

Analizzatore di articoli

Descrizione

Questo programma è in grado di elaborare degli **articoli** di giornale per calcolare il **peso** delle parole che in essi appaiono e farne una classifica. Il peso è inteso come numero di singoli articoli in cui le parole appaiono; pertanto molteplici apparizioni di una stessa parola nello stesso articolo non ne aumentano il peso. Il linguaggio di programmazione utilizzato è Java (versione 8).

Funzionalità

- Sono disponibili due modalità di **input**: argomento e CLI.
- Sono supportate nativamente due **sorgenti** di articoli: file in formato csv e l'utilizzo delle **API del The Guardian** per l'acquisizione di articoli dall'archivio online del quotidiano.
- Gli articoli elaborati con le sorgenti vengono archiviati in un file in formato JSON e possono essere direttamente rielaborati dal programma.
- Il programma è in grado di filtrare i risultati tramite una **blacklist** contenuta nel file words.txt di default o fornito dall'utente. Nel caso essa non sia presente non verrà filtrato nessun termine.
- Per espandere il codice ed aggiungere nuove sorgenti esse devono implementare l'interfaccia ArticleSource .
- Per utilizzare altre strutture per la memorizzazione dei termini esse devono implementare l'interfaccia TokensStorage .

Requisiti per il funzionamento

- Per l'esecuzione del programma deve essere presente il jre8 .
- Per la compilazione del programma deve essere presente il jdk8 e maven .

Utilizzo

1. Estrarre l'archivio article-analyzer-1.0.zip
2. Entrare nella cartella article_analyzer_1.0
3. Aprire un terminale ed eseguire java -jar article_analyzer-1.0.jar , con eventuali argomenti

Parametri passati per argomento

È disponibile l'argomento -h che fornisce una guida rapida all'uso degli argomenti. Gli argomenti funzionali sono:

- -file,--read-file Per leggere gli articoli da un'altra ricerca.
- -json,--json-file-name < arg > Il nome del file della ricerca da cui leggere.
- -api,--read-api Per leggere gli articoli utilizzando le API del TheGuardian.
- -apikey,--api-key < arg > Il valore della API key per autenticarsi con le API del TheGuardian.

NOTA: se non specificato, viene usata quella di default

- -default,--set-default Per far sì che la chiave specificata venga impostata come default.
- -csv,--read-csv Per leggere gli articoli da file CSV.
- -name,--csv-file-name < arg > Il nome del file CSV.
- -max,--max-articles < arg > Il numero massimo di articoli da elaborare.

Argomenti opzionali

- -queries,--queries < args > I termini con cui fare la ricerca.
- -tags,--tags < args > I tag con cui condizionare la ricerca.
- -show,--show Per stampare la classifica dei termini.
- -store,--store < arg > Il nome del file in cui la ricerca verrà salvata.

NOTA: se non specificato, il nome di default è “result”.

- -result,--result < arg> Il nome del file dove viene salvato l'esito dell'elaborazione.

NOTA: se non specificato, il nome di default è “result”.

- -tokens,--tokens < arg > Il numero di termini di cui fare la classifica.

NOTA: se non specificato, il numero di default è 50.

CLI

Se il programma viene eseguito senza argomenti si avvierà l'interfaccia da riga di comando che con una serie di domande di configurazione si preparerà all'elaborazione dei dati.

Struttura di file e cartelle

Nella release, oltre al file eseguibile, è presente una cartella `resources` che presenta la seguente struttura:

```
/resources
  ... /log
    ... test.json
  ... /blacklist
    ... words.txt
  ... /CSV_sources
    ... file.csv
  ... /private
    ... private.properties
  ... /results
    ... result.txt
```

- `log` contiene i file dove sono state salvate le ricerche precedenti.
- `blacklist` contiene il file `words.txt` che serve per filtrare i termini
- `CSV_sources` contiene i file CSV da utilizzare come sorgenti
- `private` contiene il file `private.properties` dove è contenuta la API key di default

NOTA: il nome della proprietà della API key è `apiKey`

- `results` contiene i file su cui vengono scritti i risultati dell'elaborazione

Compilazione

Usare il comando `mvn package` all'interno della cartella `article_analyzer` (dove si trova il file `pom.xml`). Il file compilato (`.jar`) sarà disponibile nella cartella `target` con il nome `article_analyzer-1.0-jar-with-dependencies.jar`

Librerie esterne

- maven-assembly-plugin v2.5.3 per compilare il file eseguibile .jar .
- org.apache.maven.plugins maven-surefire-plugin v2.22.0 plugin utilizzato da maven per fare i test.
- org.apache.maven.plugins maven-compiler-plugin v3.3 per la compilazione del progetto (Java 8).
- org.apache.maven.plugins maven-site-plugin v3.7.1 per generare della documentazione inerente al progetto.
- org.apache.maven.plugins maven-project-info-reports-plugin v3.7.1 genera i report dei test.
- com.fasterxml.jackson.core jackson-databind v2.15.1 per la serializzazione e deserializzazione degli articoli in e da il file di log (formato JSON).
- commons-cli v1.5.0 utility per la creazione e configurazione degli argomenti.
- me.tongfei progressbar v0.9.4 utility per creare una barra di progresso delle operazioni.
- org.apache.httpcomponents httpclient v4.5.14 client HTTP per stabilire una connessione con il webserver.
- org.json v20090211 per il parsing di documenti in formato JSON .
- org.jsoup v1.15.4 per la conversione di articoli dal formato HTML .
- edu.stanford.nlp v4.4.0 per la tokenizzazione degli articoli.
- org.junit.jupiter v5.9.2 per il testing con Junit-5.
- org.mockito Mockito-core v2.7.5 per simulare delle classi in fase di test.
- org.mockito Mockito-junit-jupiter v2.17.0 per simulare delle classi in fase di test.
- nl.jqno.equalsverifier v3.0.3 per verificare la correttezza dell'implementazione dei metodi equals() e hashCode().
- com.opencsv v5.2 per il parsing di documenti in formato CSV .
- com.fasterxml.jackson.core jackson-xml-databind v0.6.2 per serializzazione e deserializzazione.