Università degli studi di Verona

Progetto di Big Data

Wikipedia inverted index

Fabrizio Bassan VR479096

Aprile 2023

Contents

1 Introduzione

Premessa: qui ho scritto una sintesi del progetto, dal momento che é stato scritto utilizzando jupyter notebook ho voluto lasciare i dettagli piú tecnici all'interno del file, scrivendo in linguaggio MarkDown sopra la cella, il comportamento del pezzo di codice scritto sotto.
Richiesta del progetto: "L'output deve contenere, per ogni parola, la lista di pagine di wikipedia che contengono quella parola". Divideremo, quindi, in 3 parti il progetto: una fase di processing, una parte per la gestione dell'inverted index e la fase finale, la creazione dei grafici. Utilizzeremo principalmente 4 librerie: pyspark, lxml, pandas, matplotlib. Ce ne sono altre ma sono tendenzialmente di contorno a queste che sono praticamente essenziali. Ovviamente ci sará una prima fase per controllare se il file (con estensione xml) esiste, in quel caso si andrá avanti, altrimenti verrá generato un errore che bloccherá l'esecuzione.

2 Processing

La fase di processing si suddivide in piú parti.

2.1 Metodi

Qui abbiamo utilizzato due metodi (removeNestedParentheses e cleanUp) per pulire il piú possibile il file che verrá generato a partire dal file.xml in quanto questo é scritto in formato *WikiText*, un linguaggio di MarkUp particolare usato da wikipedia. Arriviamo ora alle stop words. Cosa sono le stop words? Chiedendo aiuto ad internet, la sua definizione risulta questa: *sono quelle parole che, per la loro alta frequenza in una lingua, sono di solito ritenute poco significative dai motori di ricerca, che le ignorano.*. Nel nostro caso, prendiamo le stop words inglesi, come ad esempio verbi ausiliari (is, are, am, ecc.), congiunzioni (but, and, ecc.) e cosí via.

2.2 Analisi del file

Finalmente arriviamo alla parte piú importante: il processamento del file.
Creo due file CSV (cosí mi é piú semplice analizzarli piú avanti), con due header ciascuno: index, che é la colonna comune fra i due, e altri, complessivamente, due campi ossia **text** e **title**. Molto importante sará l'utilizzo del comando *elem.clear()* che ci permetterá di non sovraccaricare la memoria primaria scrivendo quindi ogni volta che un tag text oppure title viene

trovato, una riga in uno dei due file csv creati all'inizio della cella.
Ho anche impostato un contatore formattato appositamente che stamperá la percentuale di completamento dell'operazione.

3 Inverted Index

Qui si andrá ad utilizzare principalmente la libreria PySpark.
Dopo aver creato una SparkSession, andró a unire i due file csv precedentemente creati utilizzando la colonna in comune: *index*. Ne uscirá quindi un file unico *rawMerged.csv* contenente, per ogni titolo, tutte le parole che aveva all'interno della sua pagina.
Successivamente si andrá a creare un altro file *countedWords.csv* che conterrá 3 colonne: il titolo della pagina, la parola singola utilizzata ed un counter per tener conto di quante volte quella parola é stata utilizzata in quella pagina.
Da qui, ho generato un altro file *singleWordCounted.csv* che avrá due sole colonne che indicheranno la parola ed il numero di occorrenze.
Riprendendo countedWords.csv vado ad eliminare la colonna *count* cosí da creare un file *finalFileToAnalyze.csv* per lettere tutte le parole presenti e salvarle in un set (per eliminare tutti i doppioni) e fare un casting di questo set a lista in modo da renderlo piú flessibile e comodo.
Ci servirá per creare (finalmente) il nostro file finale: *results.csv* che avrá due colonne: word e titles (che é la lista dei titoli), filtrando le parole in modo da evitare doppioni, usando la lista precedentemente creata e scrivendo, sempre in un file csv, riga per riga.
Poi creeró altri tre files, *initialLetter.csv* (due colonne, char e occurence), *titlesSize.csv* (due colonne, word e list_size) e *occurences.csv* (list_size e occurences che contiene la frequenza del numero di parole) che ci serviranno per la generazione dei grafici.
Infine
Ho utilizzato parecchio, ovviamente, la libreria PySpark in quanto ci é tornata molto utile per le operazioni complesse sui files csv (vedi ad esempio i groupBy) mentre per le operazioni piú semplici ho preferito usare pandas.

4 Grafici

Le librerie che ho utilizzato di piú qui sono principalmente pandas per l'analisi dei csv e matplotlib per la generazione dei grafici. Ne ho creati 5:

- Parole piú utilizzate: il primo grafico che mi é venuto in mente, anche quello piú effettivamente semplice. É un semplice grafico a barre generato con l'ausilio di *singleWordCounted.csv*. Rappresentata nell'asse x la parola e nell'asse y le occorrenze di quella parola. Per rendere il tutto piú "pulito ed ordinato" ho preso soltanto le prime 10, ossia quelle con i valori maggiori.

- Parole presenti in piú titoli: su queste ci ho pensato un po' di piú, come dice il titolo viene mostrato un grafico che indica le stesse cose sopra ma, in soldoni, le parole singole piú utilizzate, senza doppioni. Infatti i valori sono di molto inferiori. Qui ho utilizzato il file *titlesSize.csv*.

- Ho anche sfruttato il file *occurrences.csv* per generare i due grafici successivi: entrambi raggruppano i titoli che contengono lo stesso numero di parole solo che nel primo verrá generato in base a circa 50 valori, nel secondo, che risulta piú dettagliato, andremo a crearlo basandoci sui 5 valori maggiori. Mi spiego meglio: i titoli che contengono una parola sono quasi di 1 milione e mezzo.

- Nell'ultimo grafico ho unito due tipologie di grafici: lo scatter ed il plot (con il solo scatter veniva troppo confusionario), invece cosí (anche se il risultato sembra uno di quei giochini in stile "unisci i puntini) é piú ordinato e capibile. Ho usato due colori diversi in modo da risaltare l'unione fra i due. Questa volta il file che ci ha aiutato nella creazione di quest'ultimo grafico é stato *initialLetter.csv*.

5 Finale

Alla fine fermo l'instanza di spark e chiedo se i files creati precedentemente creati (tutti con estensione csv) volessero esser eliminati. Avrei voluto andare a chiedere singolarmente, file per file, se volesse esser cancellato oppure no ma mi é sembrata una scelta fin troppo lunga e dispersiva. Ho usato la libreria *os* in modo da eseguire automaticamente l'azione.