Progetto gestione dell'informazione

Paoli Tommaso **152542**

Panoramica

Il progetto consiste nella creazione di un motore di ricerca e nella amministrazione dei dati che il motore dovrà utilizzare per soddisfare le ricerche degli utenti. Si è deciso gi raccogliere i dati da un sito di testi e accordi di canzoni italiane e inglesi

Indice

1. Preparazione dei dati
    - Recupero dei dati
    - Preprocessing dei dati
2. Preparazione degli indici
    - Inverted Index
    - Whoosh
3. Ricerche
    - Interfaccia utente
    - Composizione query
4. Sorgenti
    - main.py
    - cli.py

- Preprocessing
- Sentiment analysis

1. Preparazione dei dati

crawler: Scarichiamo il corpus di documenti dal sito ***accordiespartiti.it***

- tramite il modulo *request* scaricheremo delle pagine dal sito per ogni **artista** e guarderemo tute le sue **canzoni** scaricandone sia il **testo** sia gli **accordi**
- Ogni canzone verrà salvata come file di testo proprio come appare sul sito e avrà come nome: ***[nome artista].[nome canzone].canzone.txt***

cleaner: Per lavorare con tutti questi dati abbiamo preferito "ordinarli"

- Questo strumento guarderà tutte le canzoni scaricate in precedenza dal crawler
- Rimuove righe e spazi vuoti
- Rimuove le righe per le tablature (se presenti)
- Racchiude tutto in unico file ***testi.txt*** contenente per ogni riga una canzone:
    - nome artista

    - titolo canzone
    - ___ SEPARATORE ___
    - testo canzone
    - ___ SEPARATORE ___
    - set degli accordi

I separatori sono solo dei caratteri arbitrari scelti dal programmatore per semplificare il codice " { } "

indexer: In previsione di preparare l'**inverted index** e **whoosh** si è preferito attuare un'operazione molto "laboriosa" subito

- Questo strumento prendera il file ***testi.txt*** riscrivendolo
- Sostituisce però tutti i testi con gli stessi testi "preprocessati"
    - Rimarranno unicamente i token in forma base delle parole "significanti" presenti più volte con annessa la loro "frequenza"
- Questa operazione è risultata essere la più "costosa" per il sistema

2 Preparazione degli indici

invertedIndex: Tramite le funzionalità base del python creiamo il nostro **inverted index**

- Tramite il modulo pickle salveremo il nostro "insieme di liste" all'interno di un file

whoosh: Lo schema sarà cosi:

titolo	artista	tokens	accordi	Punteggio sentiment
Titolo della canzone	Nomi degli artisti della canzone	Lista dei token principali della canzone	Set degli accordi	Punteggio di sentiment associato al testo e al titolo della canzone

3. Ricerche

Per testare il progetto si è deciso di preparare due strumenti che pemettono di avviare delle ricerche scelte dall'utente.

main.py Avviandolo vengono eseguite tutte le fasi del progetto per poi evvettuare alcune ricerche. Per ogni fase vengono mostrati anche i tempi di esecuzione.

Mostriamo ora un esecuzione di prova gia eseguita con le relative tempistiche.

	Crawler	Cleaner	Indexer	InvertedIndex	Whoosh
Secondi	7888.57	80.74	545.86	0.67	14.46

Preprocessing

Per questo progetto si è deciso di applicare un preprocessing un filo più "aggressivo" di come è stato studiato, e quindi verranno più spesse ignorate parole anche apparentemente significanti. Inoltre parlando di testi di canzoni si è deciso di non dare la minima importanza all'ordine delle parole ma solo al loro significato e la loro frequenza in ogni testo. Per quanto riguarda gli accordi si è presa in considerazione la misurazione della tonica di ogni canzone. (Si è deciso di ignorare i cambi di tonalità). Per ogni canzone verrà memorizzato un set con tutti gli accordi apparsi una o più volte. Dopo aver recuperato i token della canzone, (puliti stemmati e senza stopwords), vengono contate le frequenze all'interno del testo. Infine vengono rimossi tutti i token che non sono presenti almeno 1/3 delle volte che è presente il primo token.

Main.py

```
    # cancella tutti i documenti e il corpus di documenti ( se presenti
)
    Boss.reset(*paths, CANZONI_DIR)

    b = Boss()

    # avvia il crawler e scarica tutte le canzoni
    b.crawler(LINK, CANZONI_DIR)

    # avvia il cleaner che da un primo passaggio ai file delle canzoni
    b.cleaner(CANZONI_DIR, TESTI_FILE)

    # Avvia l'indexer che prepara il corpus all'inverted index e
l'indice di whoosh
    b.indexer(TESTI_FILE, INDICE_FILE)

    # Crea l'inverted index
    b.invertedIndex(INVERTEDINDEX_FILE, INDICE_FILE)

    # Crea l'indice whoosh
    b.whoosh(INDICE_FILE, WHOOSH_DIR)

    # Seleziona un termine generico da cercare
    cerca = ['amore', 'soldi', 'potere', 'sacrificio', 'forza']
[random.randint(0, 4)]

    # cerca il termine tramite entrambi gli indici
    print("Vado a cercare la parola:", cerca)
```

```
    searchWhoosh(cerca)
    searchInverted(cerca)
```