Elaborato finale Parallel Computing - Lambda Architecture per sentiment analysis

Alessandro Arezzo
E-mail address
Alessandro.arezzo@stud.unifi.it

Abstract

Il progetto presentato è stato realizzato come elaborato finale dell'insegnamento di Parallel Computing previsto dal corso di laurea magistrale in ingegneria informatica dell' Università degli studi di Firenze. Il suddetto elaborato consiste nell'implementazione di una Lambda Architecture in grado di eseguire sentiment analysis di tweets scaricati da Twitter. Il codice consente poi di visualizzare i risultati ottenuti con lo scopo di permettere la comprensione dei benefici apportati dall'architettura realizzata nel suddetto contesto considerato.

1. Introduzione

L'elaborato oggetto di questa relazione consiste nell'implementazione di una Lambda Architecture atta ad analizzare tweets scaricati dal popolare social network Twitter.

Una Lambda Architecture, nel dettaglio, rappresenta un'architettura disegnata appositamente per l'elaborazione di una grande quantità di dati immutabili ed in continuo aumento. L'implementazione di un tale sistema software risulta difatti ideale quando si ha necessità di processare non solo un insieme di dati memorizzati in un dataset da elaborare, ma bensì quando a tale computazione occorre aggiungerne una real-time su dei nuovi dati in arrivo.
L'architettura in questione trae difatti beneficio dalla suddivisione del processamento su due differenti livelli. Uno di questi, denominato **batch layer**, ad ogni ciclo di computazione si occupa di processare i dati contenuti nel dataset principale (dataset master), mentre all'altro, al quale è associato il nome di **speed layer**, viene assegnato il processamento real-time dei nuovi dati in arrivo. Questi ultimi sono poi aggiunti al dataset master così che possano essere processati dal batch layer durante il prossimo ciclo di batch.
Vi è poi un ulteriore livello chiamato **serving layer**, il quale è incaricato di memorizzare le viste fornite in output dai due processamenti.

Come si evince da questa breve descrizione, risulta evidente come l'implementazione di una simile architettura si adatti perfettamente all'elaborazione di tweets pubblicati su Twitter, i quali appunto non solo rappresentano un enorme quantità di dati da processare, ma bensì risultano anche essere in costante aumento in quanto si stima che, nel corso del mese di Ottobre 2019, ne siano stati pubblicati in media circa 6 000 al secondo [3].
Ai fini del progetto proposto, si è considerata, una particolare analisi dei tweets denominata **sentiment analysis**, la quale prevede che per ogni tweet sia calcolato un valore intero (sentiment value) che ne indica la negatività (se minore di 0), la neutralità (se pari a 0) o la positività (se maggiore di 0). Il valore in questione viene calcolato come la somma dei sentiment value associati alle parole che lo compongono, la quale informazione è a sua volta contenuta in un dizionario denominato AFINN-165 che associa ad alcune parole della lingua cui il file si riferisce il rispettivo sentiment value in un range compreso tra -5 (total-

mente negativo) e +5 (totalmente positivo). Per lo sviluppo di questo elaborato, nello specifico, si è considerato un particolare problema di sentiment analysis che prevede il calcolo e la conseguente comparazione del numero di tweets negativi, neutrali e positivi, contenenti al loro interno una certa parola assegnata, alla quale viene associato il nome di **query**.

Il progetto prevede infine una fase di esperimenti atta a verificare la bontà dei benefici apportati dalla Lambda Architecure realizzata al contesto della sentiment analysis, per mezzo dell'esecuzione di un modulo di codice che preleva i risultati prodotti tramite apposite interrogazioni formulate al serving layer e li rappresenta per via grafica.

2. Implementazione

La realizzazione di una Lambda Architecture, come descritto nel corso del paragrafo introduttivo, prevede l'implementazione delle tre componenti relative a batch layer, speed layer e serving layer.
Nel merito del progetto qui presentato, si sono utilizzati i sistemi software **Apache Hadoop** ed **Apache Storm** rispettivamente per parte batch e real-time ed il database distribuito **Apache HBase** per le funzionalità cui è adibito il serving layer.

Il funzionamento dell'architettura, il cui schema è descritto in Figura 1, prevede innanzitutto che il batch layer vada a scorrere i tweets contenuti in un dataset master, li filtri ed esegua sentiment analysis solamente di quelli in cui è presente la query prestabilita. Una volta calcolato quindi il numero di quelli appartenenti alle tre diverse classificazioni (positivi, neutrali e negativi) li inserisce nell'apposita tabella del serving layer. Lo speed layer, invece, esegue la stessa operazione real-time calcolando il sentiment value associato a dei tweets ricevuti in streaming ed a sua volta aggiorna il database contenuto in HBase così da fornire la propria speed view. Inoltre, lo speed layer aggiunge i nuovi tweets computati al dataset master così che questi possano essere analizzati da Hadoop durante i prossimi cicli di batch.
Per quanto riguarda invece il serving layer, per ogni query vi è una tabella che associa a ciascuna delle tre classificazioni il numero di tweets rilevato per esse.

Oltre a tali componenti, risulta poi essere di estrema rilevanza il dataset master contenente i tweets che vengono forniti in input al batch layer. Tale dataset difatti riporta, per ogni tweet contenutovi non solo il suo testo, necessario al sistema per eseguire sentiment analysis, ma anche altre informazioni quali la lingua ed il **timestamp**. Quest'ultimo svolge un ruolo cruciale, in quanto consente al batch layer di escludere dal processamento quei tweets che sono già stati computati dallo speed layer durante il corrente ciclo di batch.

È poi presente un modulo software che si occupa di formulare delle query al serving layer così da poterne rappresentare per via grafica i risultati rilevati.

2.1. Batch layer

Per quanto concerne l'implementazione della parte batch layer dell'applicazione, come già accennato, si è utilizzato il framework Hadoop.
Lo scopo di tale componente è quello di analizzare i tweets contenuti in un dataset, denominato dataset master ed il cui percorso è indicato in fase di configurazione, per poi salvare i risultati ottenuti nell'apposita tabella memorizzata in HBase.
Nello specifico, si sono definiti i tre moduli software che rappresentano rispettivamente driver, mapper e reducer del sistema hadoop. Tale loro implementazione può essere riassunta come segue:

- **Driver:** setta tutte le componenti necessarie per l'esecuzione di mapper e reducer ed inizializza l'esecuzione del prossimo ciclo di batch andando a manipolare i dati contenuti in HBase. Nello specifico, per l'esecuzione di tale operazione, legge la query (parola sulla base della quale si vogliono filtrare i tweets) e crea un'apposita tabella nel database dis-

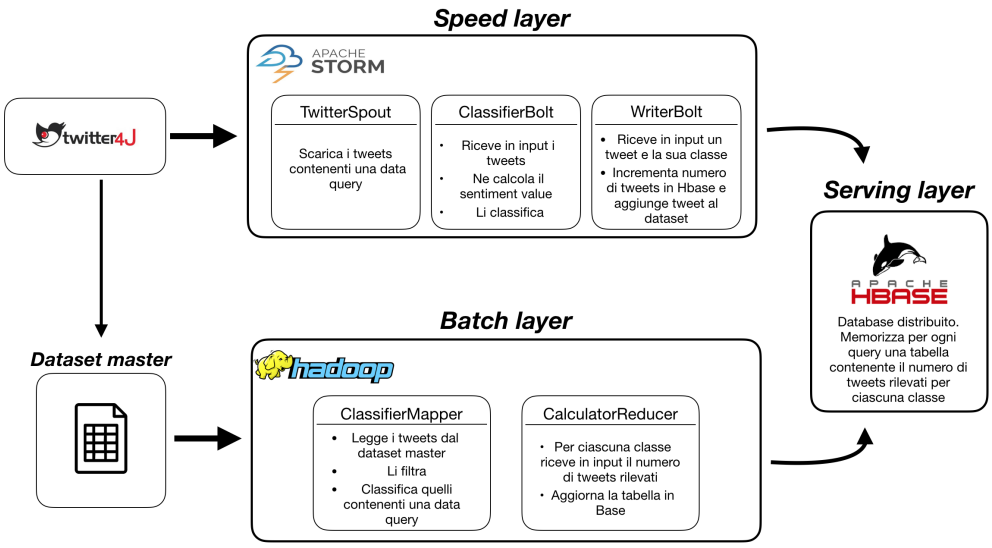

Figure 1. Schema della Lambda Architecture

tribuito in cui poter memorizzare il numero di tweets positivi, neutrali e negativi contenenti il termine in questione, se questa non è già presente. Inizializza quindi tale tabella andando a sovrascrivere il numero di tweets rilevati per ciascuna classe col valore 0.
Infine salva un timestamp che rappresenta il tempo corrente, il quale viene poi utilizzato dal mapper per escludere i tweets già computati dallo speed layer durante il corrente ciclo di batch.

- **ClassifierMapper:** ad ogni iterazione riceve in input una linea del dataset contenente i dati di un tweet e ne estrae i relativi valori di testo, lingua e timestamp. Filtra quindi i tweets considerando solamente quelli che contengono la query e che hanno un timestamp inferiore a quello salvato dal driver. Di questi, ne calcola il sentiment value ed invia al reducer la coppia chiave valore in cui la chiave indica la classificazione del tweet (positiva, negativa o neutrale) ed il valore è un semplice intero pari ad 1.

- **CalculatorReducer:** al termine dell'esecuzione del mapper, il riduttore riceve una coppia di valori per ognuna delle tre classificazioni considerate. Il primo termine è una semplice stringa che indica appunto la classe, mentre il secondo è una lista di interi pari ad 1 ed il cui numero di elementi indica quanti tweets del dataset contenenti la query sono stati catalogati con la classificazione in questione. Il reducer aggiorna quindi la tabella relativa alla query in HBase, andando ad aggiungere, per ciascuna classe, al precedente valore memorizzato il numero di tweets rilevati dal mapper. Si noti come tale valore che viene letto per poi essere incrementato rappresenta, per ogni classificazione, il numero di tweets rilevati per la categoria in questione dallo speed layer nel tempo intercorso tra l'inizio e la conclusione del corrente ciclo di batch.

2.2. Speed layer

Ai fini dell'implementazione dello speed layer, si è utilizzato il framework Apache Storm.
Il compito di tale componente è quello di scaricare da Twitter dei tweets che contengono la query e di eseguirne l'analisi real time. Tale analisi, consiste nel leggere ciascun tweet ricevuto per poi calcolarne il sentiment value, così da poterlo infine classificare come positivo, neutrale o negativo ed incrementare il valore associato alla classe

rilevata nella tabella in HBase .
Lo speed layer aggiunge infine al dataset master i nuovi tweets processati, così che questi possano essere in futuro analizzati dal modulo Hadoop durante i prossimi cicli di batch.
Per adempiere a tali funzioni, si è definita una topologia con i seguenti componenti:

- **TwitterSpout:** scarica i tweets da Twitter e li invia al primo bolt della topologia.
  Per il downloads dei dati, utilizza la libreria *Twitter4j*, la quale permette di aprire uno stream da cui scaricare i tweets in real time. Condizione necessaria affinchè sia possibile aprire lo stream in questione è che si inseriscano delle chiavi di accesso reperibili creando un'applicazione alla pagina developer di Twitter. Nello specifico, l'applicazione può essere creata al link https://developer.twitter.com e le chiavi ricevute vanno inserite nel file di configurazione del progetto.
  La libreria ha inoltre consentito di filtrare i tweets ricevuti dallo stream permettendo il download solo di quelli contenenti la query.

- **ClassifierBolt:** riceve i tweets dallo spout e ne calcola il sentiment value. Vi attribuisce quindi un valore testuale atto ad indicarne la classificazione ed invia all' ultimo bolt della topologia la coppia composta dal tweet analizzato e dalla sua classe.

- **WriterBolt:** riceve la coppia di valori composta dal tweet e dalla sua classificazione e si occupa quindi sia di aggiornare la tabella della query in HBase in modo da mantenere la consistenza dei dati relativa alla propria speed view, sia di scrivere il nuovo tweet nel dataset master.
  Per l'aggiornamento del serving layer, legge il corrente numero di tweet rilevati per la classe in questione e lo incrementa di 1.
  Scrive quindi poi nel dataset master le informazioni del tweet appena computato. Da notare, che tra tali informazioni vi è sia la lingua del tweet, info utile al batch layer per poter calcolare correttamente il sentiment value, sia il timestamp, utilizzato dal mapper di hadoop per escludere durante il processamento quei tweets che sono già stati computati dallo speed layer durante il corrente ciclo di batch.

```
hbase(main):003:0> scan 'Apple'
ROW                   COLUMN+CELL
 Negative             column=sentiment:value, timestamp=1579538220210, value=415
 Neutral              column=sentiment:value, timestamp=1579538238366, value=888
 Positive             column=sentiment:value, timestamp=1579538238369, value=577
3 row(s) in 0.1320 seconds
```

Figure 2. Esempio di tabella della base dati. Memorizza il numero di tweets contenenti il termine 'Apple' che sono stati classificati come negativi (415), neutrali (888) e positivi (577).

2.3. Serving layer

Per quanto riguarda il serving layer, si è optato per l'utilizzo di Apache HBase, una base dati distribuita atta a memorizzare sia le speed view real time fornite in output da Storm, sia le batch view generate dal processamento batch di Hadoop.
La base dati, memorizza per ogni query una tabella avente per nome proprio il termine cui questa si riferisce. Ogni riga della tabella, come si può notare dall'esempio in Figura 2, associa a ciascuna delle tre classificazioni considerate il numero di tweets contenenti la query rilevato.

2.4. Filtraggio e sentiment analysis

Una parte dell'implementazione di estrema rilevanza è poi attribuita al codice incaricato di filtrare i tweets e di calcolarne il sentiment value. Per la prima delle due operazioni, vengono innanzitutto considerati solamente i tweets contenenti la query, sia questa espressa così come è stata definita in fase di configurazione oppure interamente in lettere maiuscole e minuscole, o in alternativa dalla stessa preceduta dai simboli '#' e '@' per la ricerca di hashtag e tag. Inoltre, avviene una seconda fase di filtraggio che controlla la lingua di ciascun tweet ed esclude quelli per i quali non si dispone di un relativo dizionario AFINN-165.
Questo è difatti un elemento fondamentale ai fini del rilevamento del sentiment value di un tweet, il quale viene calcolato come la somma dei sen-

timent value associati alle parole che lo compongono, informazione questa contenuta proprio nei suddetti files. Un file AFINN-165 è difatti un dizionario che associa ad alcune parole della lingua cui si riferisce un valore di sentimento nel range tra -5 e +5. È quindi evidente come non si possa eseguire sentiment analysis di un tweet scritto in un linguaggio di cui non si ha in memoria un dizionario AFINN-165.
Si noti infine come tale informazione relativa alla lingua del tweet venga da un lato reperita dallo speed layer direttamente per mezzo dell'analisi dell'oggetto restituito dallo stream della libreria Twitter4j, mentre dall'altro sia letta dal batch layer come dato contenuto nel dataset master. Ciò è possibile grazie al fatto che lo speed layer quando aggiunge nuovi tweets al dataset vi inserisce anche l'informazione relativa alla lingua.

3. Dataset utilizzato

Per quanto riguarda i dati richiesti dal progetto, questi possono essere suddivisi in due categorie.
Da un lato, vi è il **dataset master**, il quale contiene i tweets necessari per il processamento batch. Come descritto nel corso del precedente capitolo, tale dataset viene via via popolato dallo speed layer, il quale dopo aver analizzato un tweet aggiunge il suo testo proprio al file in questione, memorizzado anche altre sue informazioni necessarie al batch layer quali la lingua ed il timestamp.
Per lo svolgimento dell'elaborato qui presentato, al fine di performare l'esecuzione della Lambda Architecture, si è partiti da un dataset di base reperibile al seguente link https://www.kaggle.com/kazanova/sentiment140 [4]. Questo, è un file in formato .csv, contenente 1 600 000 tweets per ciascuno dei quali memorizza non solo il testo ma anche altre informazioni come id, user, data ed altro. Tra tali dati non è però presente la lingua del tweet, problema questo comunque aggirabile dal momento che tutti dati sono scritti in lingua inglese. È stato quindi sufficiente considerare nel batch layer l'inglese come lingua di quei tweets per i quali tale informazione non è presente nel dataset. Allo stesso modo, non vi è neanche il timestamp per i tweets contenuti nel dataset di base scaricato, per tale motivo il batch layer computa di default ogni tweet che non possiede questa informazione all'interno del dataset.
Altri dati richiesti dal problema sono i **files AFINN-165**. Per lo sviluppo del progetto si sono considerati i dizionari disponibili al link https://github.com/dkocich/afinn-165-multilingual [1], dove sono raggruppati 64 file AFINN-165, ciascuno relativo ad un diverso linguaggio. Il progetto include poi un file .txt che associa ad ogni paese espresso con la sua notazione in lingua inglese in cui si parla uno tra tali linguaggi, il relativo nome in lingua nativa. Questa è un'informazione utile quando lo speed layer scarica un tweet di cui non è espressamente indicata la lingua: in tal caso si preleva di default la lingua parlata nel paese del tweet.

4. Esperimenti

Affinchè fosse possibile comprendere i benefici apportati dalla Lambda Architecture al caso della sentiment analysis, si sono eseguiti alcuni tests in locale. Tali esperiementi hanno riguardato nello specifico l'analisi del sentiment value di tweets contenenti la **parola 'Trump'**. La scelta di tale termine si deve al fatto che questo, rappresentando il nome del Presidente degli Stati Uniti d'America, indica una figura pubblica, ruolo che si adatta perfettamente ad eseguirvi un processo di sentiment analysis applicato per comprendere le opinioni delle persone nei suoi confronti.
La fase di testing ha fondamentalmente previsto tre fasi distinte.
Innanzitutto si è eseguito un processamento batch sul dataset master di base scaricato (vedi capitolo 3). Se ne sono quindi salvati i risultati per poi eseguirne uno real-time per mezzo di un esecuzione dello speed layer durata 60 minuti, in modo così da poterne confrontare i risultati prodotti con quelli ricavati dalla precedente analisi batch.
Poichè fosse possibile apprezzare i vantaggi in-

Figure 3. Risultati ottenuti dopo il primo processamento batch con Hadoop

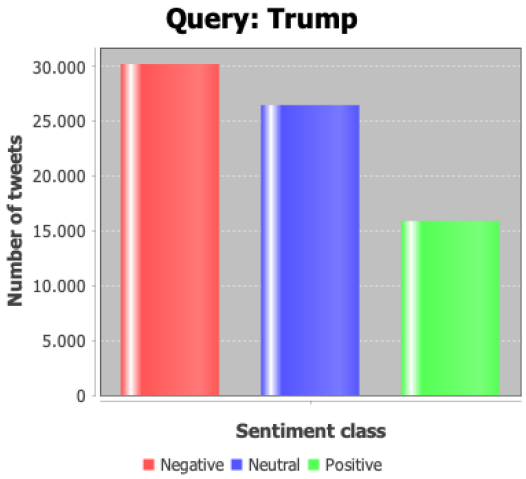

Figure 4. Risultati ottenuti dopo 60 minuti di processamento real-time con Storm

dotti dall'architettura sviluppata, si è perciò eseguito un nuovo ciclo di batch sul nuovo dataset master contenente anche i nuovi dati aggiunti dallo speed layer, al quale si è affiancato in parallelo la computazione real-time di Storm di modo da comprendere appieno l'efficacia derivata dall'avere a disposizione moduli distinti per i due differenti tipi di processamento.

I risultati ricavati, descritti di seguito, sono rappresentati da un apposito modulo di codice che, una volta eseguito, interroga ad intervalli regolari HBase prelevando i dati della tabella relativa ad una certa query e ne mostra per mezzo di un istogramma gli attuali dati contenuti.

4.1. Risultati

Per prima cosa, come detto, si è eseguito un processamento del batch layer sui dati contenuti nel dataset master di partenza scaricato [4]. I risultati prodotti, il cui grafico è mostrato in Figura 3, mostrano come una sentiment analysis eseguita da un semplice processamento batch dipenda fortemente dal dataset utilizzato. In questo caso difatti su 1 600 000 tweets contenuti nel dataset, solamente 34 contenevano la parola Trump, il che non consente di ricavare in output dei dati particolarmente rilevanti.

Di tutt'altro spessore risultano invece i risultati raccolti da Storm dopo circa 60 minuti di esecuzione dello speed layer, i quali sono mostrati in Figura 4. Questi ultimi difatti classificano circa 72 523 tweets contro i soli 34 del primo processamento batch, con la conseguente variazione dei risultati prodotti che vedono in tal caso un maggior numero di tweets negativi rilevati rispetto a quelli positivi, al contrario di quanto emerso dalla precedente analisi basata su pochi dati.

A fronte di ciò e dell'importanza del dataset utilizzato, appare lampante quanto possa essere vantaggioso avere a disposizione una Lambda Architecture ai fini di tale tipologia di processamento. Alla prossima esecuzione batch difatti i risultati prodotti saranno esattamente gli stessi mostrati in Figura 4, i quali possono essere ulteriormente migliorati per mezzo di una nuova esecuzione (parallela) dello speed layer. Per testare ciò, si è quindi eseguito in parallelo sia batch layer che speed layer per un tempo di circa 10 secondi (richiesto per il completamento del ciclo di batch coi dati correnti) e ne è emerso che in così poco tempo i dati processati sono aumentati di una quantità pari a poco più di 70 tweets.

Riassumendo quindi, dai risultati degli esperimenti, emerge quanto la validità degli esiti raccolti in fase di sentiment analysis dipenda fortemente dalla quantità di dati computati, la quale a sua volta è correlata al numero di parole del dataset che contengono la query. L'esecuzione di un modulo real-time che in parallelo rispetto a quello batch vada a scaricare ed analizzare solo

quei tweets che contengono il termine in questione, consente al dataset di essere sempre più inerente con l'analisi che si vuole effettuare ed alla computazione futura di essere sempre più performante per via della suddivisione dei ruoli assegnati ai due diversi sistemi.

5. Sviluppi futuri e conclusioni

Per quanto riguarda un possibile ampliamento del lavoro svolto, il file .txt incluso nel progetto che associa ad ogni paese la propria lingua ufficiale, oltre al relativo nome in lingua nativa ne memorizza anche le coordinate geografiche in termini di latitudine e longitudine [2]. Si potrebbe pensare quindi di utilizzare tale informazione per catalogare i dati in base ai differenti paesi, in modo da comparare cioè il sentimento dei tweets contenenti una data query al variare degli stati considerati.

Per trarre infine delle conclusioni riguardo al lavoro svolto, dall'analisi dei risultati emersi a seguito degli esperimenti condotti, si evidenzia come la Lambda Architecture realizzata sia in grado di apportare notevoli benefici alla sentiment analysis.
L'affidabilità degli esiti ricavati da tale pratica difatti dipende considerevolmente dalla quantità di dati processati, la quale a sua volta deriva dal numero di tweets del dataset che contengono la query sulla base della quale questi si vogliono filtrare. In favore di ciò, l'architettura presentata consente in primo luogo, grazie allo speed layer, di allineare il dataset all'analisi richiesta aggiungendovi sempre più tweets che contengono la query prestabilita, ed inoltre, la presenza di due moduli distinti, consente di processare la grande mole di dati in arrivo senza penalizzare una nuova computazione dei tweets già elaborati in passato.

References

[1] Github project afinn-165-multilingual. https://github.com/dkocich/afinn-165-multilingual.

[2] Github project country-bounding-boxes. https://gist.github.com/graydon/11198540.

[3] The number of tweets per day in 2019. https://www.dsayce.com/social-media/tweets-day/.

[4] Sentiment140 dataset with 1.6 million tweets. https://www.kaggle.com/kazanova/sentiment140.