Classificazione di sottogeneri musicali tramite tecniche di text-mining

Lorenzo Mammana *807391*
l.mammana@campus.unimib.it
Gabriele Molteni *781251*
g.molteni11@campus.unimib.it
Matteo Scarpone *786222*
m.scarpone@campus.unimib.it

***Sommario*—Il metal è un genere musicale nato all'incirca agli inizi degli anni '70. Nel corso degli ultimi 40 anni la musica metal ha avuto talmente tante variazioni in stile e tematiche che nel tempo si è diversificato andando a formare dei veri e propri sottogeneri musicali. Queste correnti differiscono sia dal punto di vista musicale sia dal punto di vista dei temi affrontati nelle liriche. È possibile ipotizzare che alcuni temi siano propri di precisi sottogeneri musicali. In questo studio verrà mostrato come è possibile usare strumenti di text-mining e machine learning al fine di andare a classificare in modo automatico i vari sottogeneri musicali partendo dalle liriche delle canzoni.**

INDICE

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. Questo studio si concentra sulla possibilità di distinguere sottogeneri musicali dello stesso genere. In particolare ci si è concentrati sull'analisi di sottogeneri del genere musicale metal in quanto questo tipo di musica presenta una vasta varietà di sottogeneri, 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) [1], 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) [2] 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 del gruppo
- 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 in quanto le altre lingue sono troppo poco frequenti per poter essere usate e sarebbero un fattore di confusione per gli algoritmi di classificazione. La rimozione delle liriche mancanti o non in inglese riduce il numero di record 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 agevolare 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 nel dataset. Questo introduce 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 selezionati dagli autori sono i seguenti:

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

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%.

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	0.294338005
Black Metal	0.176705929
Power Metal	0.130578749
Doom Metal	0.12060161
Thrash Metal	0.082879827
Rock	0.068964018
Heavy Metal	0.05873217
Progressive Metal	0.054402117
Folk Metal	0.012797575

È interessante notare l'andamento dei generi nel tempo: il dataset fornisce una buona copertura dei generi per anno (Figura 1).

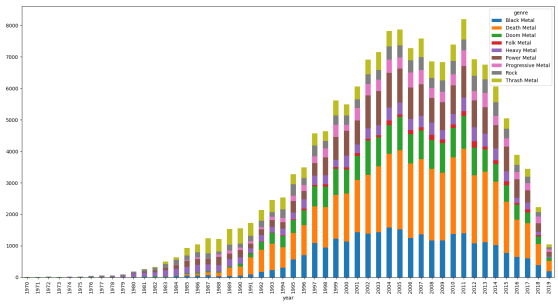


Figura 1: Distribuzione dei generi musicali negli anni

Si vede come alcuni generi comincino effettivamente a svilupparsi a partire da un determinato anno.

III. Preprocessamento del testo

Per poter eseguire una classificazione ottimale è necessario procedere con un'operazione di filtraggio del testo per ottenere dei *token* ben formati. Sono state quindi effettuate operazioni per rimuovere punteggiatura e parole troppo o troppo poco frequenti. Per farlo si è proceduto costruendo un workflow

Knime [3] utilizzato come wrapper per il codice Python 3 [4]. La scelta di utilizzare Python è dovuta al consumo estremamente elevato di risorse e tempo da parte di Knime sul dataset in questione.

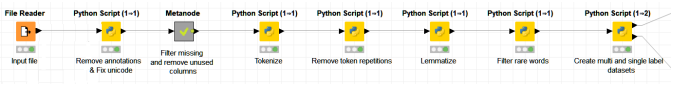


Figura 2: Workflow preprocessamento del testo in Knime

A. *Tokenizzazione*

La tokenizzazione è stata eseguita con il package *NLTK* (Natural language toolkit) [5], 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* [6]. 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.

```
['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 ulteriormente il numero di token estratti è stato eseguito il loro stemming, cercando quindi di ricondurre token simili alla loro radice comune. É stato utilizzato il *LancasterStemming* [7], 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 e quindi 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 un tempo di esecuzione molto minore del secondo. 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. Latent Dirichlet Allocation

Prima di procedere con la classificazione vera e propria si è provato ad utilizzare LDA, principalmente per verificare se esistesse una struttura all'interno dei dati evidenziabile senza utilizzare le label. Il modello viene addestrato utilizzando un valore di α asimmetrico appreso automaticamente sui dati che dovrebbe migliorare le performance su un dataset sbilanciato, settando 9 topic vengono prodotti i risultati mostrati in figura 3.

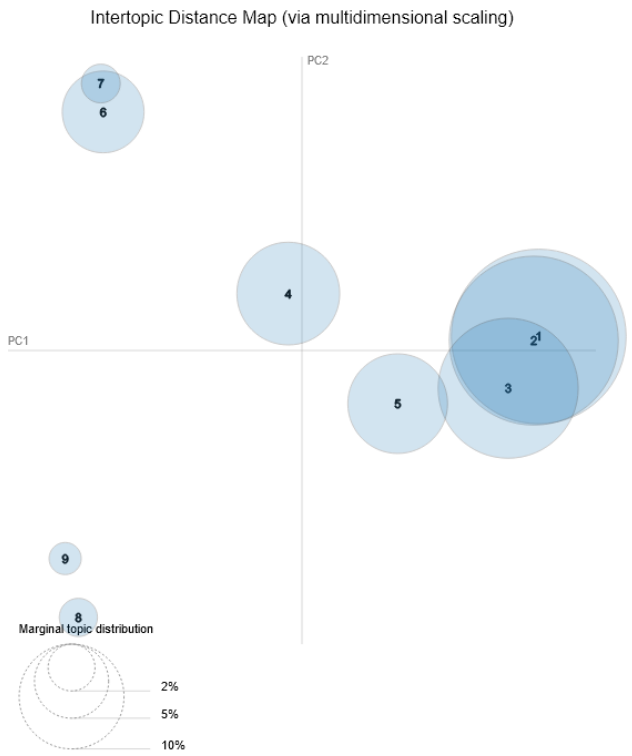


Figura 3: Distanza tra i topic ottenuti

Il grafico prodotto mostra dei topic abbastanza distribuiti e con poche sovrapposizioni, che sono comunque giustificate considerando la similarità intrinseca dei topic in oggetto. Visualizzando le parole più frequenti all'interno dei topic si è cercato di assegnare il sottogenere musicale a loro associato, i risultati sono mostrati in tabella VI.

Tabella VI: Top 10 parole frequenti per topic

id											Potential topic
1	ey	dark	night	fal	light	dream	soul	heart	sky	cold	Death
2	tim	nev	know	see	feel	way	lik	com	lov	lif	Doom
3	lif	die	dead	lie	kil	pain	fuck	mind	hat	liv	Thrash
4	blood	burn	hel	fir	dea	shal	black	com	sin	flam	Black
5	god	war	man	evil	land	pow	new	ear	nam	rid	Power
6	on	tak	fre	stand	ris	high	break	hand	watch	play	Heavy
7	fight	got	get	strength	right	along	emot	big	wear	chas	Progressive
8	hard	yeah	dant	sing	song	rock	anyth	dog	wom	bal	Rock
9	gold	welcom	doom	sev	tal	wint	strange	snow	hous	sink	Folk

Alcuni topic sono facilmente identificabili grazie alla conoscenza del dominio, parole come "god", "land" o "power" sono facilmente ricollegabili al Power metal, mentre per generi come il Death o il Doom i risultati prodotti sono piuttosto ambigui. Per ottenere un'indicazione più rigorosa riguardo la bontà dei risultati vengono calcolate le misure di Perplexity e Topic Coherence (variante sliding window) con valori rispettivamente pari a -6.5242 e 0.3978.

V. 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 nello stesso modo per garantire consistenza ai risultati prodotti. Come input vengono costruite tre matrici *Bag of Words* (BOW): una contenente dati in formato TF-IDF, una in cui ogni cella rappresenta le frequenze di ogni parola all'interno del documento (TF) e una matrice binaria in cui ogni cella rappresenta se la parola è contenuta o meno nel documento (Binary-TF).

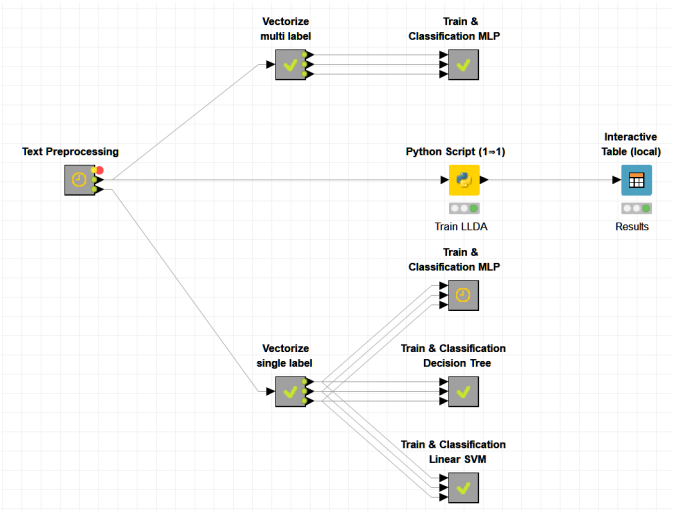
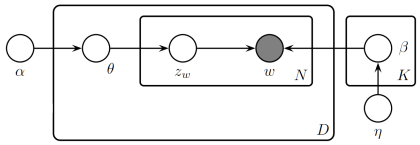


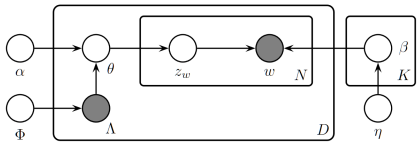
Figura 4: Workflow classificazione in Knime

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 5.



(a) LDA tradizionale



(b) Labeled LDA

Figura 5: 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 F1 Micro-average, F1 Macro-average e due metriche pensate per task di questo tipo ovvero la Hamming-loss [8] e la Exact Match Ratio. La prima è definibile come la frazione di label non correttamente classificate:

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

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 VII:

Tabella VII: Risultati LLDA

Macro-F1	Micro-F1	HL	EMR
0.3200	0.3300	0.3158	0.0318

Come ci si aspettava i risultati sono poco performanti in quanto il dataset è complesso ed estremamente sbilanciato. Il valore HL mostra che il modello è in grado di classificare le label con un buon grado di correttezza. Il valore EMR mostra invece che il modello non è in grado di classificare quasi nessuna label di un record perfettamente.

2) Percettrone multistrato: Un altro modello usato in letteratura per la classificazione multi-label è il percettrone multistrato (MLP) [9], l'architettura scelta è stata ottenuta in maniera sperimentale ed è così definita (Tabella VIII):

Tabella VIII: 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 IX.

Tabella IX: Risultati MLP

Input matrix	Macro-F1	Micro-F1	HL	EMR
TF-IDF	0.4812	0.3633	0.1899	0.1832
TF	0.4694	0.3307	0.1949	0.1760
Binary-TF	0.4855	0.3678	0.1898	0.1832

Particolarmente interessante il valore molto basso della Hamming-Loss. In questo caso un buon 18% delle label viene perfettamente classificato.

B. Classificazione singola-label

Sul dataset singola label vengono testati tre differenti modelli. Anche in questo caso vengono utilizzate le matrici TF-IDF, TF, e Binary-TF come input.

1) Percettrone multistrato: Il percettrone multistrato descritto in precedenza può essere adattato per operare su singola label modificando l'attivazione del layer di output e il tipo di loss utilizzata. Per l'addestramento viene usata una loss pesata sulle frequenze delle classi in modo tale da avvantaggiare le classi poco frequenti rispetto a quelle molto presenti nel dataset.

2) Macchina a vettori di supporto: La macchina a vettori di supporto (SVM) è uno dei modelli più noti per il compito di classificazione supervisionata. Anche in questo caso le classi vengono pesate rispetto alla loro frequenza. È stata usata una SVM con kernel lineare. È stato testato anche il kernel *rbf* ma i tempi richiesti per l'addestramento sono risultati troppo alti.

3) Albero di decisione: Infine è stato testato un semplice albero di decisione, sempre pesando le classi.

Come misure di accuratezza sono state usate il Micro ed il Macro F1-score, i risultati ottenuti sono mostrati in tabella X.

Tabella X: Risultati ottenuti su singola label

model	input matrix	Macro-F1	Micro-F1
MLP	TF-IDF	0.3009	**0.4227**
MLP	TF	0.2408	0.4080
MLP	Binary-TF	0.2844	0.4221
SVM	TF-IDF	0.3008	0.3743
SVM	TF	0.3023	0.3838
SVM	Binary-TF	**0.3100**	0.3849
Tree	TF-IDF	0.2216	0.2936
Tree	TF	0.2243	0.2955
Tree	Binary-TF	0.2252	0.2930

Il percettrone ottiene i risultati migliori quando l'attenzione viene posta sulla classe più frequente. Più interessante è forse il risultato ottenuto quando l'attenzione è posta sulle classi meno frequenti. In questo caso il modelli migliore è la SVM, anche se il vantaggio non è particolarmente significativo per giustificarne l'utilizzo rispetto al percettrone.

VI. Discussione

I risultati ottenuti sono stati al di sotto delle aspettative: ci si aspettava un riscontro maggiore sulla classificazione singola label. Come già detto in precedenza, all'interno del dataset, sia per la classificazione singola che multipla, sono stati introdotti due bias rilevanti. Questo, con la possibile combinazione di feature non perfettamente adatte, può essere stata la causa principale di fallimento degli esperimenti. Sembra che il modello migliore in grado di operare sul dataset sia proprio la versione originale di LDA. Questo ci ha fatto anche ipotizzare che l'implementazione di LLDA utilizzata potrebbe non essere ottimale.

VII. Conclusioni

Nonostante i risultati non siano stati eccezionali, il report mostra che esistono delle fondamenta nella classificazione di sottogeneri musicali a partire dalle liriche delle canzoni. I risultati prodotti da LDA mostrano l'esistenza di una struttura

sottostante i dati; se si fosse in grado di produrre un dataset maggiormente bilanciato e con maggiore consistenza delle classi, sarebbe probabilmente possibile ottenere una buona discriminazione.

RIFERIMENTI BIBLIOGRAFICI

[1] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," vol. 3, 01 2001, pp. 601–608.

[2] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, Aug. 2009, pp. 248–256. [Online]. Available: https://www.aclweb.org/anthology/D09-1026

[3] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel, "Knime: The konstanz information miner," in *Data Analysis, Machine Learning and Applications*, C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 319–326.

[4] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[5] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[6] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993. [Online]. Available: https://www.aclweb.org/anthology/J93-2004

[7] C. D. Paice, "Another stemmer," *SIGIR Forum*, vol. 24, no. 3, p. 56–61, Nov. 1990. [Online]. Available: https://doi.org/10.1145/101306.101310

[8] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, April 1950.

[9] F. F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," 1963.