Twitter Analytics:
Big Data Pipeline

Matteo Wissel 534693, Federico Bianchi 534835
Dipartimento di Ingegneria Informatica Roma 3
Progetto Corso Big Data AA 2022/2023
Prof. Riccardo Torlone
Repository: https://github.com/magicWiss/BIG_DATA_TWITTER_ANALYSIS

Sommario

Introduzione

Nel mondo moderno, l'importanza dei dati è diventata cruciale per prendere decisioni informate e sviluppare soluzioni innovative. In questo contesto, i social network hanno assunto un ruolo di rilievo poiché generano enormi quantità di dati in tempo reale. Tra questi, Twitter si pone come un vero e proprio serbatoio di informazioni, rappresentando una fonte inesauribile di dati di interesse per la società e le aziende.

L'utilizzo dei dati di Twitter ha già dimostrato un impatto significativo nella società, con applicazioni in vari ambiti come l'analisi delle elezioni politiche, la rilevazione di emergenze, il monitoraggio delle tendenze di mercato e la prevenzione di fenomeni di disinformazione. Il potenziale di queste informazioni è in costante crescita, spingendo sempre di più verso la necessità di sviluppare soluzioni avanzate per l'analisi e la gestione di Big Data.

La seguente relazione si focalizza sulla progettazione e l'implementazione di un'architettura Big Data appositamente sviluppata per l'analisi dei dati di Twitter in modalità batch. Attraverso questo sistema, si intendono sfruttare le potenzialità delle tecnologie Big Data per estrarre valore, consentendo una migliore comprensione della società e dei suoi comportamenti online.

La relazione è organizzata nel seguente modo: il *Capitolo 1* introduce in breve il sistema, le tecnologie utilizzate ed i task di analytics perseguiti, il *Capitolo 2* approfondisce nel dettaglio l'architettura implementata giustificando le scelte implementative; il *Capitolo 3* presenta degli esempi di risultati generato delle analisi implementate; *Capitolo 4* analizza i risultati ottenuti dal sistema in termini di efficienza ed efficacia ed infine il Capitolo 5 definisce punti di miglioria del sistema.

Twitter Analytics: automated big data analytics pipeline

In questo Capitolo si introduce in breve il sistema implementato, si presenta il contesto applicativo mediante un user case reale, si analizzano i dati su cui il processo lavora ed infine vengono elencati i possibili insights estraibili dai dati.

Contesto applicativo (Use case)

Il sistema si propone di implementare un'architettura big data specializzata nell'analisi periodica dei dati estratti da Twitter.

Un utente ha interesse nell'analizzare periodicamente un insieme di dati estratti da Twitter.

I dati per il quale l'utente ha interesse effettuare analisi sono caricati in un repository centralizzato sottoforma di file di diverso formato (.csv o .json). Questi ultimi potrebbero essere memorizzati su file distinti (ognuno generato utilizzando logiche distinte).

Una volta eseguito, il processo implementato integra i dati presenti nei file, li processa estraendo nuove features utili per la fase di analisi ed infine produce delle dashboard (sia statiche che dinamiche) consultabili dall'utente utilizzatore per facilitare il suo processo decisionale.

Questo use case trova applicazione in ambiti come il monitoraggio delle opinioni dei clienti riguardo un prodotto o servizio, l'analisi della percezione dell'opinione pubblica su tematiche sociali o politiche, e il tracciamento di eventi e trend di rilevanza (Rodrigues, 2021).

Descrizione dati twitter

In questa Sezione ci si concentra nell'analisi dei dati e dei formati file supportati dal sistema.

Gli utenti di Twitter seguono una sintassi standard durante la pubblicazione di tweet, la quale comprende l'utilizzo di hashtag, retweet e menzioni di altri utenti. Gli hashtag sono parole o frasi precedute dal simbolo "#", mentre le menzioni consentono di menzionare altre persone, aziende, marchi o precisamente altri utenti di Twitter utilizzando il simbolo "@" seguito dal nome utente. Ogni tweet è limitato a 140 caratteri, il che permette agli utenti di pubblicare i propri messaggi in modo rapido e conciso.

La struttura dei dataset può variare a seconda della sorgente da cui essi vengono prelevati. Il formato standard, quello delle Twitter API, prevede un file json con strutture annidate contenenti una quantità di informazioni molto elevata. In questo caso, è possibile filtrare i soli campi di interesse per evitare un eccessivo overhead in termini di dimensioni.

Nel caso dei dataset CSV provenienti da Kaggle o da altri data pool, le informazioni sono molto più schematizzate e presentate in formato tabellare, senza campi annidati. Queste ultime richiedono meno elaborazioni per filtrare i dati necessari alle successive elaborazioni.

Una nota molto importante nell'analisi di questi dati è la necessità, anche nel caso di dati tabellari, di fare elaborazioni per estrarre l'autore del tweet a cui si risponde, in quanto tra i metadati sono specificati solamente l'id dell'autore del tweet corrente e l'id del tweet a cui si risponde. Oltre a questo, è necessario effettuare un parsing del testo per rimuovere eventuali simboli che possono influenzare le analisi.

Utilità analisi dati Twitter

Il sistema presentato in questa relazione può essere impiegato per diversi scopi di decision making.

Alcuni esempi di task sono:

- Monitorare l'andamento dell'opinione pubblica su temi specifici
- Individuare trend emergenti, identificare utenti influenti
- Analizzare l'efficacia di campagne di marketing e valutare il sentiment associato a un prodotto, servizio o evento
- Analizzare la concorrenza
- Migliorare il customer service

Si noti che, a partire dai dati grezzi estratti da Twitter, è possibile estrarre un quantitativo di conoscenza limitata. Per operare analisi più complesse, come quelle implementate nel nostro sistema, è necessaria un processamento più o meno complesso dei dati di input. La nostra implementazione viene discussa in breve nella sezione successiva e più in dettaglio nel Capitolo 3.

Sistema

La pipeline di analisi ideata è di tipo batch e può essere schedulata nell'eseguire le elaborazioni con cadenza prefissata. Inoltre, la pipeline permette di operare su più dataset in ingresso caricati in un repository centralizzato.

Quest'ultima caratteristica permette di utilizzare la pipeline anche nei casi in cui, per vari motivi, i tweet sul quale si vuole effettuare analisi sono memorizzati su file distinti e di diverso formato.

Di seguito si presenta una vista di alto livello dell'architettura implementata, dove per ogni stage della pipeline si definiscono le tecnologie utilizzate.

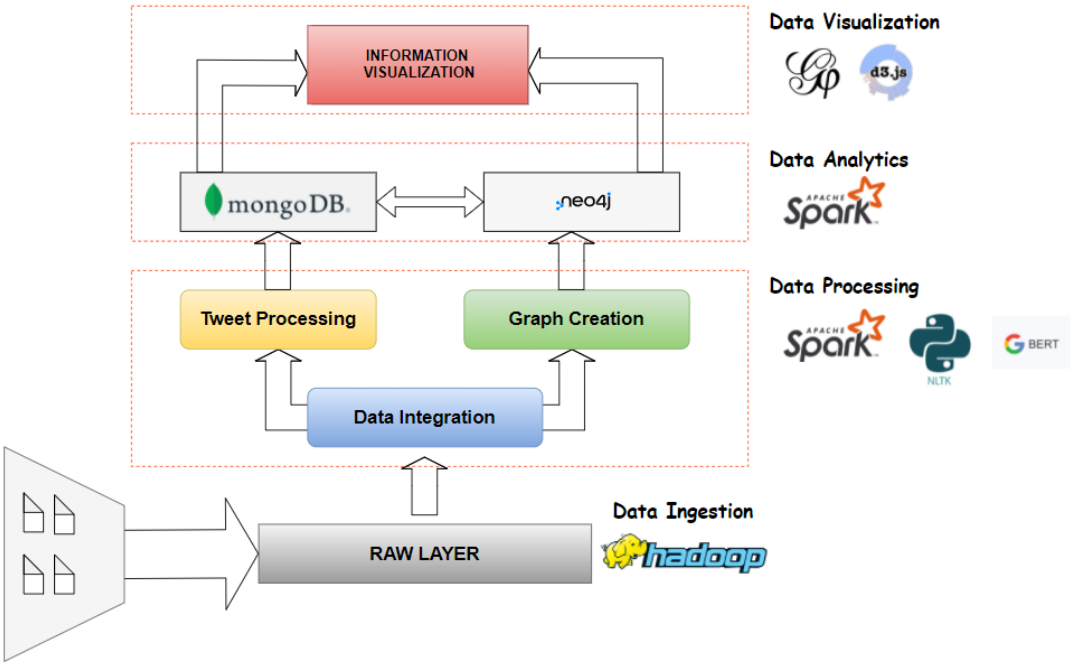


Figura 1 - Architettura alto livello

Il sistema è specializzato nell' elaborazione di dati prodotti sul social network Twitter. Le modalità di produzione di questi dati possono essere svariate tra cui:

- API Twitter utilizzando specifiche logiche di estrazione (key-words, intervalli temporali, specifici utenti)
- Dataset pre-elaborati estratti da archivi online (e.g. *Kaggle*)

Come si può notare dall'architettura, il sistema è trasparente a come i dati sono acquisiti. L'input del sistema è un insieme di file, potenzialmente di formato e struttura diversa, contenenti diverse informazioni relative ai tweet per il quale si vuole effettuare analisi (**varietà dei dati**).

Il sistema opera da prima una integrazione dei diversi dataset presenti nei file di partenza. Successivamente vengono eseguite una serie di operazioni complesse volte ad estrarre dal dataset grezzo nuove features, tra cui il sentiment del tweet, i topic discussi, gli hashtag utilizzati e le relazioni instaurate tra gli utenti.

Queste nuove features sono utilizzate per effettuare le analisi richieste. Queste ultime possono essere classificate nei seguenti gruppi:

- ***Tweet Focused***: basate principalmente sull'analisi del testo presente nei diversi tweet
- ***User Focused***: basate sull'analisi della rete sociale indotta dalle interazioni utenti nel social
- ***Hybrid*****:** basate sull'analisi dei dati generati dall'integrazione dei risultati prodotti dalla 1° e 2° categoria ed utili per estrarre insights più complessi (discussi nella Sezione *Task di analytics)*

Si presentano adesso, per ognuna delle tre categorie, i moduli implementati all'interno dell'architettura, ideati per estrarre una o più features utili ai fini dell'analisi.

Analisi Tweet Focused

- **Topic Analysis:** l'analisi dei topic trattati nel testo del tweet. Questo permette di categorizzare i tweet in base all'argomento di cui parlano
- **Sentiment Analysis:** l'analisi del sentiment del testo del tweet. Questo valore permette di capire qual è l'opinione generale dell'autore del tweet e può essere utilizzato anche per definire l'opinione di tale utente relativamente agli argomenti trattati

Analisi User Focused

- **Community Detection:** analisi della rete sociale che permette di generare gruppi di utenti più o meno grandi in base alla struttura della rete
- **User Centrality:** permette di analizzare la popolarità di un utente all'interno della rete sulla base delle interazioni che questo ha con gli altri utenti

Analisi Hybrid

- **Influence Analysis:** unendo le informazioni relative alla **topic analysis** all'analisi su grafo, si possono identificare gli utenti più influenti su un certo argomento all'interno della rete sociale

Tecnologie

In questa Sezione si analizzano le tecnologie utilizzate per la realizzazione dell'architettura. Si analizzano sia le tecnologie utilizzate per le operazioni di memorizzazione, sia le tecnologie utilizzate in ambito di processamento e visualizzazione dei dati.

Raw Layer: HDFS

Data la necessità di unire dati provenienti da diverse fonti ed in diversi formati, si è scelto di adottare un primo strato architetturale che consenta la memorizzazione di questo tipo "grezzo" di dati.

Per far ciò, la scelta è stata quella di utilizzare il File System Distribuito di Hadoop: ***HDFS***.

La scelta è stata motivata dall'alta efficienza di questo sistema, analizzato nella parte teorica del corso. Inoltre, l'esperienza pregressa dal lavoro svolto nel primo progetto, ha guidato ancor di più questa scelta.

Tra i vantaggi di Apache HDFS, l'*alta scalabilità*, che permette di gestire quantità sempre cresenti di dati, e l'*elevata affidabilità*, che permette di supportare grandi carichi computazionali in maniera efficiente, hanno reso questa tecnologia un punto di riferimento nel mondo dei Big Data.

MongoDB

La decisione di adottare MongoDB come database per la memorizzazione e l'analisi dei dati relativi a tweet è stata motivata da diverse considerazioni. Poiché i dati prodotti per le analytics (analizzato in modo più approfondito nel Capitolo 4) presentano una struttura intrinsecamente annidata, dove ad esempio ad ogni hashtag è associata una lista di utenti e viceversa, la flessibilità del modello di dati di MongoDB si è rivelata particolarmente adatta a gestire questa complessità senza dover ricorrere a uno schema rigido.

La struttura di memorizzazione di MongoDB in formato BSON ha offerto prestazioni efficienti durante le operazioni di lettura e scrittura su dati annidati.

Neo4J

Data la presenza di interazioni tra utenti, e quindi relazioni tra essi, la struttura a grafo si presta in maniera ottimale a rappresentare i dati a nostra disposizione.

Per gestire questo tipo di informazione si è scelto di utilizzare il Graph Database ***Neo4J***, tecnologia affermata nell'ambito dei dati contenenti relazioni.

Neo4J permette di accedere velocemente ai dati memorizzati al suo interno, anche grazie alla possibilità di definire degli indici su particolari attributi, come ad esempio l'identificativo del nodo (nel nostro caso l'utente).

Più in generale, la rete sociale di Twitter nel caso di tweet e risposte a tweet (che comprendono anche i *retweet,* la condivisione del tweet di un altro utente), è il caso più comune di struttura a grafo. I nodi di questo sono gli utenti (autori dei tweet) e gli archi rappresentano le interazioni tra gli utenti.

Oltre a questo, Neo4J offre la possibilità di effettuare analisi sul grafo attraverso l'uso di una libreria proprietaria che implementa i più comuni algoritmi di Graph Analysis. Questa libreria, chiamata **Graph Data Science**, offre delle API che permettono l'uso delle sue funzionalità anche in semplici script Python, questo facilita l'automazione delle analisi e l'aumentare della complessità di queste ultime.

SPARK

Spark è un framework altamente scalabile e distribuito, ideale per gestire grandi volumi di dati caratteristici del contesto Big Data. La sua capacità di calcolo in memoria permette di ottenere prestazioni elevate e tempi di esecuzione ridotti.

Spark offre una vasta gamma di librerie e moduli per l'analisi dei dati, tra cui Spark MLlib, che semplifica lo sviluppo di modelli di Machine Learning su dati distribuiti, sfruttando al massimo la potenza di calcolo di Spark. Tale libreria è stata utilizzata per la creazione dei modelli di topic modeling discussi nei capitoli successivi.

Per quanto riguarda la fase di analisi, Spark può essere facilmente integrato con MongoDB per via del connettore ufficiale, semplificando il recupero e l'elaborazione dei dati direttamente dal database, riducendo la complessità delle operazioni di analisi. Inoltre, l'utilizzo combinato di Spark e MongoDB permette di gestire grandi quantità di dati, eseguire analytics avanzati e implementare modelli di Machine Learning con efficienza e scalabilità, sfruttando al meglio le caratteristiche di entrambe le tecnologie.

NLTK

Per via della natura dei dati su cui si concentra l'architettura, si è avuto l'esigenza anche di implementare operazioni di processamento di linguaggio naturale. Per l'implementazione di queste operazioni, necessarie per l'addestramento dei modelli e l'estrazione di meta-informazioni dal testo, si è deciso di utilizzare NLTK.

NLTK (Natural Language Toolkit) è una libreria Python ampiamente utilizzata per il preprocessing, l'analisi e la manipolazione del linguaggio naturale. Essa fornisce una vasta gamma di strumenti e risorse per svolgere attività come tokenizzazione, stemming, analisi grammaticale e classificazione del testo. NLTK è uno

strumento prezioso per elaborare dati testuali in ambito di analytics, consentendo di estrarre significato e informazioni dai testi in modo efficiente e accurato.

D3.js

Un requisito per l'architettura è la possibilità dell'utente di interagire ed esplorare i risultati delle analytics condotte. Per questo requisito si è deciso di utilizzare D3.js.

D3.js è una libreria JavaScript molto popolare per la visualizzazione di dati utilizzando SVG, HTML e CSS. Essa offre un'ampia gamma di funzionalità per creare grafici interattivi e dinamici, visualizzazioni dati e dashboard, rendendola fondamentale per l'analisi e la comunicazione dei risultati delle analisi dei dati.

Con questa libreria, è possibile rappresentare dati complessi in modo visivamente accattivante e facilmente comprensibile, consentendo agli utenti di esplorare i dati e scoprire pattern o insight nascosti.

Gephi

Per lo stesso requisito indicato per l'utilizzo di d3.js, Gephi è stato selezionato come tool per la visualizzazione dei dati in formato grafo.

Gephi è un software open-source utilizzato per l'analisi e la visualizzazione di reti complesse. Questo strumento offre una vasta gamma di funzionalità per analizzare le relazioni tra nodi e archi all'interno di un grafo, permettendo di scoprire pattern e strutture nascoste nei dati. A differenza di Neo4J, Gephi non è un database, bensi uno strumento di visualizzazione. Per questo motivo, quest'ultimo offre molte funzionalità relative alla visualizzazione di grandi reti, tra cui algoritmi predefiniti specializzati in questo task, tra cui ForceAtlas e YifanHu.

Utilizzando Gephi, è possibile visualizzare reti complesse, identificare comunità di nodi, individuare hub e nodi periferici e rivelare altre proprietà significative delle reti analizzate. Questo software è particolarmente utile per l'analisi delle reti sociali, delle reti di comunicazione e molte altre applicazioni in cui le relazioni tra gli elementi sono di fondamentale importanza.

Architettura

In questo Capitolo si analizza nel dettaglio l'architettura presentata in *Figura 2*.

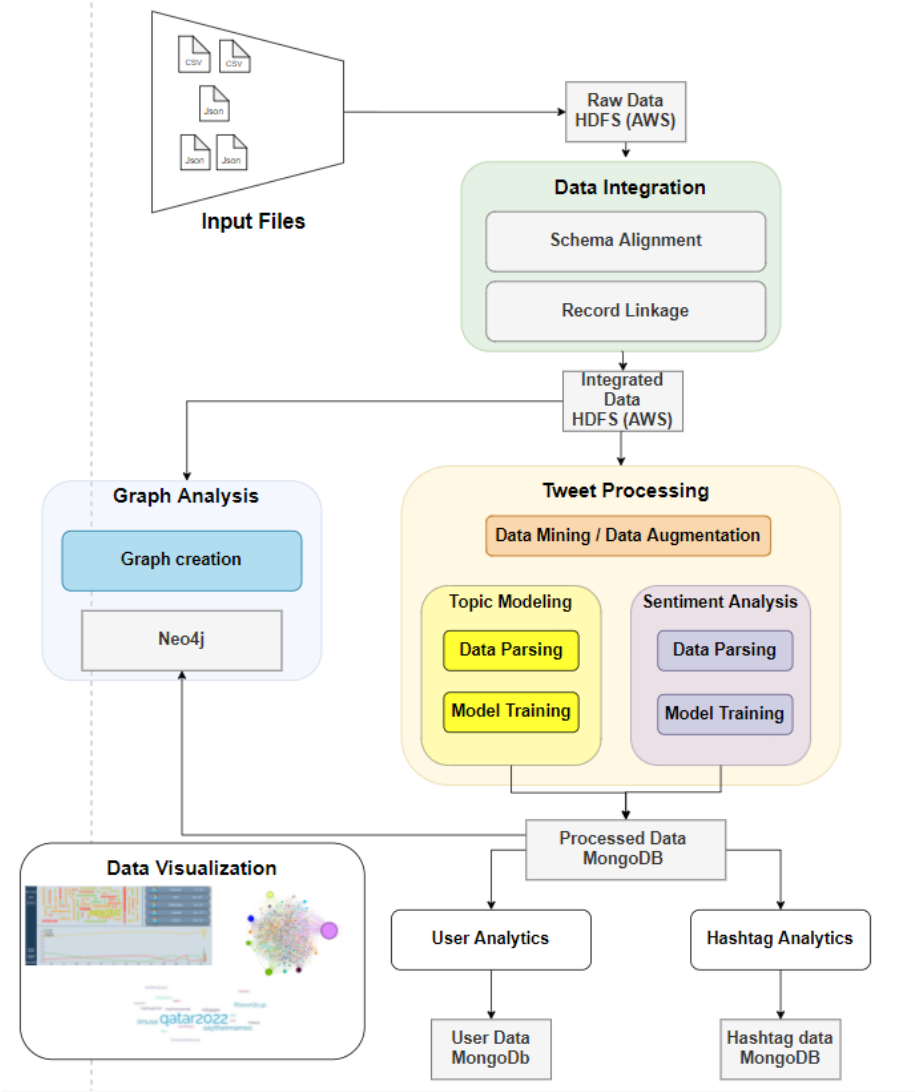


Figura 2 - Architettura

Data Acquisition

I dataset su cui opera il sistema (singolo file o file multipli) sono caricati in una cartella dedicata nell'HDFS. Il sistema supporta diverse tipologie di formati (.csv, .json).

Data integration (Varietà)

Il primo modulo dell'architettura è specializzato nell'affrontare la **varietà** dei dati in ingresso al sistema. Esso, infatti, implementa le operazioni di *Data Integration* necessarie per la standardizzazione ed unificazione dei dati su cui si vuole effettuare analisi. Il modulo è suddiviso in due componenti:

- Schema Alignment
- Record Linkage

Schema Alignment

La prima operazione operata sui dataset in ingresso è la trasformazione della struttura del loro schema.

In questa parte della pipeline ci si concentra sulla conversione dei dati in un formato standard (csv) e sull'unificazione del loro schema.

Per via della natura semi-strutturata dei dati in ingresso, si è deciso di definire uno schema unico (schema mediato) tale da poter mantenere le informazioni necessarie per operare le analytics prefissate.

Lo schema mediato scelto si presenta nel seguente modo:

$$tweet\_id, author\_id, conv\_id, conv\_author\_id, text, timestamp$$

Alcuni di questi campi sono estraibili in modo diretto (tweet_id, author_id, timestamp) per altri invece è richiesta una logica custom.

Record Linkage

Unificati i diversi dataset in un'unica struttura, è necessario eliminare eventuali duplicati. I duplicati possono generarsi per diversi motivi, ad esempio due dataset possono contenere delle sovrapposizioni e quindi alcuni tweet possono essere presenti in entrambi, soprattutto nei casi in cui i dataset appartengono a domini vicini tra loro (e.g., *'data science', 'data visualization'*)

Fondamentale per questo scopo è il campo *tweet_id*, in quanto identifica in modo univoco ogni singolo tweet prodotto nel social network di riferimento.

Output

L'output di questo modulo è un unico dataset in formato csv che contiene tutti i tweet sui quali si vuole effettuare l'analisi.

Il dataset finale è quindi memorizzato sul sistema HDFS e viene utilizzato dai successivi moduli ***Graph Analytics*** e ***Tweet Processing***.

Graph analytics

Questo modulo ha la responsabilità di gestire la creazione del grafo indotto dalle interazioni dei vari utenti e di caricare il risultato delle elaborazioni sul sistema Neo4j.

Grafo

Il grafo generato deve poter rappresentare le interazioni "*REPLIED_TO*" che si generano tra i diversi utenti. Questa relazione è intrinsecamente orientata, in quanto un utente X "risponde" ad utente Y, ma non è necessariamente vera l'affermazione contraria.

Per questo motivo la struttura del grafo ideata per gli scopi di analytics è la seguente:

- Nodi: **Users**
- Archi: direzionali da un utente ad un altro (identificati univocamente dal **tweet id** e dal **conversation id**). In particolare, un utente X risponde ad un utente Y quando uno dei tweet di X ha come *conv_author_id* il valore dell'id di Y

Si nota che nella versione base del grafo non sono presenti informazioni aggiuntive.

Le interazioni tra utenti d'interesse in questo contesto sono le relazioni di "*REPLIED_TO*" ovvero:

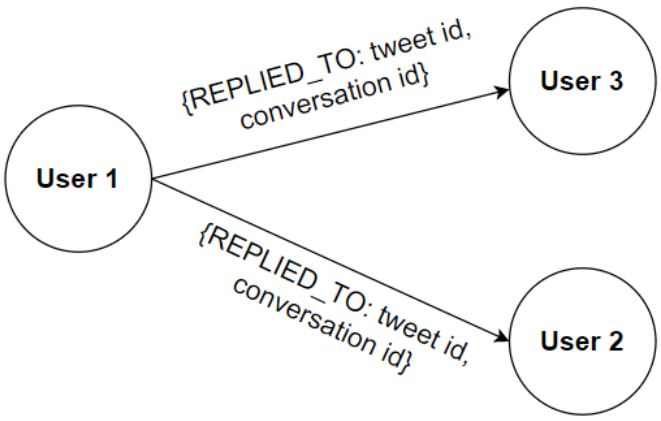


Figura 3-Struttura grafo Neo4j

Questo ci permette di poter individuare nel grafo "*community di utenti*" sfruttando i noti algoritmi di community-detection e "*user autorevoli*" sfruttando ad esempio l'algoritmo *PageRank*.

Creazione del grafo

Il grafo viene costruito a partire dal dataset "Integrated_data.csv". Per ogni riga si estraggono (se presenti) si estraggono i valori "authod_id", "tweet_id", "conversation_id", "conversation_author_id". Author_id viene utilizzato per identificare il nodo di partenza della relazione; conversation_author_id denota il nodo di arrivo dell'arco; tweet_id e conversation_id sono informazioni dell'arco e identificano univocamente l'arco all'interno del grafo.

Si noti che tweet_id è univoco, mentre conversation_id permette di aggregare tutti i tweet appartenenti alla stessa conversazione (in risposta ad uno stesso utente).

Community Detection

Questo tipo di analisi permette di definire dei gruppi di utenti correlati tra loro secondo vari criteri che dipendono dal particolare algoritmo.

Da queste analisi è possibile, ad esempio, rilevare *gruppi di utenti particolarmente attivi in un social network* e dunque identificare utenti che potrebbero essere 'fake' in quanto inattivi dopo un certo periodo di intensa attività o semplicemente utenti che hanno abbandonato il social network. O anche individuare *utenti che presentano delle affinità* e dunque targetizzare un certo tipo di utenti e fornire pubblicità mirata.

- **Weakly Connected Component:** questo algoritmo, presente nella suite della libreria *Graph Data Science*, permette di ottenere tutte le componenti debolmente connesse presenti all'interno del grafo. Per componente debolmente connessa si intende un sottoinsieme di nodi di un grafo in cui esiste un percorso tra tutti i nodi. Questa informazione andrà ad arricchire i nodi del grafo di una nuova proprietà su cui è possibile svolgere ulteriori operazioni di visualizzazione
- **Louvain:** l'obiettivo di questo algoritmo, anch'esso implementato dalla libreria *GDS*, è lo stesso di quello sopracitato. La principale differenza tra i due sta nella struttura dell'algoritmo stesso, che è in questo caso ricorsivo. In breve, ad ogni iterazione, l'algoritmo seleziona una comunità e la riassegna a una nuova comunità in modo da massimizzare la modularità del grafo. La modularità è una misura

di quanto i nodi di un grafo siano divisi in comunità. L'algoritmo continua poi ad iterare fino a quando non si possono più migliorare le comunità. Alla fine dell'esecuzione, verrà restituito un elenco delle comunità del grafo e dei nodi ad esse associati.

User Centrality

Questo tipo di analisi permette di valutare *l'importanza* dei nodi in un grafo in basse alle proprie connessioni.

- **Pagerank:** si basa sull'idea che la popolarità di un nodo è determinata dal numero di link che puntano ad esso e dall'importanza dei nodi che puntano ad esso.
  L'algoritmo PageRank funziona in modo iterativo. In ogni iterazione, viene assegnato ad ogni nodo un punteggio. Questo valore è determinato dalla somma dei punteggi dei nodi che puntano ad esso, ponderati per la loro importanza. L'algoritmo continua a iterare fino a quando i punteggi non si stabilizzano. Alla fine dell'esecuzione, verrà restituito un elenco di nodi, ordinati per importanza decrescente.

Tweet Processing

In questa Sezione si analizza il modulo *Tweet Processing*, modulo specializzato nell'arricchimento del dataset di partenza mediante estrazione di nuove features utili ai fini delle analisi.

Il modulo *Tweet Processing* è focalizzato nelle operazioni di *data augmentation*, ovvero nell'estrazione di informazioni e metadati a partire dai dati in ingresso.

Queste nuove informazioni permettono di comprendere in modo più dettagliato il dataset originale ed operare analisi più complesse.

Per operare tali trasformazioni il modulo *Tweet Processing* sfrutta le potenzialità di Machine Learning, Text Mining ed NLP.

Per garantire una suddivisione ottimale delle operazioni di trasformazione, il modulo si compone di 4 sotto moduli, ognuno responsabile dell'estrazione di una specifica nuova feature. Questi sotto moduli sono:

- **Text Mining (Hashtag-Extraction)**: modulo specializzato nell'estrazione dei diversi hashtag presenti nei tweet
- **Topic Modeling**: questo modulo è specializzato nella clusterizzazione dei tweet tramite l'utilizzo di due modelli non supervisionati distinti, LDA model e K-Means. L'obiettivo è quello di clusterizzare i tweet in base al topic discusso da ognuno di essi
- **Sentiment Analysis**: modulo specializzato nel calcolo del sentiment associato ad ogni tweet

Si analizzano ora i moduli precedentemente elencati.

Text mining

Il primo sotto modulo è specializzato nelle operazioni di text-mining dei tweet. Per questa prima versione dell'architettura ci siamo limitati nell'estrazione dal testo degli hashtag (keywords caratterizzate dalla presenza del carattere '#' come prefisso). Le keyword svolgono un ruolo di tag in Twitter e permettono di avere una sintesi succinta del contenuto del tweet.

Gli hashtag inoltre sono importanti in fase di monitoraggio dei trend in quanto gli stakeholders possono valutare l'andamento dei tweet in cui è utilizzato uno specifico hashtag da loro definito (vedi alcuni riferimenti).

Topic Modeling

In questa Sezione si analizza in breve l'operazione di *topic modeling* e si introducono le scelte implementative effettuate all'interno dell'architettura.

Il *topic modeling* è una tecnica di analisi del testo utilizzata nell'ambito dell'elaborazione del linguaggio naturale e del data mining. Consiste nell'identificazione e nell'estratto di temi o argomenti chiave presenti in un grande corpus di testi senza la necessità di etichettarli in anticipo. L'obiettivo principale è quello di scoprire automaticamente le strutture latenti all'interno dei testi, consentendo di raggruppare documenti simili in cluster e di identificare le parole più rilevanti associate a ciascun tema.

A tale scopo possono essere utilizzati due tipologie diverse di modelli che si differenziano rispetto al numero di topic associati ad uno specifico tweet, nello specifico:

- **LDA Model**: associa ad ogni tweet più topic
- **K-means**: associa ad ogni tweet un solo topic

Nell'architettura presentata, i dati in ingresso al sotto-modulo subiscono un processo di pulizia e tokenizzazione volta ad ottimizzare i risultati ottenuti dai due modelli. Inoltre, in quanto l'obiettivo è la creazione di un'architettura completamente automatizzata, i modelli utilizzano un set costante di iperparametri, i quali possono essere modificati mediante il processo ausiliare di *Data Quality* (discusso nella sezione *Data Quality*).

Si introducono adesso le scelte implementative relative alle fasi di preparazione dei dati, addestramento del modello *LDA* e addestramento del modello *K-means*.

Preparazione dati (Cleaning & Parsing)

In quanto il testo memorizzato nell'output del sotto-modulo *Text-Mining* è ancora un testo sporco (composto da hashtag, emoji, emoticon) per ottimizzare il risultato computato dai modelli *LDA* e *K-means*, i dati subiscono un processo di pulizia, volto ad eliminare eventuale testo garbage non utile ai fini della computazione. In quanto l'analisi dei dati è completamente automatizzata, si è optato per la realizzazione di una unica pipeline di *parsing* del campo *text* per entrambi i modelli.

Le operazioni implementate nella pipe-line di pulizia sono le seguenti:

1. Trasformazione testo in lowercase
2. Eliminazione caratteri di punteggiatura
3. Eliminazione stopwords
4. Eliminazione emoji
5. Lemming del testo

LDA Model

Il *Latent Dirichlet Allocation (LDA)* è un modello statistico generativo utilizzato per il *topic modeling* di testi. Consente di scoprire argomenti nascosti all'interno di un corpus di documenti e assegnare a ciascun documento una distribuzione di probabilità sugli argomenti (David M. Blei, Andrew Y. Ng, & Michael I. Jordan, 2003) .

I principali componenti del modello *LDA* includono:

1. ***Collezione di K argomenti (topics):*** *LDA* assume la presenza di K argomenti all'interno del corpus di documenti. Gli argomenti sono distribuzioni di probabilità sui vocaboli del corpus e rappresentano i temi nascosti che compaiono nei documenti.
2. ***Distribuzione di Dirichlet sui temi***: Per ogni argomento, è presente una distribuzione di *Dirichlet*, che modella la probabilità con cui i vocaboli del corpus sono associati all'argomento.
3. ***Distribuzione di Dirichlet sulle proporzioni di argomenti per ciascun documento***: Per ciascun documento, è presente una distribuzione di *Dirichlet*, che rappresenta la probabilità con cui gli argomenti compaiono nel documento.

L'obiettivo del modello è stimare i parametri di queste distribuzioni, ovvero le distribuzioni di argomenti sui vocaboli e le distribuzioni di proporzioni di argomenti nei documenti. Questi parametri vengono stimati utilizzando l'algoritmo di inferenza basato su tecniche MCMC (Markov Chain Monte Carlo) o variazionale (David M. Blei, Andrew Y. Ng, & Michael I. Jordan, 2003).

Il modello può essere sintetizzato dalla seguente equazione:

$$p(D|\alpha,\beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn},\beta) \right) d\theta_d.$$

Figura 4-Formula per il calcolo dei topics associati ad un testo nel modello LDA

Dove:

- **θ_d (Dirichlet(α)):** Distribuzione di proporzioni di argomenti per il documento d
- **z_di (Multinomial(θ_d)):** Indice dell'argomento associato alla parola i nel documento d
- **w_di (Multinomial(β_{z_di})**: Parola osservata per l'indice di argomento z_di
- **α**: è il parametro di smoothing per la distribuzione di *Dirichlet* sui documenti (dirà quanto i documenti sono concentrati su pochi argomenti o distribuiti su molti)
- **β**: è la matrice delle distribuzioni di probabilità dei vocaboli sugli argomenti

Gli iperparametri utilizzati per l'implementazione di questi modelli possono sono i seguenti:

1. numero topics per documento: 4
2. Alpha: auto*
3. Betha: auto
4. Numero iterazioni: 10000
5. Threshold rilevanza: 0.25

Poiché la pipeline è completamente automatizzata, questi iperparametri sono stati selezionati mediante *l'elbow-method* confrontando i risultati ottenuti su dataset di diverso tipo e grandezza.

L'output di questa computazione è una lista di topic associata ad ogni record.

Tweet_id	…	Topic
1000006122852536320		[0,1,8]
1003361453254656		[4,1,2,7]

K-means

K-means è un algoritmo di *clustering* non supervisionato, il cui obiettivo è quello di suddividere un insieme di punti dati in K cluster, in modo tale che i punti all'interno di ciascun cluster siano simili tra loro e che i cluster siano il più differenti possibile gli uni dagli altri.

Il modello *K-means* può essere adattato al task di *topic modeling* mediante rappresentazione numerica del testo, utilizzando un approccio *bag-of-words (BoW*) o il *term frequency-inverse document frequency (TF-IDF).* Successivamente, *K-means* può essere applicato a questi vettori per raggruppare i documenti in K *cluster*.

Un potenziale svantaggio di utilizzare *K-means* per il *topic modeling* è che questo non tiene conto delle strutture probabilistiche complesse che possono sottostare ai dati testuali. Pertanto, i risultati ottenuti potrebbero non essere altrettanto significativi o interpretabili come quelli ottenuti con metodi di *topic modeling* specifici come *LDA* o *PLSA*.

A differenza del modello *LDA*, analizzato precedentemente, per il *K_means* si è optato per un tuning semi-dinamico del modello. I parametri utilizzati per l'addestramento del modello sono i seguenti:

1. K (numero di clusters): pari a 3 se il file è unico, numero di file altrimenti
2. Numero massimo iterazioni: 1000
3. Seed: 37

Anche in questo caso, fatta eccezione per il valore k, gli altri iperparametri sono stati scelti seguendo lo stesso processo visto per *LDA*.

La nuova feature viene inserita all'intero del dataset processato.

Tweet_id	…	Topic	Cluster
1000006122852536320		[0,1,8]	1
1003361453254656		[4,1,2,7]	3

Sentiment Analysis

L'operazione di *sentiment analysis* permette di identificare lo stato emozionale dell'utente nella scrittura del tweet. Questa operazione viene svolta con tecniche di *Natural Language Processing* che possono essere basate sull'uso di *keywords*, sull'*analisi delle espressioni e di pattern* o, più di recente, con l'uso di architetture *encoder/decoder* che vengono pre-addrestrate su grandi moli di dati e offrono quindi una velocità di elaborazione molto alta, oltre ad un'elevata precisione nel giudizio.

In questo progetto sono state esplorate due tecnologie per questa operazione:

Sentiment Analysis con l'uso del solo BERT

Una prima soluzione prevedeva l'utilizzo di modelli BERT offerti dalle API di Tensorflow. Si è quindi utilizzato il modello *'cardiffnlp/twitter-roberta-base-sentiment'* sviluppato dal gruppo di ricerca in NLP della Cardiff University. Questo modello sfrutta le potenzialità di *RoBERTA* ed è stato poi *'fine-tuned'* su un dataset di tweet.

Questo modello fornisce, per ogni testo processato, un array composto da tre valori reali che indicano rispettivamente il valore di *positività* del testo, il valore di *sentimento neutro* del testo ed il valore di *negatività* del testo. Questo permette un'analisi molto approfondita dei tweet ma richiede un'altissima

quantità di tempo, in quanto la libreria lavora al livello di singolo tweet ed il numero di testi nel dataset può essere molto elevato.

Sentiment Analysis con le pipeline di SparkNLP

Date le difficoltà incontrate nell'implementazione della prima soluzione si è scelto di utilizzare le pipeline offerte dalla libreria SparkNLP, sviluppata dall'azienda John Snow Labs.

Come si può intuire dal nome, questa libreria è basata su Apache Spark ed eredita quindi tutti i vantaggi che appartengono ad essa, tra cui la *grande velocità di esecuzione*, *l'alta scalabilità* e la sua *efficienza*.

In questo caso ad ogni tweet viene associato un solo valore, una stringa, che indica se il testo ha una visione positiva o negativa rispetto all'argomento che tratta.

Con questa soluzione i tempi di esecuzione sono stati abbattuti ed infatti si passa dall'ordine delle ore della prima ai pochi minuti di questa.

Collections and Creation (MongoDB)

In questa Sezione si definisce la logica relativa alla creazione ed il caricamento delle collezioni MongoDB mediante Spark.

Inoltre, si giustificano alcune scelte relative allo schema prescelto per le collezioni generate.

Per effettuare le analisi descritte in precedenza, si è deciso di creare cinque collezioni distinte, ognuna utile per uno specifico task di analisi.

Le collezioni scelte sono le seguenti:

- Tweets
- User2Hashtag
- Hastag2Users
- User2Setiment2Date
- Hashtag2Sentiment2Date

Tweets Collection

Tweets è la collezione centrale. Qui vengono caricati tutti i risultati delle elaborazioni analizzate in precedenza. L'aggregato rappresentato in questa collezione è il singolo tweet. Lo schema seguito è il seguente:

{
tweet_id: *(id univoco associato al tweet)*
author_id: *(id univoco associato all'utente)*
converstion_author_id: *(id univoco dell'utente a cui si risponde)*
conversation_id: *(id univoco della conversazione)*
timestamp: *(data di pubblicazione)*
hashtags: *(lista di hashtag usati nel tweet)*
parsed_text: *(testo parsato)*
prediction: *(cluster computato da Kmeans)*
limitedTopics: *(lista di topics associati computati con LDA)*
sentiment: *(valori computati in termini di positivo, medio, negativo)*
generalSentiment: *(predizione sul sentiment)*
}

Per ottimizzare l'accesso a specifici tweet, si è creato un indice sul campo *tweet_id*. Si nota che a partire da questa collezione vengono create le collezioni secondarie, utilizzate in fase di analisi per alimentare le dashboard di visualizzazione.

User2Hashtag Collection

La collezione user2hashtag memorizza per ogni utente la lista degli hashtag che ha utilizzato.

{
- **UserId:** *(id univoco dell'utente)*
- **Total**: *(Numero totale di tweet pubblicati)*
- **Hashtags**: *(Lista di hashtag con numero di occorrenze es.{"qatar2022":100})*
- **Clusters**: *(Lista dei cluster e occorrenze in cui i suoi tweet sono stati aggregati es. {"0":100,"1":600})*
- **Topics**: *(Lista dei topics e occorrenze in cui i suoi tweet sono stati aggregati es. {"0":100,"1":600})*
- **Sentiment**: *(media del sentiment positivo, medio e negativo associato ai suoi tweet)*

}

Questa collezione facilita le interrogazioni lato utente, e permette di estrarre informazioni tra cui:

1. *Chi sono gli utenti più attivi*
2. *Quali hashtag hanno utilizzato maggiormente*

Inoltre, è la collezione utilizzata per alimentare la parte dedicata agli utenti nella dashboard d3.

Hashtag2User Collection

La collezione Hashtag 2 users ha lo scopo di memorizzare per ogni hashtag la lista di utenti che hanno utilizzato quello specifico hashtag.

{
- **Hashtag:** *(hashtag in formato stringa)*
- **Total**: *(Numero totale di tweet in cui è usato l'hashtag)*
- **Users**: *(Lista di users con numero di occorrenze es. {"124421245":100})*
- **Clusters**: *(Lista dei cluster e occorrenze in cui l'hashtag è stato aggregato es. {"0":100,"1":600})*
- **Topics**: *(Lista dei topics e occorrenze in cui l'hashtag è stato aggregato es. {"0":100,"1":600})*
- **Sentiment**: *(media del sentiment positivo, medio e negativo associato all'hashtag)*

}

Questa collezione facilita le interrogazioni lato hashtag e permette di estrarre informazioni importanti tra cui:

1. Hashtag più utilizzati
2. Quali utenti utilizzano maggiormente uno specifico hashtag

User2Sentiment2Date

La user2sentiment2date è utilizzata per analizzare l'evoluzione del sentiment computati sui tweet di uno/o più utenti.

{

UserId: *(Id univoco dell'utente)*
Timestamp: *(Data pubblicazione tweet)*
Positive: *(Valore sentiment medio positivo dell'utente in quel giorno)*
Negative: *(Valore sentiment medio negativo dell'utente in quel giorno)*
Medium: *(Valore sentiment medio medio dell'utente in quel giorno)*

}

L'aggregato modellato in questa collezione è l'utente ed il tempo.

Questo permette di estrarre informazioni quali, ad esempio:

1. *Che andamento ha il sentiment degli utenti per uno specifico periodo temporale?*
2. *Qual è il sentiment medio di uno specifico user X*

È utilizzata per alimentare la parte di dashboard d3 relativa al sentiment alla selezione dell'utente.

Hashtag2Sentiment2Date

La collezione hashtag2sentiment2date è speculare alla precedente. Modella il sentiment associato agli hashtag. Permette quindi di rispondere alle stesse domande del dataset precedente relativo però agli hashtag.

{

Hahstag: *(Hashtag in formato stringa)*
Timestamp: *(Data pubblicazione tweet)*
Positive: *(Valore sentiment medio positivo dell'utente in quel giorno)*
Negative: *(Valore sentiment medio negativo dell'utente in quel giorno)*
Medium: *(Valore sentiment medio medio dell'utente in quel giorno)*

}

L'aggregato modellato in questa collezione è l'utente ed il tempo.

Questo permette di estrarre informazioni quali, ad esempio:

1. *Che andamento ha il sentiment degli utenti per uno specifico periodo temporale?*
2. *Qual è il sentiment medio di uno specifico user X*

È utilizzata per alimentare la parte di dashboard d3 relativa al sentiment alla selezione di un hashtag.

MongoDB - Neo4J Enrichment

In questa Sezione si analizza il meccanismo di *data enrichment* volto ad arricchire i dati presenti sui sistemi *MongoDB* (nello specifico la collezione Users) e *Neo4J,* mediante integrazione dei risultati delle analisi operate sui due sistemi.

Nello specifico, l'obiettivo è quello di integrare i risultati ottenuti dai dati presenti nei due sistemi per rispondere a domande di analisi più complesse tra cui:

- Gli utenti più influenti su un determinato topic
- I topic maggiormente discussi nelle diverse community
- Il sentiment generale relativo ad uno specifico hashtag in una specifica community

Per effettuare questa operazione è necessario un meccanismo di integrazione tale da combinare i risultati delle analisi operate sui singoli sistemi.

Influence Analysis per topic

Una volta estratti i topic trattati all'interno dei tweet salvati in *MongoDB*, la rete in *Neo4J* è stata filtrata e ancora analizzata per identificare i nodi più *influenti* al suo interno.

Per far ciò è stato utilizzato l'algoritmo per il calcolo della '***Closeness Centrality***', che misura la distanza media di un nodo rispetto a tutti gli altri nodi del grafo.

Si noti che anche l'algoritmo *Pagerank* offre una buona metrica riguardo all'influenza di un nodo sulla rete ma si è scelto di variare per esplorare ulteriori possibilità.

Analytics ed esempi di risultati

Si presentano in questo Capitolo le analisi e degli esempi di risultato ottenuti dal sistema utilizzando cinque dataset in input, ognuno dei quali focalizzato su uno specifico topic tra cui: *Mondiali Qatar 2022, Guerra Ucraina Russia, Data Science.*

Tweet focused

Per quanto riguarda l'analisi specializzata sui tweet (*hashtag analysis, topic analysis, sentiment analysis*) il sistema offre due tipologie di dashboard distinte, una dinamica, implementata in d3.js, la seconda statica.

Dashboard Dinamica: Hashtag-User-Sentiment analysis

La dashboard in questione, implementata in d3.js nell'ambito del corso ***Visualizzazione delle Informazioni*** con un nostro collega, è concentrata sull'analisi delle relazioni tra hashtag e users.

È dunque possibile individuare per uno specifico hashtag la lista di utenti che lo hanno utilizzato maggiormente e viceversa: selezionato un utente, è possibile individuare quali hashtag questo ha utilizzato più spesso.

Oltre a queste informazioni, la dashboard permette di analizzare l'andamento del sentiment dei tweet in cui compare uno specifico hashtag o l'andamento dei tweet dell'utente selezionato.

Un esempio di istanza di questa dashboard è visionabile al link: https://magicwiss.github.io/visualization/

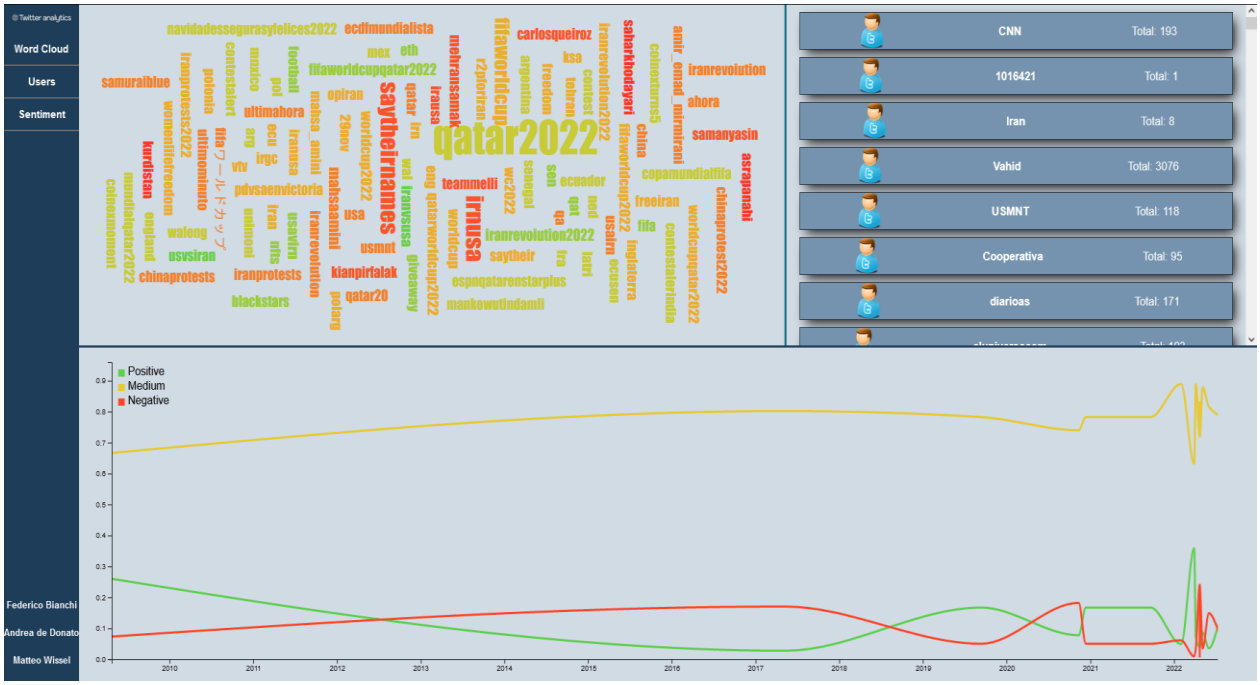


Figura 5 - Dashboard Dinamica

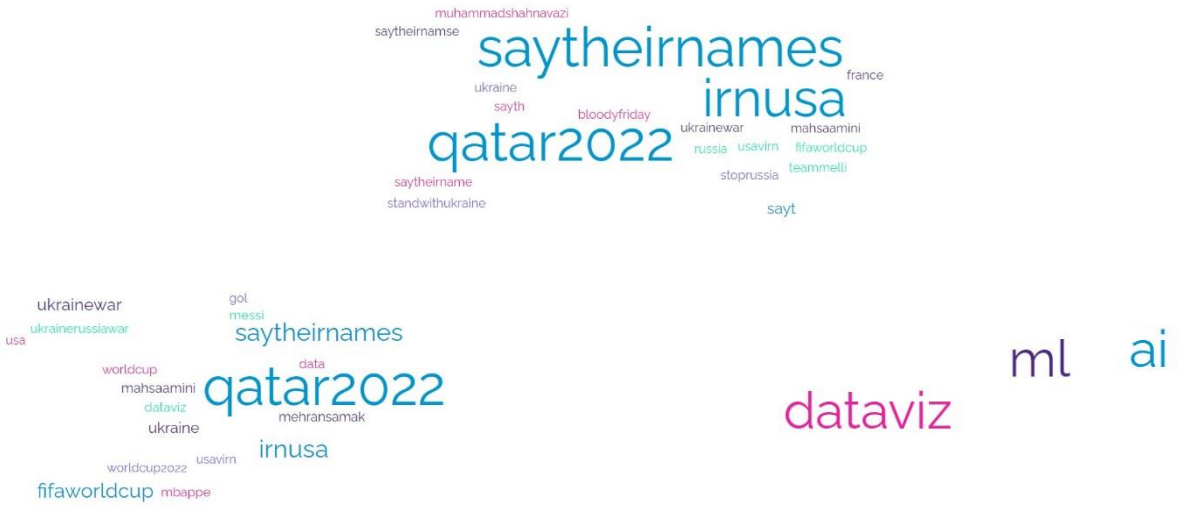
Dashboard Statica: Topic to Hashtag analysis

Queste dashboard si concentrano sul mostrare come gli hashtag si distribuiscono nei diversi topic-clusters. Tale visualizzazione permette di individuare in che gruppo/gruppi si collocano i diversi hashtag e quindi il contesto in cui sono utilizzati.

Hashtag to topic

Wordcloud per gli hashtag dei topic: ***Qatar2022, UkraineWar, Lancio ChatGPT***

Hashtag cluster



Wordcloud per gli hashtag dei cluster

User Focused Analytics: Risultati

Weakly Connected Components

Il grafo delle community generato dall'algoritmo Weakly Connected Components mette in evidenza la presenza di una componente connessa che domina sulla struttura. Questo perché è molto probabile che esista un percorso che collega due nodi anche molto distanti tra loro nella rete sociale. Il resto dei nodi sono quindi quelli che appartengono a contesti estremamente diversi o, ancora, utenti inattivi o 'fake' che non hanno avuto interazioni significative con il resto della rete.

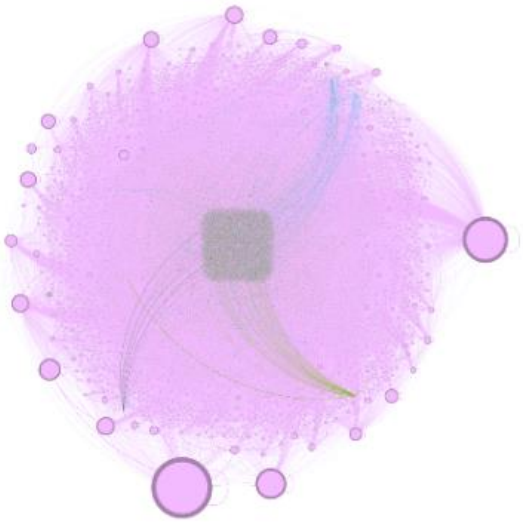


Figura 6 - grafo WCC

Louvain

Il grafo generato dall'esecuzione di Louvain invece presenta un maggior numero di community

.

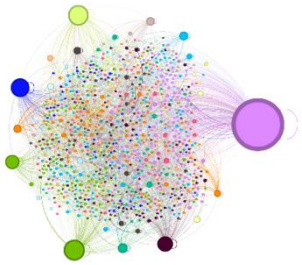


Figura 7 - sottografo Louvain

Questo tipo di visualizzazione può fornire risultati più significativi di quella precedente in quanto effettua delle ulteriori partizioni sulla rete e quindi è possibile analizzare più nel dettaglio le relazioni presenti tra gli utenti.

Il grafo nella prima figura mostra un sottoinsieme dei nodi presenti nella rete, in particolare quelli con score Pagerank maggiore, mentre quello in basso mostra la situazione con tutti i nodi della rete.

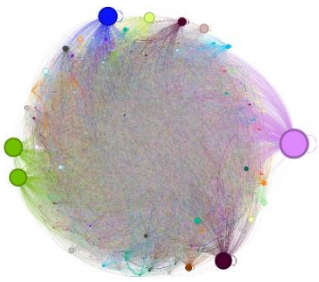


Figura 8 - grafo completo Louvain

Come si può notare, le community sono comunque molto connesse tra loro e questo evidenzia la relazione con il grafo indotto dai risultati di WCC. Infatti, c'è un'alta probabilità che i nodi delle diverse community abbiano comunque avuto qualche interazione tra loro.

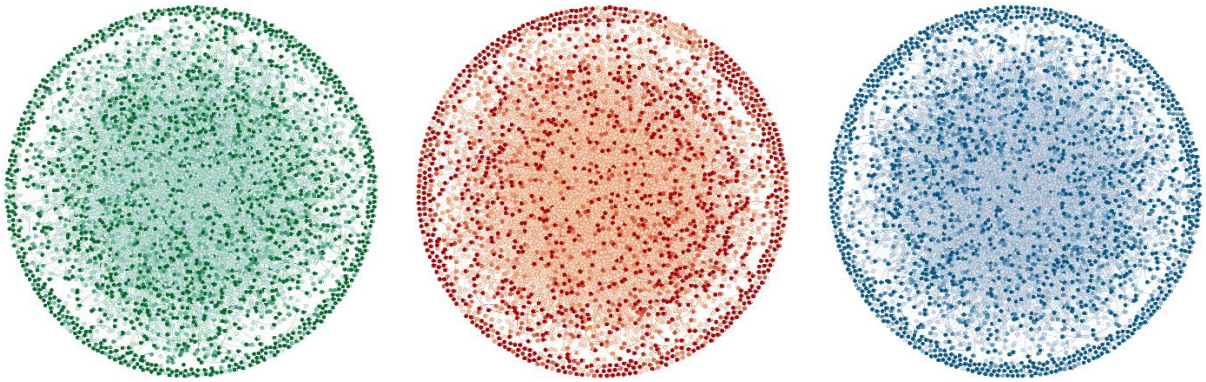
Pagerank

Nei grafi delle sezioni precedenti, i nodi sono scalati nella loro dimensione in proporzione al proprio score Pagerank. Si può notare come ci siano, rispetto al numero totale dei nodi nel grafo, pochi nodi molto più popolari rispetto agli altri, mentre i restanti si attestano su uno score pressoché uguale per tutti. Questo evidenzia la presenza di influencer o comunque personaggi di caratura mondiale all'interno della rete.

Hybrid Analysis: Risultati

Di seguito sono mostrati i grafi relativi ai tre topic più comuni nel nostro dataset. In particolare, questi grafi sono stati filtrati in base all'appartenenza del tweet al determinato topic e successivamente è stata calcolata la *Closeness Centrality* di ogni nodo rimanente dal filtraggio.

Come si può notare dalla scala dei colori, la prevalenza dei nodi della rete non ha uno score di *Closeness Centrality* molto alto, mentre una parte dei restanti possiede sicuramente un valore più elevato, e questi rappresentano gli utenti più influenti, ovvero quelli con il maggior numero di interazioni con il resto della rete.



La struttura dei tre grafi è molto simile e questo ci conferma il pattern che si presenta in questo tipo di reti sociali, ovvero la presenza di molti utenti, di cui però solo alcuni riescono a spiccare per la popolarità.

Valutazione sistema

In questo Capitolo si effettua un'analisi prestazionale del sistema, in termini di efficienza ed efficacia. Per valutare le performance del nostro sistema abbiamo utilizzato diversi test-case, ognuno dei quali caratterizzato da un volume di dati più grandi memorizzati su 1 o più file.

Efficacia

L'efficacia del sistema è stata misurata in base ai risultati intermedi prodotti dai diversi moduli. Nello specifico, l'efficienza complessiva del sistema è stata calcolata testando il modulo di integrazione, il modulo topic modeling ed il modulo di sentiment. Abbiamo scelti questi moduli in quanto influenzano in modo diretto la qualità delle analisi riportate dal sistema.

Data integration: Efficacia

L'efficacia di questo modulo è stata misurata verificando la corretta eliminazione di tweet duplicati e la corretta individuazione delle relazioni Reply_to.

Per lo scopo, sono stati utilzzati 4 test case diversi, ognuno dei quali aventi caratteristiche distinte rispetto a numero di file in input, numero di record in input, tipologia file in input, numero duplicati.

Numero file in input	Numero record in input	Duplicati reali	Duplicati individuati	Relazioni individuate
1	50k	20	18*	90%
2	100k	200	200	100%
3	150k	2000	2000	100%
4	175K	20000	20000	100%
5	200K		20000	20000	100%

*si nota che l'unico test case fallito è il primo. Questo poiché, nella logica iniziale il campo tweet_id, campo sulla quale si base il processo di deduplica, veniva convertito in intero. Per via di uno scale troppo elevato associato ad alcuni di questi valori, il tweet_id è stato troncato, comprtando quindi la non deduplica di alcuni record

LDA Model: Efficacia

Le metriche utilizzate per la valutazione del modulo LDA sono:

- **Perplessità (Perplexity)**: La perplessità è una misura di quanto bene il modello è in grado di predire il corpus di testo. Minore è il valore della perplessità, migliore è la capacità predittiva del modello.
- **Coerenza degli argomenti (Topic Coherence)**: La coerenza degli argomenti valuta quanto coerenti e interpretabili sono gli argomenti estratti dal modello. Questa metrica misura quanto le parole associate a un argomento sono semanticamente simili tra loro.

In quanto gli iperparametri utilizzati non vengono modificati per ogni run, per garantire la completa automazione del processo, abbiamo definito una soglia minima di accuratezza, per entrambe le metriche.

Anche in questo caso sono stati preparati diversi test-case, ognuno avente un numero di record diversi e soprattutto riguardanti argomenti differenti.

I risultati sono i seguenti.

Test case	Perplexity target (max)	Perplexity real	Topic Coherence target	Topic Coherence real
1	35	29,12	0,70	0.80
2	35	15,1189	0,70	0.87
3	35	30,124	0,70	0.69
4	35	12,33	0,70	0.90

K-means: Efficacia

Le metriche utilizzate per la vautazione di questo modulo sono:

- **SSE (Sum of Squared Errors)**: La SSE calcola la somma dei quadrati delle distanze tra ciascun punto del dataset e il centroide del cluster a cui è assegnato. Una minore SSE indica una maggiore compattazione dei cluster.
- **Silhouette Score**: Il punteggio silhouette misura quanto bene ogni punto si adatta al proprio cluster rispetto agli altri cluster. Un punteggio silhouette più vicino a 1 indica un'assegnazione del cluster migliore.

I test sono stati effettuati con gli stessi test-case utilizzati per il modulo LDA. I risultati sono i segueti:

Test case	SSE target	SSE real	Silhouette target	Silhouette real
1	20	18	0.5	0.675
2	20	15	0.5	0.8
3	20	9	0.5	0.75
4	20	24	0.5	0.58

Efficienza

L'efficienza del sistema è stata misurata analizzando il tempo impiegato per la produzione dei risultati finali.

Di seguito riportiamo l'andamento del tempo richiesto per l'esecuzione della pipeline al crescere del numero di dati considerati.

Test case	#rows (K)	Time (mm)
1	50	30
2	100	39
3	150	45
4	250	48
5	500	52

Il cui andamento è riportato nella seguente figura:

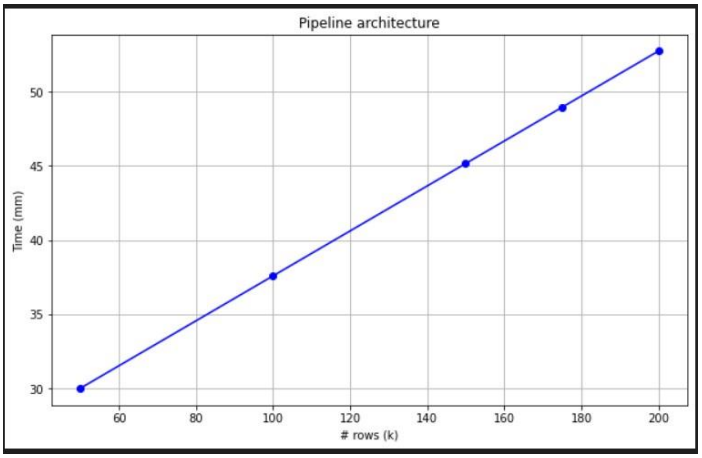


Figura 9 - andamento tempi

Si può notare come il tempo totale dell'esecuzione cresce in maniera proporzionale al volume di dati considerato.

Inoltre, analizzando il grafico in Figura [10] è possibile verificare come si distribuisce il tempo di esecuzione rispetto ai singoli moduli.

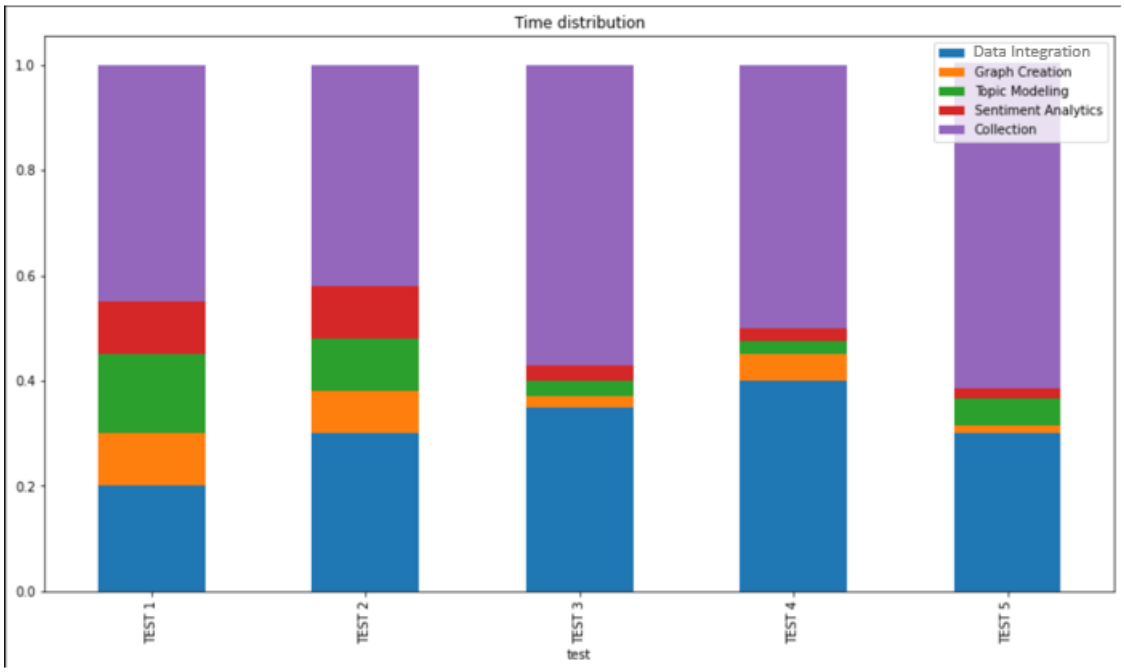


Figura 10 - dettaglio tempi di esecuzione

Come si può notare, le proporzioni si tengono più o meno costanti al crescere del volume di dati.

Sentiment Analysis: un confronto

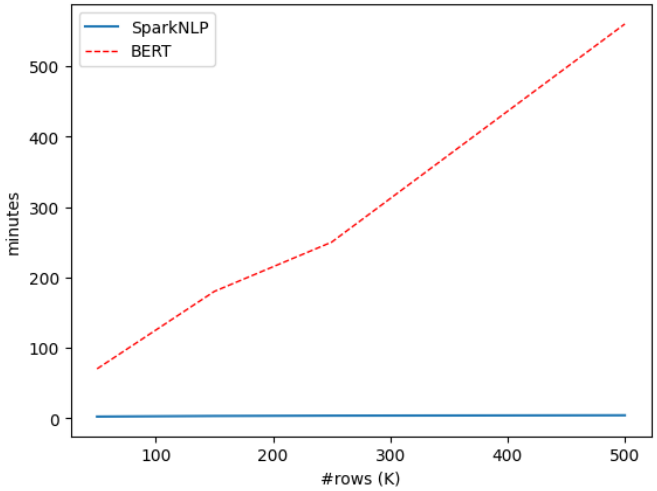


Figura 11 - dettaglio tempi Sentiment Analysis

Come accennato nella parte descrittiva delle tecnologie, un grande ostacolo evidenziato dai primi test è stato quello dell'eccessivo tempo di elaborazione nella fase di *sentiment analysis.* Questo accadeva con l'utilizzo del modello *BERT* associato all'esecuzione su RDD di Apache Spark.

Il problema di questo approccio, però, era l'esecuzione del modello su ogni singola cella di testo, e questo richiede una notevole quantità di tempo.

Con l'utilizzo della libreria SparkNLP i risultati sono migliorati in maniera esponenziale. Come si può notare dal grafico l'ordine di grandezza del tempo di esecuzione passa dalle ore ai minuti.

Inoltre, è interessante notare come con l'utilizzo di quest'ultima tecnologia il tempo di esecuzione rimanga pressoché costante sui quattro minuti, mentre utilizzando il solo modello BERT il tempo di esecuzione richiesto è direttamente proporzionale alla quantità di testi da elaborare.

Miglioramenti architetturali – Lavori futuri

In questo Capitolo si analizzano alcune migliorie che potrebbero essere implementate all'interno dell'architettura per ottimizzare i risultati e l'efficienza generale del sistema.

Data Quality

Dall'architettura si evince come tutti i risultati intermedi prodotti a partire dall' integrazione dei dataset nel primo layer siano memorizzati in maniera permanente all'interno del sistema *HDFS*. Questa scelta permette la realizzazione di un sistema di *Data Quality* volto ad individuare eventuali points of failure del sistema.

Più nello specifico, i task del *data quality* si concentrano sul monitoraggio della performance dei diversi moduli presenti all'interno della pipeline e hanno come obiettivo quello di garantire dati accurati e affidabili, nonché permettere di individuare eventuali problematiche o aree di miglioramento.

Meta-Learning

Come indicato nella Sezione *Topic Modeling,* una scelta implementativa operata sul sistema per garantire l'automazione delle analisi è la costanza degli iperparametri utilizzati per i due modelli non supervisionati.

Questa scelta garantisce l'automatizzazione del processo di analisi, ma potrebbe portare alla computazione di risultati non ottimali.

Per questo motivo, a monte dell'addestramento dei modelli *LDA* e *K-means*, potrebbe essere inserito un sistema di *Meta-Learning* volto a stimare il valore sub-ottimale degli iperparametri considerando ad esempio caratteristiche statistiche e lessicali dei dati in ingresso.

Nello specifico, il "*Meta Learning"*, o "Learning to Learn", è una branca dell'apprendimento automatico che mira a sviluppare modelli di machine learning in grado di apprendere come imparare. L'obiettivo è migliorare l'efficienza e la capacità di generalizzazione dei modelli, consentendo loro di adattarsi rapidamente a nuove situazioni basandosi sulle esperienze di apprendimento passate. In sostanza, il meta learning rende i modelli più adattabili, efficienti e capaci di generalizzare meglio su nuovi problemi anche con pochi dati di addestramento" (Levine).

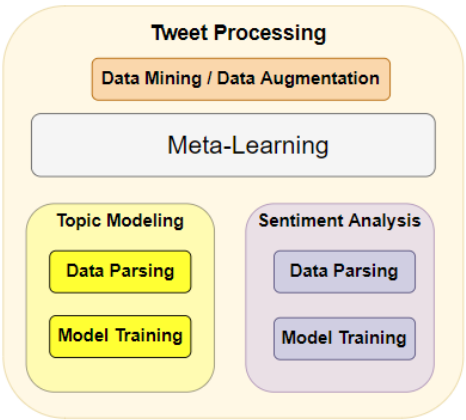


Figura 12 - Topic Modeling con modulo Meta Learning

Il sistema di *Meta-Learning* garantirebbe sia un incremento dell'efficienza dei modelli sopra citati sia l'automatizzazione del processo in quanto l'utente, salvo casi eccezionali in cui i modelli performano sottosoglia, non dovrebbe intervenire per effettuare l'operazione di tuning.