Classificazione di sottogeneri musicali tramite tecniche di text-mining

Lorenzo Mammana 807391 Gabriele Molteni 800900 Matteo Scarpone 800900

***Sommario*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduzione

Numerosi studi in letteratura sono stati effettuati riguardo la classificazione automatica di generi musicali sfruttando le liriche delle canzoni. Ci si aspetta che generi musicali diversi contengano tematiche e linguaggi molto differenti tra di loro e quindi la classificazione dovrebbe essere possibile. In questo studio si è deciso di verificare la possibilità di estendere questa analisi andando a valutare la possibilità di distinguere sottogeneri musicali di uno stesso genere. In particolare ci si è concentrati sull'analisi di sottogeneri della musica metal, questo tipo di musica presenta una enorme varietà di sottogeneri musicali, ciascuno con il proprio stile linguistico e le proprie tematiche. La classificazione delle liriche è stata effettuata utilizzando tecniche di text-preprocessing e text-mining. Uno dei metodi più noti per la classificazione è quello basato su Latent Dirichlet Allocation (LDA), questo tipo di metodo permette di eseguire l'inferenza di *Topic* all'interno di un testo in maniera non supervisionata. Avendo a disposizione dati etichettati con uno o più sottogeneri si è utilizzato una versione più moderna di LDA, denomita LLDA (Labeled LDA) che permette di etichettare i *topic* ottenendo una classificazione supervisionata.

II. Dataset

Il dataset è stato costruito eseguendo lo scraping di due diversi siti web Darklyrics e Metal Archives, il primo sito è una collezione di liriche, mentre il secondo è stato utilizzato per integrare le informazioni riguardo al genere degli artisti. La prima versione del dataset prodotta è composta da 259166 liriche, ogni lirica è composta dai seguenti campi:

- band - Il nome del gruppo compositore
- album - L'album in cui è contenuta
- year - L'anno di uscita
- song - Il nome della canzone
- lyrics - Le liriche della canzone
- genre - Il genere o i generi della canzone
- lang - La lingua della canzone

Il dataset estratto contiene numerosi campi mancanti come mostrato in tabella I:

Tabella I: Attributi mancanti e unici

Attributo	Mancanti	Valori unici
band	0	8456
album	11	26089
year	46	73
song	784	194215
lyrics	17891	226154
genre	55929	1694
lang	0	37

Tutte le righe con attributi mancanti sono state eliminate dal dataset utilizzato per la classificazione finale. La lingua della lirica viene prodotta con un algoritmo in grado di calcolare la percentuale di appartenza di un testo ad una certa lingua, in tabella I il numero di *lang* mancanti è nullo solamente perchè viene utilizzato il valore MISSING come placeholder. In tabella II viene invece mostrata la distribuzione delle prime dieci lingue contenute nel dataset.

Tabella II: Top 10 frequenza lingue

attributo	frequenza
en	0.796778
MISSING	0.069033
ro	0.041854
de	0.025347
es	0.019949
fi	0.007833
pt	0.005483
fr	0.005336
sv	0.004460
no	0.004121

Per la classificazione finale sono state mantenute esclusivamente le liriche in lingua inglese, le altre lingue sono troppo poco frequenti per poter essere utilizzate e sarebbero un fattore confondente troppo grande per gli algoritmi di classificazione. La rimozione delle liriche mancanti o non in inglese riduce il numero di righe a 200556.

A. Gestione del genere musicale

Come si vede in tabella I il numero di generi univoci presenti all'interno del dataset è estremamente elevato, questo è dovuto al fatto che i dati estratti tramite scraping non sono ben categorizzati, ma possono presentare leggere variazioni o ordinamenti differenti. Per poter rendere fattibile la classificazione è stata utilizzata la conoscenza del dominio per raggruppare i sottogeneri in macro-sottogeneri, in particolare la scelta è stata quella di fissare nove generi principali a cui vengono ricondotte tutte le liriche nle dataset. Questo introduce ovviamente un bias in quanto la categorizzazione è basata esclusivamente su conoscenza a priori riguardante il genere, senza essere influenzata dall'effettivo contenuto delle liriche. Un secondo bias è relativo al fatto che i generi sono associato al gruppo musicale e non al singolo album, ciò che accade nella realtà è che un artista produce album di diversi generi, mentre invece nel dataset questa distinzione non è evidente.

I nove generi proposti sono i seguenti:

- Rock
- Heavy
- Thrash
- Power
- Folk
- Progressive
- Death
- Doom
- Black

I generi sono stati selezionati in base alle pre-conoscenze degli autori in modo tale che possa essere stabilito un ordine tra di essi; nella lista precedente ogni genere è teoricamente contenuto all'interno dell'insieme composto dal genere successivo. Vengono prodotti due dataset utilizzati per il task di classificazione, il primo è un dataset multi-label, mentre il secondo utilizza l'ordinamento per associare una singola label come attributo di output. I dataset prodotti sono fortemente sbilanciati, il dataset multilabel può teoricamente rappresentare $2^9 - 1 = 511$ generi diversi, nel dataset sono presenti solamente 110 di queste possibili combinazioni. Come mostrato in tabella III sono solamente 18 i generi con frequenza superiore all'1%, ci si aspetta quindi che la classificazione su questo dataset sarà particolarmente complicata.

Tabella III: Generi con frequenza superiore all'1%

Generi	Frequenze
{Death Metal}	0.214409
{Black Metal}	0.097342
{Power Metal}	0.072505
{Thrash Metal}	0.063777
{Heavy Metal}	0.058732
{Death Metal, Thrash Metal}	0.050035
{Doom Metal}	0.046568
{Death Metal, Doom Metal}	0.037722
{Progressive Metal}	0.031522
{Heavy Metal, Power Metal}	0.031199
{Death Metal, Black Metal}	0.027881
{Heavy Metal, Rock}	0.024937
{Thrash Metal, Power Metal}	0.019998
{Heavy Metal, Thrash Metal}	0.019103
{Progressive Metal, Power Metal}	0.014146
{Doom Metal, Black Metal}	0.012251
{Doom Metal, Heavy Metal}	0.010021

Nel caso singola-label il dataset rimane piuttosto sbilanciato, ma la classificazione dovrebbe essere nettamente più semplice da eseguire. Le frequenze sono mostrate in tabella IV.

Tabella IV: Frequenze generi singola label

Genere	Frequenza
Death Metal	47379
Black Metal	28444
Power Metal	21019
Doom Metal	19413
Thrash Metal	13341
Rock	11101
Heavy Metal	9454
Progressive Metal	8757
Folk Metal	2060

É interessante notare l'andamento dei generi nel tempo, il dataset fornisce una buona copertura dei generi per anno, in particolare si vede, come è accaduto realmente, un aumento della "pesantezza" del genere musicale con il passare degli anni (Figura 1).

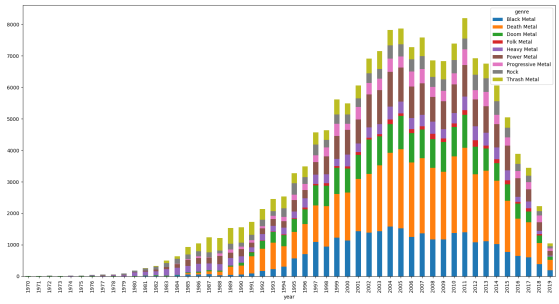


Figura 1: Distribuzione dei generi musicali negli anni

III. Preprocessamento del testo

Le liriche estratte inizialmente sono molto sporche, contengono punteggiatura, parole poco o troppo frequenti etc... Per poter eseguire una classificazione corretta è necessario procedere con un'operazione di filtraggio del testo per ottenere dei *token* ben formati. Per farlo si è proceduto costruendo un workflow Knime utilizzato come wrapper per il codice Python, la scelta di utilizzare Python è dovuta al consumo estremamente elevato di risorse da parte di Knime sul dataset in questione, le operazioni che in Python richiedono pochi minuti con un consumo limitato di memoria in Knime necessitano di un numero estremamente più elevato sia di tempo che di memoria.

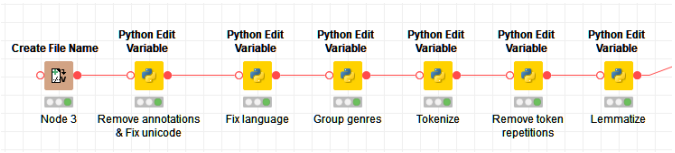


Figura 2: Workflow preprocessamento del testo in Knime

A. Tokenizzazione

La tokenizzazione è stata eseguita con il package *nltk* (Natural language toolkit), uno dei più utilizzati in questo ambito. Il primo passo è quello di separare ogni lirica nelle frasi che la compongono, questo viene ottenuto per mezzo di un modello supervisionato di *Sentence boundary disambiguation* (SBD) preaddestrato su un insieme molto ampio di testi. L'applicazione di questo modello e la seguente rimozione della punteggiatura produce un risultato di questo tipo:

```
Searching the darkness. And emptineeess. I'm hiding away from the sun
['Searching the darkness', 'And emptineeess', 'I m hiding away from the sun']
```

Si procede poi all'estrazione vera e propria dei singoli token, questa viene eseguita tramite l'utilizzo di espressioni regolari, in particolare utilizzando il *Penn Treebank*. Questo permette di rimuovere token troppo frequenti basati sulla grammatica inglese e risolvere le contrazioni di alcune parole. I token vengono poi convertiti in minuscolo, vengono rimossi i numeri e le parole più corte di tre caratteri.

```
['Searching the darkness', 'And emptineess', 'I m hiding away from the sun']
['searching', 'darkness', 'emptineeess', 'hiding', 'away', 'sun']
```

Viene poi eseguita la rimozione delle ripetizioni nelle parole, l'algoritmo utilizzato rimuove iterativamente le lettere multiple dai token e verifica su un dizionario se la parola esiste o meno, in modo tale da decidere se continuare l'eliminazione o fermarsi. Si è proceduto verificando quali sono lo parole più frequenti per ogni genere, si nota che le dieci parole più frequenti sono piuttosto simili all'interno di ogni genere come visibile in tabella V.

```
['searching', 'darkness', 'emptineeess', 'hiding', 'away', 'sun']
['searching', 'darkness', 'emptines', 'hiding', 'away', 'sun']
```

In questo caso l'algoritmo sbaglia sulla parola *emptiness* andando a rimuovere la seconda s.
Si è poi analizzata la frequenza dei token per genere, come visibile in tabella V i primi dieci token di ogni genere sono piuttosto simili, questo è comunque corretto dato che i sottogeneri fanno tutti parte dello stesso insieme iniziale.

Tabella V: Top 10 parole frequenti per genere

Rock	**Heavy**	**Thrash**	**Power**	**Folk**	**Progressive**	**Death**	**Doom**	**Black**
away	got	death	away	blood	away	blood	away	black
got	know	know	know	come	eyes	death	eyes	blood
know	life	life	life	land	know	eyes	know	death
life	like	like	never	life	life	life	life	eyes
like	never	never	night	like	like	like	like	life
love	night	one	one	never	never	never	never	night
never	one	see	see	night	one	one	one	one
one	see	time	time	one	see	see	see	see
see	take	way	way	see	time	time	time	time
time	time	world	world	time	world	world	world	world

In totale vengono estratti 209780 token differenti.

B. Lemmatizzazione

Per ridurre ulterirormente il numero di token estratti si è proceduto eseguendo il loro stemming, cercando quindi di ricondurre token simili alla loro radice comune. É stato utilizzato il *LancasterStemming*, un tipo di stemming molto "aggressivo" che riduce di molto la lunghezza dei token, la scelta dell'utilizzo di questo algoritmo è dovuta principalmente alla grande dimensione del dataset, si è deciso di sacrificare leggermente la leggibilità dei dati per ottenere un aumento di velocità nell'esecuzione degli algoritmi. Sulla frase mostrata in precedenza l'algoritmo produce i seguenti risultati:

```
['searching', 'darkness', 'emptineeess', 'hiding', 'away', 'sun']
['search', 'dark', 'emptin', 'hid', 'away', 'sun']
```

L'algoritmo in questo caso sembra funzionare correttamente, applicato all'intero dataset il numero di token viene ridotto fino a 136261.

C. Filtraggio dei token per frequenza

Il numero di token contenuti nel dataset resta estremamente elevato, è fondamentale rimuovere i token rari, poichè sono poco influenti per il task di classificazione. Anche in questo caso si è preferito utilizzare Python invece di Knime in quanto il primo richiede all'incirca due minuti, mentre il secondo circa quaranta minuti. L'algoritmo utilizzato sfrutta il concetto di *Multiset* per filtrare in maniera estremamente rapida i token, permettendo però di mantenere le ripetizioni che potrebbero essere utili in fase di classificazione. I token vengono filtrati mediante l'utilizzo di una soglia così definita:

$$T = \gamma * N \tag{1}$$

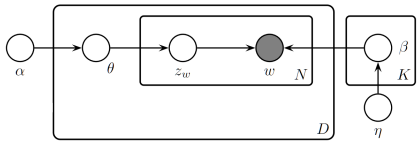
Il parametro γ regola la percentuale di token da filtrare, per la classificazione è stato utilizzato $\gamma = 0.01$ in modo tale da filtrare tutti quei token che compaiono in meno dell'1% delle liriche. Il filtraggio produce 1305 token, questo sta a significare che più del 99% dei token sono estremamente poco frequenti!

IV. Classificazione

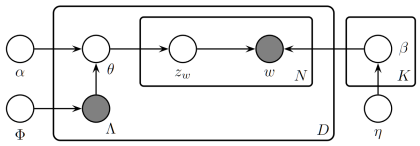
La classificazione segue due procedimenti distinti nel caso di singola label o di label multipla. In comune vi è il fatto che il dataset viene suddiviso in 80% training e 20% test.

A. Classificazione multi-label

1) Labeled LDA: Il primo modello utilizzato per la classificazione multi-label è un LLDA, la scelta di utilizzare un algoritmo di questo tipo è evidente, il semplice LDA, essendo un algoritmo non supervisionato non si presta bene per una classificazione di questo tipo, inoltre ogni lirica è etichettata e questo ci porta a ragionare in termini di classificazione supervisionata piuttosto che non. Rispetto al tradizionale approccio utilizzato da LDA, questo tipo di algoritmo restringe la distribuzione delle parole invece che su tutti i topic, solamente su quelli che rispettano l'etichetta presente nel training set. Questo vincolo è mostrato in maniera grafica in figura 3.



(a) LDA tradizionale



(b) Labeled LDA

Figura 3: Differenza tra modello non supervisionato e supervisionato

Il modello presenta una minima differenza con LDA, il training ed il testing vengono eseguiti utilizzando un campionamento di Gibbs collassato, esattamente come accade per il modello tradizionale. Il modello viene addestrato per 100 epoche, con parametri $\alpha = 0.1/9$ e $\beta = 1$. La scelta dei parametri è influenzata dal fatto che un valore piccolo di α dovrebbe portare il modello a classificare più frequentemente i campioni con pochi topic contemporaneamente, inoltre dai test sperimentali è la combinazione di parametri che ottiene la likelihood migliore sui dati. Le metriche utilizzate per la valutazione sono le classiche Micro-average, Macro-average e due metriche pensate per task di questo tipo ovvero la

Hamming-loss e la Exact Match Ratio. La prima è definibile come la frazione di label non correttamente classificate:

$$HL = \frac{1}{|N| * |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} xor(y_i j, z_i j) \tag{2}$$

Ed è utile per verificare se il modello non sta classificando in maniera completamente casuale, la seconda invece verifica il numero di multilabel perfettamente classificate:

$$EMR = \frac{1}{|N|} \sum_{i=1}^{|N|} I(Y_i = Z_i) \tag{3}$$

I risultati sono riportati in tabella VI:

Tabella VI: Risultati LLDA

F1-Mi	F1-Ma	HL	EMR
0.3300	0.3200	0.3158	0.0318

Come ci si aspettava i risultati sono molto bassi, il dataset è complesso ed estremamente sbilanciato, questo è abbastanza per giustificare il fallimento del modello. E' relativamente buono il valore della HL, il modello sembra apprendere delle differenze sensate, l'EMR mostra invece come non sia praticamente mai in grado di ottenere una classificazione perfetta di tutte le label correttamente.

2) Percettrone multistrato: Un altro modello utilizzato in letteratura per la classificazione multi-label è il percettrone multistrato (MLP), l'architettura scelta è stata ottenuta in maniera sperimentale ed è così definita (Tabella VII):

Tabella VII: Architettura MLP

Layer	Output	Param #
dense (Dense)	(None, 256)	334336
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
dense_2 (Dense)	(None, 9)	1161

Il modello viene addestrato per 30 epoche utilizzando l'ottimizzatore Adam ($lr = 0.01$) con riduzione del learning rate di un fattore $\phi = 0.1$ in caso di non decrescita della loss sul test set. I risultati ottenuti sono decisamente migliori come illustrato in tabella VIII.

Tabella VIII: Risultati MLP

F1-Mi	F1-Ma	HL	EMR
0.4300	0.3200	0.1493	0.2493

Particolarmente interessante il valore molto basso della Hamming-Loss, inoltre in questo caso un buon 25% delle label viene perfettamente classificato.

B. Classificazione singola-label

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

RIFERIMENTI BIBLIOGRAFICI

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel fsdasdaunctions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.