni termini compaiono spesso nello stesso contesto linguistico, il WE fa sì che vengano identificati come semanticamente simili, e di conseguenza li rappresenta con vettori vicini nello spazio. Vengono presentati di seguito in ordine di efficienza 3 metodi di Word Embedding implementati:

- **Word2Vec**: è uno dei primi metodi di WE e consiste di una semplice rete neurale a 2 strati. Inizialmente ogni termine viene convertito in formato *One-Hot Encoding* (OHE), diventando dunque una lista di 0 e 1 che ne determinano la posizione all'interno del corpus; tale caratteristica risulta poco efficiente quando si ha a che fare con testi molto lunghi, dotati quindi di vocabolari particolarmente ampi (come nel caso dei copioni). A questo punto il testo in forma OHE viene utilizzato come input per una rete neurale che sfrutta la *Backpropagation* per addestrare i pesi. Finita la fase di *training* viene generata in uscita la rappresentazione WE dei *tokens*. Nell'ambiente *Python* tale tecnica viene implementata dalla libreria *gensim* e in particolare dalla funzione *Doc2Vec* che lavora direttamente sui documenti invece che sui termini.

- **Singular Value Decomposition (SVD)**: Sfrutta le proprietà dell'algebra lineare per scomporre opportunamente la Document-Term Matrix in vettori di componenti principali a lunghezza arbitraria. In riferimento allo studio viene applicata la Truncated SVD adatta a lavorare con matrici sparse per ridurre la dimensionalità, l'algoritmo di T-SVD viene applicata su entrambe le document-term matrix a nostra disposizione, avremo quindi una rappresentazione sia per la la matrice con TF che con quella TFIDF, e i risultati vengono impiegati come *features*.

- **GloVe**: sia per la parte di classificazione che di segmentazione è il metodo che ha consentito di raggiungere i migliori risultati. Come i precedenti metodi mappa i termini in spazi vettoriali: sfrutta un modello pre-trainato per assegnare dei vettori alle singole parole, la rappresentazione finale di un documento viene ottenuta operando una media tra tutti i vettori dei termini contenuti nel documento, questi vettori media vengono quindi utilizzati come *features*.

Supervised Learning

La prima tipologia di modelli impiegati fa riferimento alla classe degli algoritmi di *Supervised Learning*; in particolare si è deciso di sfruttare le *features* (le quali rappresentano i documenti dopo il *Word Embedding*) estratte precedentemente come input alla classificazione per prevedere il genere di ciascun film. Poiché i generi cinematografici sono molti (circa 12 per i film selezionati) si è deciso di ridurre la dimensionalità del problema di classificazione a 5 *labels*: film d'azione, drammatici, commedie, horror e documentari. Si è optato per mantenere tali generi in quanto la maggior parte dei film era concentrata in tali categorie. Inoltre, al fine di ottenere un campione bilanciato, viene applicata una procedura di *undersampling* la quale consente di ridurre la dimensionalità delle classi predominanti (in particolare film drammatici e commedie) riportando ciascuna categoria a circa 1200 unità. Infine il campione è stato suddiviso in *training set* e *validation set* al fine di addestrare e ottimizzare i parametri.

Definita la dimensionalità del problema di classificazione è stato necessario valutare quale tra le *features* estratte fosse maggiormente efficiente nel discriminare tra i diversi generi. Per la selezione si è deciso di sfruttare un algoritmo di classificazione semplice e computazionalmente poco oneroso, ovvero il **K-Nearest Neighbour** (KNN); tale classificatore, essendo tra i più semplici e meno performanti, ha consentito dunque di fissare una soglia minima di accuratezza utilizzata come metro di misura dei miglioramenti dei successivi metodi. Il KNN è stato quindi applicato su tutte le *features*, è stata valutata l'accuratezza e il GloVe ha mostrato i migliori risultati, di conseguenza viene utilizzato per tutti i successivi classificatori. II risultati del KNN sono riassunti nella tabella 4.1:

	Precision	Recall	F1-measure	Support
Action	0.59	0.77	0.67	224
Comedy	0.45	0.77	0.57	239
Documentary	0.95	0.75	0.84	243
Drama	0.51	0.31	0.38	259
Horror	0.74	0.47	0.58	214

Tabella 4.1: KNN: Precision, Recall, F1-Measure

con un *accuracy* totale sul test di circa 0.61. Salta subito all'occhio come vi sia una sostanziale disomogeneità tra i risultati sui diversi generi, con il genere Drama che mostra il peggior andamento.

Il secondo classificatore impiegato è un **Random Forest** con 500 *trees* e funzione di Gini per la misurazione della qualità degli *split*. Per prestazioni (riassunte di seguito nella tabella 4.2) si posiziona subito dopo il KNN, mostrando tuttavia sia un' accuratezza più elevata, pari a circa 0.67, sia una distribuzione di metriche più omogenea (come si può notare dalle F1-*measures*). Nonostante il miglioramento appena descritto tale classificatore non può essere considerato ancora efficiente.

	Precision	Recall	F1-measure	Support
Action	0.69	0.71	0.70	420
Comedy	0.60	0.65	0.63	382
Documentary	0.84	0.87	0.85	380
Drama	0.52	0.41	0.46	380
Horror	0.66	0.70	0.68	384

Tabella 4.2: Random Forest: Precision, Recall, F1-Measure

Il terzo classificatore, nonché quello che ha mostrato le migliori prestazioni, è una **Support Vector Machine**; essendo un metodo più complesso dei precedenti, poiché comporta un *setting* dei parametri più ampio, è stata applicata una procedura di *parameters tuning* attraverso una *cross-validation*. Come risultato il miglior set di parametri è stato: parametro di regolarizzazione C (penalità l_2) pari a 3 e *Radial basis function kernel* (rbf, risulta anche computazionalmente meno oneroso del kernel lineare e polinomiale). Le metriche di *performance* vengono riassunte nella tabella 4.3.

	Precision	Recall	F1-measure	Support
Action	0.70	0.75	0.72	224
Comedy	0.64	0.70	0.67	239
Documentary	0.89	0.88	0.88	243
Drama	0.63	0.53	0.58	259
Horror	0.73	0.76	0.74	214

Tabella 4.3: SVM (kernel rbf): Precision, Recall, F1-Measure

Sul *test set* ottiene i migliori risultati in termini di *accuracy* con un risultato pari a 0.73, e riesce in parte a risolvere il problema della disomogeneità tra generi.

Il quarto classificatore è una **Shallow Artificial Neural Network** (ANN) con 3 strati *Fully connected* con rispettivamente 128, 64 e 32 nodi (i primi due con attivazione relu, l'ultimo con attivazione sigmoidale). Poiché si è notata una tendenza di *overfitting* sul *training set* sono stati aggiunti uno strato di *BatchNormalization* e 2 strati di *dropout*, ottenendo così una prestazione su *test set* assimilabile a quella del *training* (come mostra l'immagine 4.2).

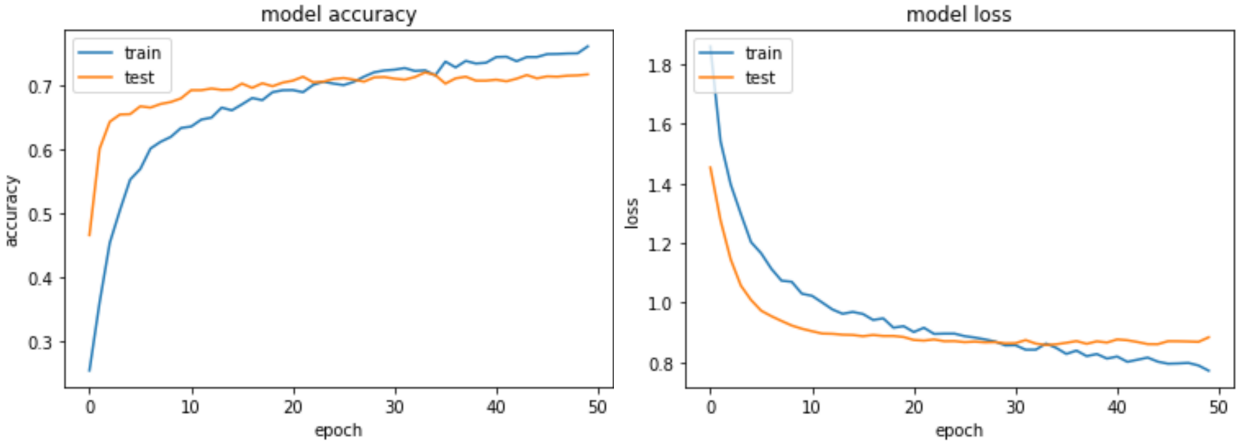

Figura 4.1: ANN loss e accuracy

Le prestazioni della rete vengono mostrate nella tabella 4.4:

	Precision	Recall	F1-measure	Support
Action	0.71	0.76	0.73	420
Comedy	0.64	0.70	0.67	382
Documentary	0.90	0.88	0.89	380
Drama	0.59	0.45	0.51	380
Horror	0.72	0.76	0.75	384

Tabella 4.4: Shallow ANN: Precision, Recall, F1-Measure

la quale ottiene un'accuratezza totale pari a 0.72, posizionandosi al secondo posto tra tutti i classificatori.

L'ultima tipologia di classificatore è una **Convolutional Neural Network** (CNN) composta da 2 strati convoluzionali con attivazione relu, uno strato di *max pooling* e due strati *fully connected* (rispettivamente 100 nodi attivazione relu, 50 nodi attivazione sigmoidale); anche in questo caso per prevenire il rischio di *overfitting* vengono inseriti 2 *layers* di *dropout*.

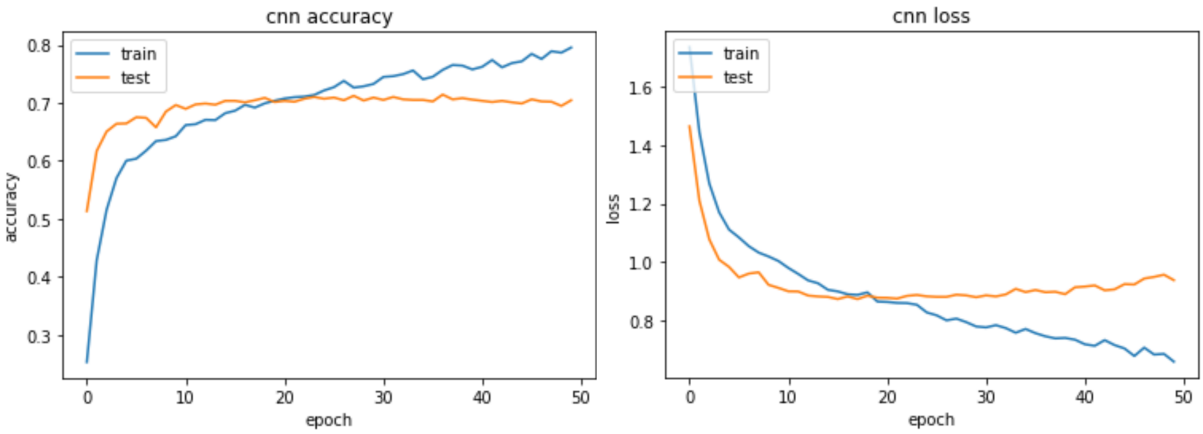

Figura 4.2: CNN loss e accuracy

La rete per sua natura risulta la più onerosa in termini computazionali e, non portando miglioramenti rispetto alla semplice ANN e alla SVM, non viene considerata efficiente. Ottiene un *accuracy* pari a 0.71 e metriche di *performance* riassunte nella tabella 4.5:

	Precision	Recall	F1-measure	Support
Action	0.72	0.72	0.72	420
Comedy	0.63	0.69	0.66	382
Documentary	0.89	0.89	0.89	380
Drama	0.58	0.44	0.50	380
Horror	0.70	0.79	0.74	384

Tabella 4.5: CNN: Precision, Recall, F1-Measure

In conclusione si può affermare che il miglior classificatore risulta essere la SVM, non solo per prestazioni generali, ma soprattutto per aver mostrato una *performance* marginale sui generi maggiormente omogenea. Tale metodo risulta inoltre molto veloce in termini computazionali

e facile da implementare, il che lo rende preferibile alle reti neurali (maggiormente complesse da definire e computazionalmente più onerose). Nonostante i risultati in termini di accuratezza non siano stati particolarmente elevati la complessità del problema li rende più che accettabili.

Unsupervised Learning

Per la parte di Unsupervised Learning utilizziamo il **K-Means Clustering** per valutare quanto le rappresentazioni vettoriali dei documenti in nostro possesso siano più o meno in grado di separare i film in base al genere, ulteriore focus è stato quello di trovare il modo migliore per creare delle visualizzazioni degli embedding in spazi a due e tre dimensioni. I vettori che rappresentano i nostri Script vengono passati al K-Means che utilizzerà la distanza euclidea per raggruppare i film in 5 Clusters, per visualizzare spazialmente la disposizione dei copioni è stata utilizzata la tecnica **t-distributed stochastic neighbor embedding (t-SNE)**, questo algoritmo opera in due fasi, in primo luogo viene costruita una distribuzione di probabilità nello spazio ad alta dimensionalità dove ad ogni coppia di punti viene associato un valore direttamente proporzionale alla loro similarità, successivamente viene definita una seconda distribuzione di probabilità nello spazio a bassa dimensionalità che ci interessa, l'algoritmo quindi minimizza la differenza tra le due distribuzioni tramite discesa del gradiente trovando così la disposizione spaziale più adatta a emulare la controparte reale ad alta dimensionalità. Elenchiamo i risultati delle analisi in base alla rappresentazione utilizzata. Per valutare la bontà del clustering vengono utilizzate le misure interne dell'indice di **Silhouette** e della **Purity**, mentre verrà utilizzato l'**Adjusted Rand Score** come misura esterna essendo le classi di appartenenza dei singoli cluster conosciute a priori.

Vediamo in tabella 5.1 i risultati per i due indici di Rand e per l'indice di Silhouette:

Representation	Rand Index	Adjusted Rand Index	Silhouette Index
1 Doc2Vec	0.731	0.22	0.015
2 GloVe	0.682	0.102	0.066
3 TFIDF SVD	0.671	0.128	0.005
4 TF SVD	0.562	0.077	0.074

Tabella 5.1: Clustering Scores

Analizzando i risultati si può subito scartare la rappresentazione SVD basata su TF in quanto è quella che performa peggio sia per quanto riguarda il Rand Index che la sua versione Adjusted, osservando la tabella 5.2 della Purity dei cluster appare evidente il pessimo risultato:

Cluster	Main Genre	Purity	Clust. size
0	Action	46%	191
1	Horror	28%	3405
2	Comedy	41%	1554
3	Documentary	93%	581
4	Documentary	100%	4

Tabella 5.2: SVD-TF Purity score

SVD basato su TFIDF (Tabella 5.3) e GloVe (Tabella 5.4) si piazzano nella posizione intermedia, i risultati per queste due rappresentazioni mostrano Rand molto simile, ma la versione Adjusted ci indica la SVD sul TFIDF come leggermente migliore

Cluster	Main Genre	Purity	Clust. size
0	Documentary	84%	625
1	Comedy	28%	2074
2	Documentary	89%	390
3	Horror	32%	1519
4	Action	40%	1127

Tabella 5.3: GloVe Purity score

Cluster	Main Genre	Purity	Clust. size
0	Horror	55%	11
1	Action	27%	1872
2	Comedy	32%	1122
3	Documentary	94%	675
4	Comedy	25%	2055

Tabella 5.4: SVD con TFIDF Purity score

Viene riportata di seguito (immagine 6.1) una visualizzazione del corpus con le *labels* assegnate dal SVD TFIDF e le reali label dei dati:

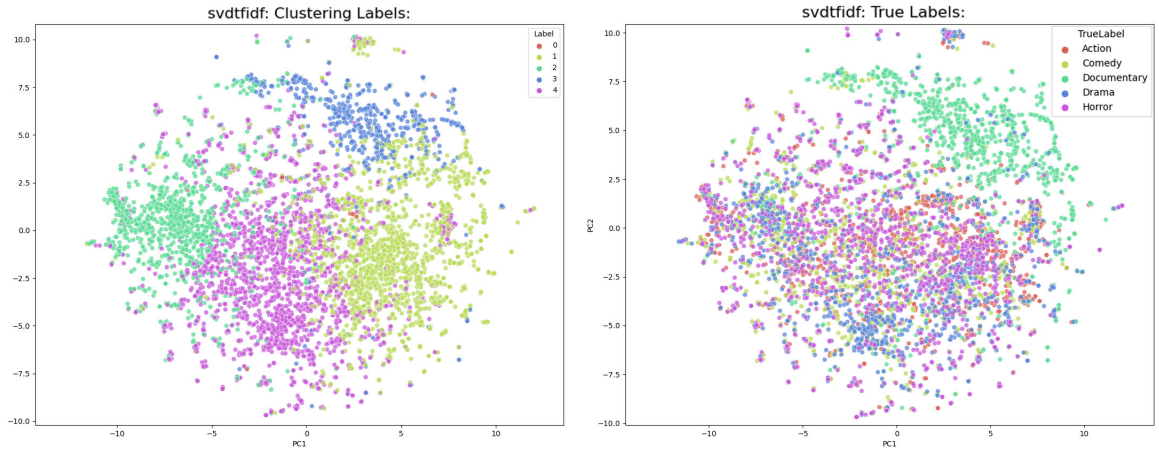

Figura 5.1: SVD-TFIDF Clustering result

Per quanto riguarda la rappresentazione basata sul modello *pre-trained* di Doc2Vec si ottengono i migliori risultati in assoluto; questa rappresentazione infatti permette a K-Means di ottenere un *Adjusted Rand Index* quasi doppio rispetto all'opzione del SVD su TFIDF appena vista; interessante notare come le due rappresentazioni migliori presentino gli indici di *Silhouette* più bassi, a riprova del fatto che i 5 cluster sono quasi completamente sovrapposti ad eccezione della classe dei documentari che riesce a staccarsi dal resto anche visivamente nella proiezione a 2 dimensioni che ci fornisce t-SNE. Viene infine riportata la Tabella 5.5, la quale riassume i risultati del *Clustering* con Doc2Vec in qualità di miglior metodo sviluppato.

Si può notare come la rappresentazione Doc2Vec presenti dei pattern più efficaci riuscendo ad ottenere un ottima separazione per i documentari e una separazione media generalmente più alta per gli altri cluster rispetto agli altri metodi. Data la difficoltà del task le *features* estratte

Cluster	Main Genre	Purity	Clust. size
1	Action	45%	276
2	Horror	46%	1538
3	Drama	28%	1337
4	Comedy	50%	1722
5	Documentary	92%	862

Tabella 5.5: Doc2Vec Purity score

non producono comunque un risultato soddisfacente, in questo senso è evidente che senza una fase di *learning*, come visto nella sezione *Supervised* (Sez.4), sia quasi impossibile distinguere tra i generi più comuni affidandosi ad un semplice algoritmo di *clustering*. L'utilizzo di altri algoritmi come l'*Agglomerative Clustering* e il DBSCAN hanno evidenziato inoltre risultati peggiori del K-Means.

Viene riportata in conclusione in figura 5.2 la rappresentazione t-SNE dei vettori Doc2Vec con le reali labels del miglior *embedding* possibile.

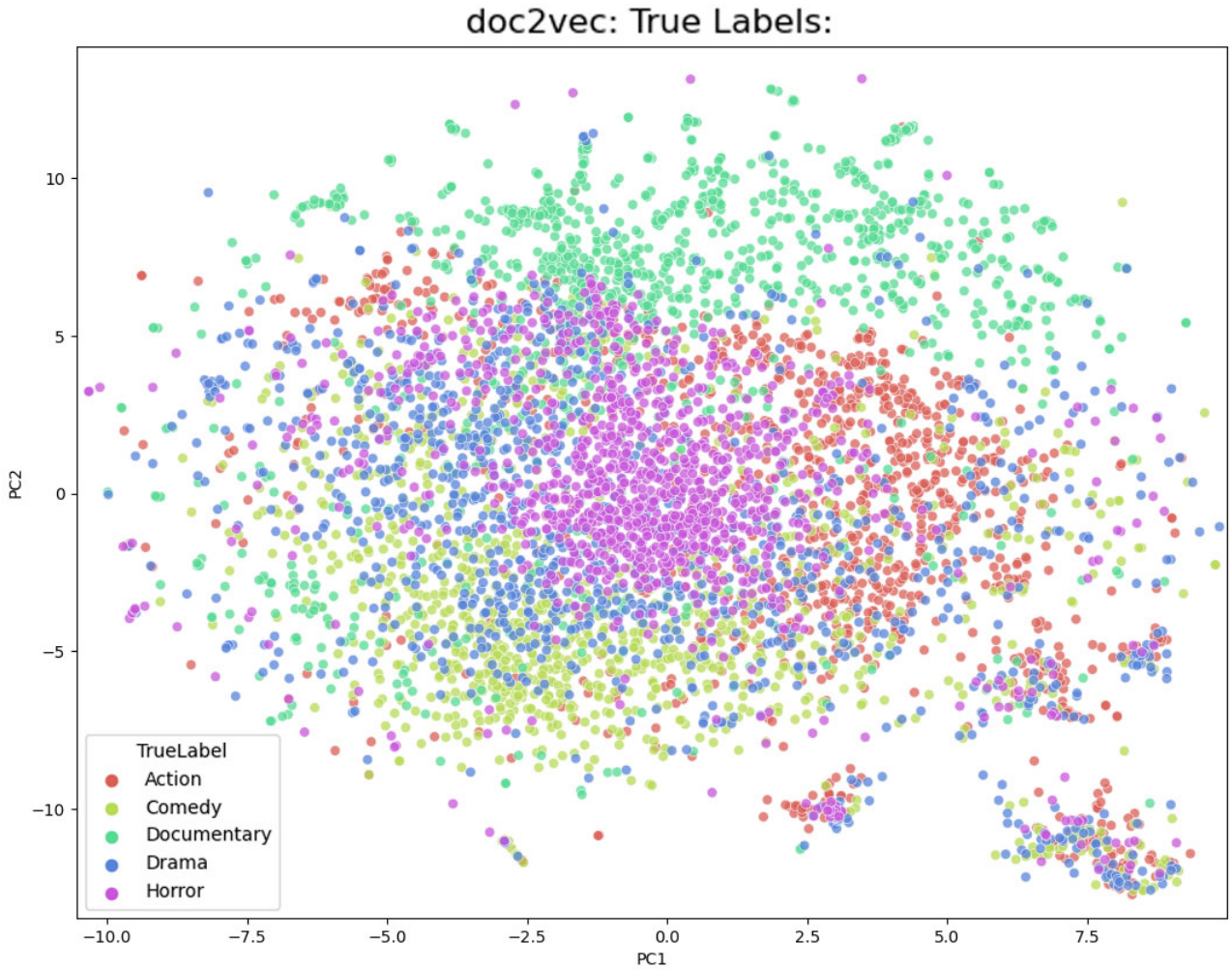

Figura 5.2: Doc2Vec Real labels

Topic Modelling

Con l'intento di vedere quali siano gli argomenti principali trattati nei copioni, è stato costruito un *Topic Model*. In particolare, dopo una prima fase di *pre-processing*, è stato costruito un modello **LDA *Latent Dirichlet Allocation***. LDA categorizza i documenti attraverso un modello probabilistico generativo; in dettaglio, dato un documento, cerca di trovare quali *topic* possono aver generato i termini al suo interno. Per la costruzione del modello viene utilizzato ***Gensim***, che è una libreria Python per il *topic modelling*, *document indexing* e il *similarity retrieval*. Poiché il modello LDA richiede a priori di specificare il numero di *topic* da restituire è stata scelta una numerosità pari a 10; questo dopo aver effettuato varie prove e testato il modello con alcuni indicatori di performance. Le metriche utilizzate per descrivere le performance del modello sono la *Perplexity* e la *Coherence*. La prima è una misura di incertezza, minore è il valore e migliore sarà il modello. La seconda è una misura di similarità semantica tra le parole più significative del modello, maggiore è il suo valore tra le *top words* e migliori saranno le performance del modello, in quanto sono più interpretabili. Il modello realizzato ha le seguenti performance:

- *Perplexity*: -8.48;
- *Coherence Score*: 0.4;

I *topic* trovati dal modello sono i seguenti:

Figura 6.1: Topics individuati

I risultati non paiono essere ottimali in quanto non vi è una marcata differenza delle parole utilizzate in ognuno di essi. Questo risultato era abbastanza prevedibile, poiché sono testi di conversazioni, quindi vi sono alcune parole che prevaricano. Quello che si evidenzia è la

differenza di linguaggio in alcuni copioni, come ad esempio il *topic* 7 mostra un linguaggio più scurrile.

Conclusioni e possibili sviluppi

Nonostante il problema complesso, lo studio di *text mining* ha consentito di applicare e confrontare alcuni metodi differenti; seppur non vi siano stati grandi risultati in nessuna delle 3 parti in cui si suddivide l'analisi, sono stati identificati alcuni *pattern* interessanti:

- Il processo di *features extraction* ha consentito di sviluppare sia metodi *Bag of word*, sia di *Word Embedding*; la seconda tipologia si è mostrata migliore per affrontare i *task* di classificazione e segmentazione, con il GloVe che si attesta come miglior metodo.
- E' necessario fare una considerazione circa il genere dei film: nonostante a ciascun film sia stato assegnato il principale genere, è possibile che un film possa appartenere a più classi (commedia horror, film d'azione drammatico,...). Si ritiene che tale condizione abbia influito in maniera significativa sui risultati.
- Sia la classificazione che la segmentazione hanno identificato il genere documentario come il più discostato dal resto del gruppo, per la prima ottenendo delle elevate *recall* e *precision*, per la seconda posizionandosi in maniera relativamente distante dagli altri generi (rispetto al grafico delle prime 2 componenti principali).
- La classificazione ha consentito di ottenere un'*accuracy* massima di 0.73 grazie all'algoritmo *Support Vector Machine* (*classifier*); tale metodo risulta anche avere le *performance* più omogenee tra le classi (si guardi tabelle Sez.4). Le 2 reti neurali, convoluzionale e non, hanno mostrato un'accuratezza vicina alla SVM, tuttavia risultano computazionalmente più onerose e i risultati tra i generi sono meno omogenei. Seppur l'*accuracy* non sia particolarmente elevata il confronto tra diverse tecniche di *features extraction* e di *supervised learning* portano a credere che la difficoltà del *task* impedisca di raggiungere risultati molto migliori.
- Per quanto concerne la segmentazione (si veda la Sez.5) purtroppo non si sono ottenuti cluster consistenti rispetto ai generi: infatti la corrispondenza cluster-genere si verifica con una frequenza molto ridotta. Nonostante ciò il grafico delle *features* rispetto alle prime 2 componenti principali ha evidenziato che alcuni generi si discostano maggiormente degli altri (si veda la figura 5.2), come ad esempio il genere documentario e commedia.
- Il *topic modelling* non ha evidenziato particolari differenze tra i termini dei diversi *topics* se non per il 7 che contiene molti termini scurrili.

Lo sviluppo di nuove tecniche sia per la parte di *features extraction* sia per la parte di *modelling* potrebbe sicuramente portare miglioramenti nei risultati. Inoltre l'utilizzo di una nuova label (riguardante ad esempio la dimensione temporale) potrebbe portare a risultati migliori nei primi due task.