ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea Magistrale in Ingegneria Informatica

TESI DI LAUREA

in

Sistemi Distribuiti M

Supporto ad Applicazioni di Web Reputation basate su Piattaforma Apache Storm

CANDIDATO
Angelo Maglione

RELATORE:
Prof. Ing. Paolo Bellavista

CORRELATORE:
Prof. Ing. Antonio Corradi
Dott. Alessandro Proscia

Anno Accademico 2015/16

Sessione III

Ringraziamenti

Desidero, in primis, ringraziare il mio relatore **Prof. Ing. Paolo Bellavista** ed il mio correlatore **Dott. Alessandro Proscia** per la disponibilità e la cortesia avute nei miei confronti: particolarmente preziose sono risultate le loro indicazioni con le quali sono stato costantemente guidato nell'elaborazione di questa tesi.

Ringrazio **l'azienda** Imola Informatica per avermi fornito tutti gli strumenti di cui avevo bisogno.

Ho dedicato questa tesi **ai miei genitori:** sono contento di poter dar loro questa gioia odierna, perché io provo assiduamente, non sempre riuscendovi purtroppo, a essere il figlio perfetto che loro meriterebbero.

Ringrazio la mia **famiglia**, mia madre, mio padre, mio fratello e mia sorella poiché sono stati il mio punto di riferimento, perché hanno avuto la pazienza di sostenermi sempre e comunque e per la fiducia che mi hanno dimostrato; se mai nulla mi è mancato fino ad ora è stato solo grazie a loro.

Grazie anche **ai nonni e agli zii**, quelli che ci sono e quelli che non ci sono più, tutti hanno sempre fatto quadrato intorno a me, insegnandomi, ognuno a proprio modo, il bello della vita. C'è chi aspetta con ansia la mia laurea e si agita tra mille preparativi e chi, invece, assisterà a questa ma da una "platea migliore".

Ringrazio la mia **dolce metà** Maria Laura, per i suoi preziosi consigli, per avermi sempre compreso e per essermi stata sempre accanto soprattutto nei momenti di tensione.

Un grosso ringraziamento va, poi, a tutti **gli amici,** in particolar modo a Massimiliano e Tommaso, per avermi sempre supportato durante gli anni di studio, aiutandomi nei momenti difficili e creando sempre nuove occasioni per ridere e stare insieme.

Grazie a tutti per essermi stati vicini.

Sommario

1. Introduzione

Parametro indispensabile di valutazione di un qualsiasi prodotto o servizio, ai giorni nostri, è la web reputation. Sono sempre più numerose le aziende che monitorano la propria "reputazione online". Quest'ultima può esser definita come l'insieme dei messaggi, commenti e feedbacks, positivi, neutri o negativi che siano, di utenti che esprimono la loro opinione tramite il web su un determinato servizio o prodotto rivolto al pubblico.

E' proprio in questo quadro che va a delinearsi l'interesse incalzante ed emergente di analizzare i dati della rete, per sfruttarli a vantaggio del marketing aziendale su grande scala. Un commento negativo e poco entusiasta può ledere l'immagine professionale e rinomata di cui gode l'azienda. Quindi è necessario tenere sottocontrollo la propria web reputation, affinché l'utente non venga condizionato dai commenti negativi. Il monitoraggio si concretizza in un'analisi empirica dei feedbacks dei clienti, aiutando le imprese produttrici e fornitrici al miglioramento delle proprie attività.

La raccolta e l'elaborazione di tutti questi commenti consentirà la previsione dell'andamento di mercato del prodotto; consentirà, inoltre, un intervento mirato al raggiugimento del successo aziendale. I risvolti statistici avranno anche un'utilità di applicazione nel campo pubblicitario.

L’applicazione sviluppata, svolta nell’azienda Imola Informatica con la collaborazione di TEIA Technologies, si pone l’obiettivo di analizzare in tempo reale tramite l’utilizzo di Apache Storm, dati provenienti da fonti eterogenee, classificarli tramite KNIME utilizzando tecniche di classificazione quali SVM, alberi decisionali e Naive Bayesian, renderli persistenti mediante l’utilizzo del database NoSQL HBASE e di visualizzarli in tempo reale attraverso dei grafici

utilizzando delle servlet, al fine di costituire un valido strumento di supporto per i processi decisionali.

Il presente elaborato introduce nel capitolo 1 i concetti fondamentali dei Big Data. Successivamente vengono presentati e confrontati nel capitolo 2 framework quali Storm, Spark e Streams per il processamento di dati in real time soffermandosi sugli strumenti utilizzati. Nel capitolo 3 vengono confrontati e categorizzati database NoSQL quali HBase, MongoDB e Cassandra riportando per ognuno di essi gli scenari migliori di utilizzo. Nel capitolo 4 viene presentata l'architettura, il funzionamento e viene mostrato l'output dell'applicazione. Nel capitolo 5 vengono valutate le performance degli strumenti utilizzati prima e dopo aver fatto il tuning dei parametri di deploy e vengono messi a confronto i diversi algoritmi di classificazione utilizzati. Infine troviamo le conclusioni di tale lavoro ed i possibili sviluppi futuri.

1. Panoramica sui BigData

"What is Big Data? A meme and a marketing term, for sure, but also shorthand for advancing trends in technology that open the door to a new approach to understanding the world and making decisions ."

Nel corso degli ultimi anni l'evoluzione tecnologica ha portato ad un notevole incremento dei dispositivi in grado di generare informazione di natura molto diversa. La maggior parte di questi dati non presentano una struttura predefinita e quindi non sono facilmente riconducibili ad una struttura tabellare. All'interno di un'azienda possiamo trovare numerosi esempi di dati non strutturati: documenti di vario genere, immagini in vari formati, email, forum interni. Le soluzioni che in genere utilizzano le aziende per gestire ricerche, anche complesse, su tali dati, sono normalmente più che sufficienti. Tuttavia, se si volesse estendere l'utilizzo di tali soluzioni per ricerche e classificazioni di contenuti all'esterno dell'azienda, si riscontrerebbe qualche problema, dovuto alla grande mole di dati a cui potenzialmente si dovrebbe attingere. Le tecnologie tradizionali, trattandosi di dati destrutturati, non sarebbero in grado di analizzare dati di vario genere, dati pubblicati sui diversi Social Network e Pagine Web. Di pari passo all'esplosione delle reti sociali e all'affermarsi degli smartphone, i Big Data si aggiudicano il primo posto come termine più ricorrente negli ultimi anni nel mondo dell'innovazione, del marketing e dell'informatica. La figura 1.0 riporta un grafico prodotto con Google Trend, che mostra l'andamento delle ricerche del termine "Big Data"

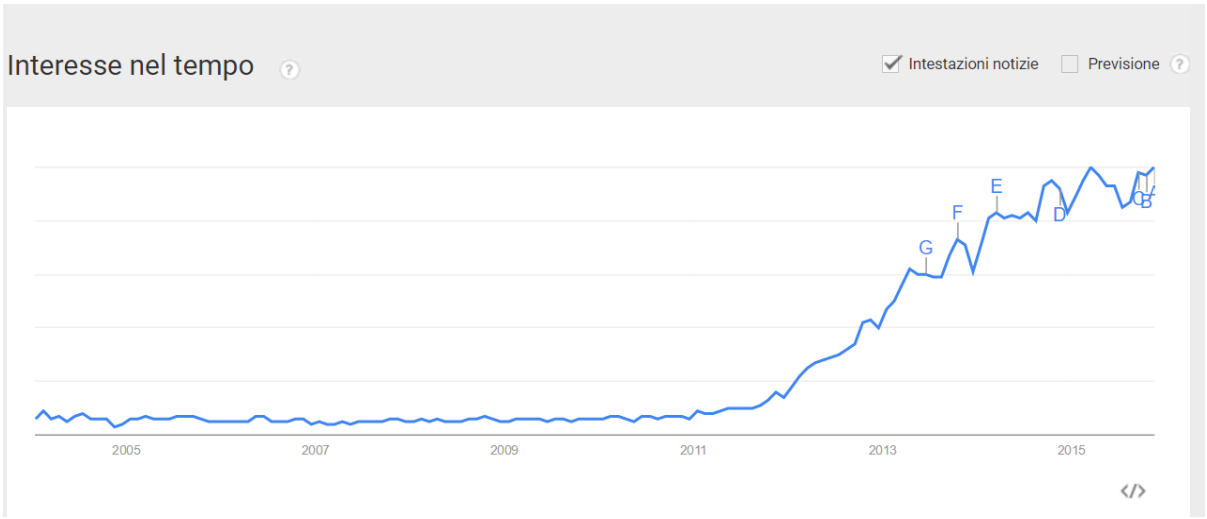


Figura 1.0: Andamento delle ricerche del termine “Big Data” con Google Scholar

1.1 Caratteristiche dei BigData

Con il termine Big Data indichiamo dati prodotti in grande quantità, con notevole rapidità e nei formati più diversi, la cui elaborazione richiede tecnologie e risorse che vanno ben al di là dei sistemi tradizionali di gestione e immagazzinamento dei dati. Molti analisti, per poter definire il concetto di Big Data, utilizzano le quattro caratteristiche descritte di seguito:

Volume;

Velocità;

Varietà;

Veridicità;

In aggiunta a queste caratteristiche proposte da IBM[3], è possibile aggiungere un ulteriore caratteristica[4]:

Valore;

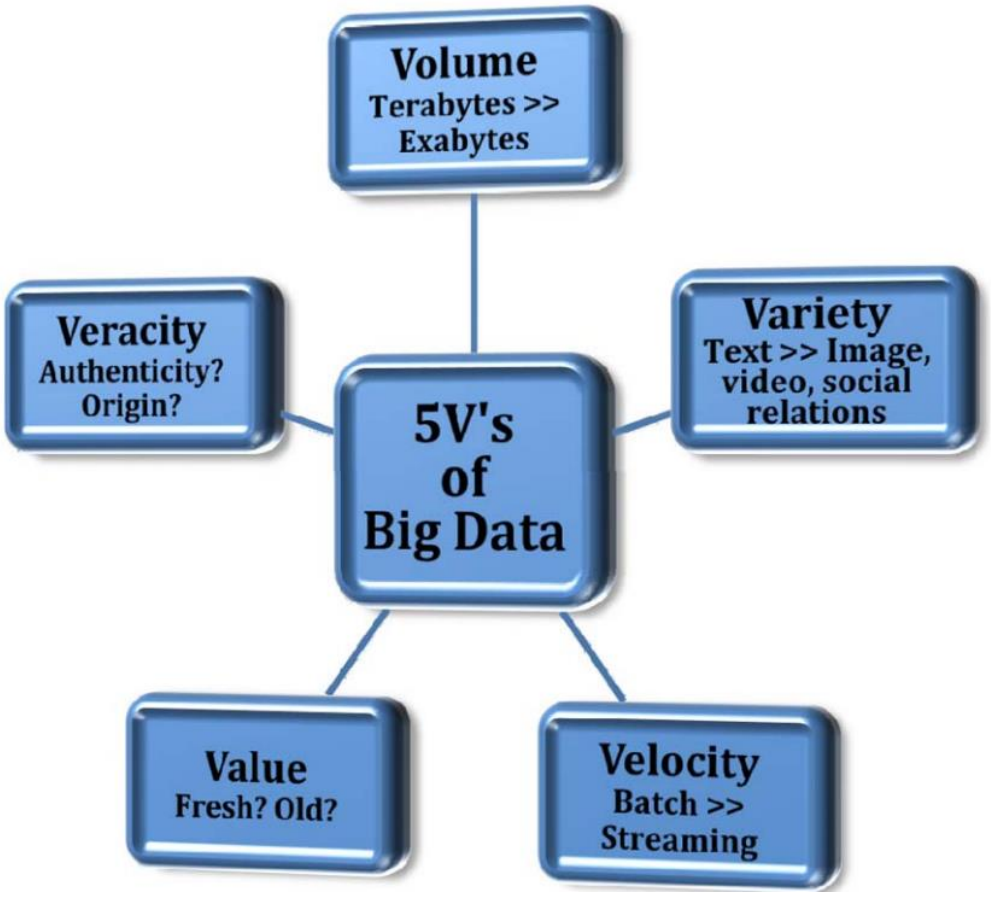


Figura 1.1: Le 5 V dei Big Data

Esaminiamo brevemente ciò che ognuno di loro rappresenta.

1.1.1 Volume

Uno degli aspetti che caratterizzano i big data è la loro quantità.

Il Volume si riferisce alla capacità di acquisire, memorizzare ed accedere a grandi volumi di dati. Si è riscontrato che negli ultimi due anni è stato generato il 90% dei dati di tutto il mondo.

IDC(International Data Corporation) stima che, entro il 2020, l'insieme di tutti i dati digitali creati e consumati in un anno sarà pari a 40 zettabyte:solo Twitter sembra generare più di 7 terabytes di dati ogni giorno, mentre Facebook 10 Terabytes.

Senza entrare nel dettaglio, a seguito di un'attenta valutazione delle alternative disponibili, la scelta di Facebook è ricaduta su Hadoop, dando vita al più grande cluster con questa tecnologia. Google ha sviluppato invece un proprio File System e un proprio Data Base per far fronte alle quantità elevate di dati provenienti dall'indicizzazione del Web.

Alcune tipologie di Big Data sono transitorie, come ad esempio i dati generati da sensori, che verrebbero persi se non immediatamente salvati(es. log dei web server, pagine web che possono essere rimosse, etc).

Non tutti sanno, ad esempio, che un motore di un aereo, genera circa 10 TB di dati ogni trenta minuti di volo, e poiché nelle tratte nazionali vi sono due motori significa che un volo Genova-Catania genera 60 TB di dati, mentre un volo Milano New York, con un quadrimotore ben 640 TB. E' chiaro dunque che uno degli aspetti fondamentali per operare con i Big Data è quello dell'immagazzinamento di tutti i dati originali, indipendentemente dal loro immediato utilizzo. Le operazioni di pulizia potrebbero comportare perdita di informazione utile in futuro. E' evidente come adottando questo modo di procedere, l'ammontare di dati da mantenere nei sistemi aziendali diventi estremamente elevato.[1]

In alcuni casi e posto di poter investire cifre elevatissime per l'acquisto sia di dispositivi di archiviazione e sia di sistemi aventi capacità di calcolo necessaria per elaborare i dati, l'immagazzinamento di dati potrebbe avvenire in un normale RDBMS. Tuttavia, le performance non sono all'altezza dell'investimento effettuato.

1.1.2 Varietà

Solo per fare alcuni esempi, proviamo ad elencare quali potrebbero essere le principali fonti dati per la Big Data Analytics:

Dati strutturati in tabelle(relazionali) che restano il modello di dati preferenziale per le principali piattaforme analytics.

Dati semistrutturati(XML e standard simili)

Dati di eventi e macchinari(messaggi batch o real time, sensori, RFID e periferiche) che sino a pochi anni fa venivano memorizzati solo con profondità temporali molto brevi per problemi di storage

Dati non strutturati(linguaggio naturale, audio, video)

Dati non strutturati da social media

Dati dalla navigazione web

Dati Gis(GeoSpatial,GPS)

Dati scientifici(astronomici, genetica, fisica)

Questa elencazione mostra quale sia la potenziale varietà di dati da trattare in un'applicazione sviluppata per trasformare i dati in informazioni di business

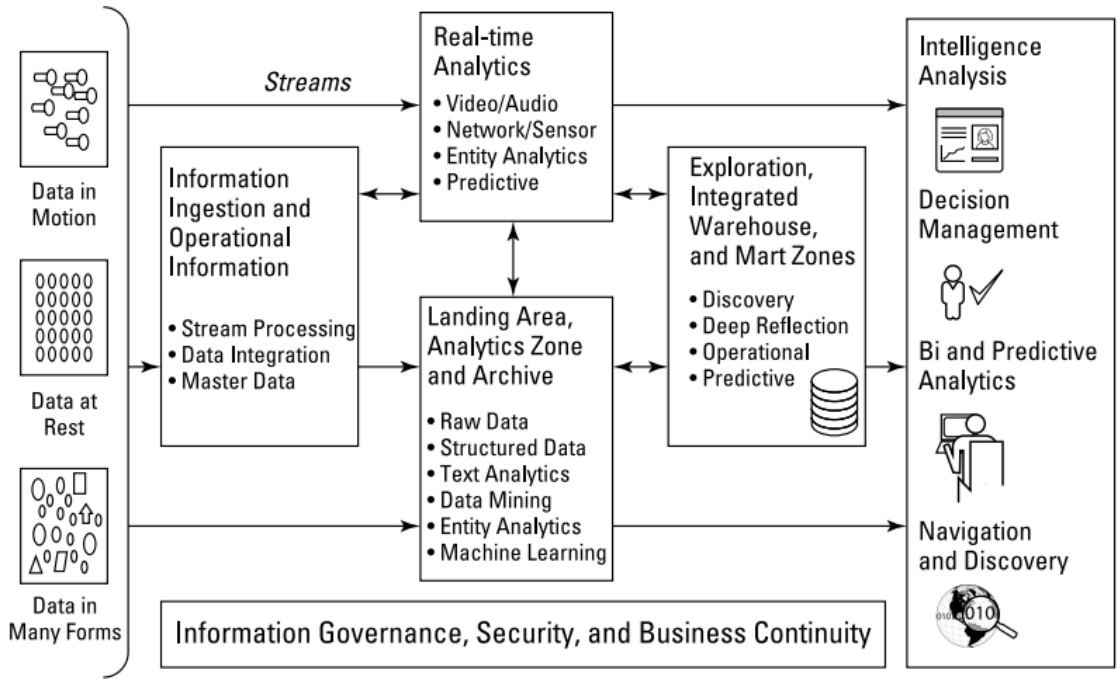


Figura 1.2: Eterogeneità ed applicazioni dei dati

L'eterogeneità di formati,strutture e fonti sono la seconda caratteristica dei Big Data. Per il salvataggio di dati semistrutturati, la scelta ricade spesso su

database NoSql, che forniscono i meccanismi adatti a organizzare i dati ma non impongono una rigidità dello schema. Proprio per questo motivo tali database vengono chiamati schemaless database in quanto privi di uno schema, a differenza degli RDBMS dove lo schema deve essere disegnato prima dello sviluppo di codice applicativo, permettono di adattarsi alla variabilità dei dati.

1.1.3 Velocità

Diversamente da quanto si potrebbe immaginare, il termine velocità in questo caso non fa riferimento alla crescita, ma alla velocità con cui nuovi dati si rendono disponibili.

La velocità con cui le fonti generano nuovi dati rende necessario l'utilizzo di strumenti in grado di tenerne il passo. L'esigenza di ottenere tempi rapidi di risposta ha portato allo sviluppo di database chiave/valore e database colonnari appartenenti alla famiglia dei database NoSql che offrono ottime performance in termini di scrittura/lettura in real time.

1.1.4 Veridicità

Tutti i dati raccolti costituiscono un *valore* per un'azienda. Più dati si hanno a disposizione più informazioni e valore si riescono ad estrarre anche se il solo volume dei dati non garantisce una buona qualità: da dati poco accurati non è possibile supportare processi decisionali in maniera proficua. La *veridicità* e la qualità dei dati diventa pertanto un requisito fondamentale affinché i dati possano davvero creare nuove idee e costituire valore.

1.1.5 Valore

Si traduce nella necessità di cooperazione interdisciplinare. E' proporzionale alla veridicità e insieme sollevano le problematiche più difficili per l'uso industriale dei big data (è da notare che in uno studio recente di Deloitte la

viability è nominata come la quinta V, e la somma delle cinque V dà il "valore" (valore). Come appuriamo che i dati siano utili, attendibili e accurati da un'enorme varietà di data sets online? E una volta che lo abbiamo fatto, come ne estraiamo il valore? Inoltre, i big data costituiscono una questione interdisciplinare che richiede la cooperazione dell'accademia, dell'industria e dell'impresa.

Oggi, la maggior parte degli approcci ai big data contiene inevitabilmente una differenza tra l'ideale e l'attuale. Metodi e algoritmi che si incentrano sugli aspetti dei data base come le analisi statistiche nel business, nel management e nella biomedicina, datadriven process monitoring, pronostici e sistemi di controllo e ottimizzazione, sono stati ampiamente analizzati negli ultimi anni, ma sono ancora sulla linea di partenza. Molte imprese, comunque, considerano i loro big data come delle informazioni riservate, ostacolando dunque lo sviluppo di nuovi approcci da parte della ricerca accademica. Quindi, le imprese dovrebbero prendere in considerazione una più ampia cooperazione con i ricercatori e gli ingegneri, fornendo dei loro dati pubblici per gli sviluppi delle tecniche esistenti e per la promozione di nuove idee. Acquisire, attraverso un'appropriata analisi, un appropriato utilizzo e gestione, il più alto valore dai big data disponibili, dovrebbe essere la più alta aspirazione per le industrie moderne.

1.2 Potenziale economico e di impatto dei Big Data

Perché un’azienda dovrebbe analizzare i post, i tweet sui social network? Perché dovrebbe essere interessata ad analizzare dati provenienti dalle più diverse fonti investendo cifre spesso elevate?

Blog, tweet e commenti sui social network permettono alle persone di esprimere le proprio idee e percezioni sui prodotti e servizi. E’ dall’analisi di

questi dati che si colgono le opportunità e si trae supporto per i processi decisionali in modo tale da avere una grande incidenza sull'attività aziendale.
Attraverso la gestione e l'analisi di dati provenienti dall'esterno le grandi società possono migliorare degli aspetti importanti del proprio modo di agire grazie alla conoscenza delle opinioni degli utenti: possono trarre vantaggio capendo le esigenze del cliente cercando di essere i primi a soddisfare un fabbisogno sulla concorrenza, possono capire in tempo reale che impatto ha il lancio di un nuovo prodotto sul mercato, possono analizzare *tout court* i comportamenti dei clienti a 360°[13].
In ambito marketing, l'uso dei Big Data è famigliare nella costruzione dei così detti metodi di raccomandazione, come quelli utilizzati da ImdB e Amazon per proporre prodotti sulla base degli interessi di un cliente rispetto a quelli di milioni di altri. Tutti i dati provenienti dalla navigazione di un utente, dai prodotti valutati o ricercati, dai suoi precedenti acquisti permettono ai colossi del commercio di suggerire i prodotti più adatti agli scopi del cliente,spingendolo a comprare per necessità o per semplice impulso. Ci sono notevoli esempi di aziende in tutto il mondo che sono bene conosciute per il vasto ed efficace utilizzo dei dati. Di seguito vengono menzioniate brevemente le best practices delle aziende mondiali che hanno sfruttato il valore dei Big Data:
Il programma di fidelizzazione di Tesco genera un ammontare di dati sui clienti, da cuil'azienda estrae informazioni, per costruire campagne promozionali sino ad arrivare alle strategie di segmentazione della clientela.
Harrah's ha aumentato il ricavo dell'8-10% analizzando i dati sulla segmentazione dei clienti

Fedex, la società di trasporto espresso più grande al mondo, ha ottenuto in tempo realei dati su spedizioni e consumatori provenienti da più di 46000 centri di distribuzione ensupply chain.

Secondo un sondaggio di Gartner, il 64% delle organizzazioni globali sta investendo, o ha intenzione di investire, in tecnologie per i Big Data contro il 54% che già lo faceva o pensava di farlo nel 2011. È anche vero che solo l'8% delle imprese ha implementato tali tecnologie.

Non solo le imprese, ma anche i governi, le agenzie governative e i servizi di pubblica sicurezza possono trarre valore dai big data. I servizi di intelligence analizzano i big data monitorando i contenuti, in qualsiasi lingua, di siti web e social network, per individuare possibili minacce alla sicurezza del paese.

Per approfondire gli utilizzi già in atto nel settore pubblico, possiamo far riferimento al documento *Fact Sheet: Big Data Across the Federal Government [5]*, fornito dal governo americano, dove possiamo vedere che gli investimenti del Dipartimento della Difesa per progetti realizzati nell'ambito di big data ammonta a 250 milioni di dollari. Tra l'elenco possiamo trovare progetti per:

- l'identificazione di minacce alla sicurezza, basato sull'analisi di attività in rete, per Sistemi di lettura del testo e salvataggio delle informazioni attraverso una rappresentazione semantica
- Ricerca biologica e ambientale
- Controllo e prevenzione di malattie
- Ricerca sul cancro
- Progetti della NASA
- Tecniche di calcolo su dataset di enormi dimensioni

Molti altri possono essere i vantaggi relativi all'analisi dei Big Data:

Nel 2009 un team di ricerca di Google annunciò un risultato notevole sulla rivista Nature; il team sosteneva di essere in grado di tracciare, e monitorare i casi di influenza negli Stati Uniti puntando sulle ricerche effettuate sul motore, relative alla patologia, con un solo giorno di ritardo rispetto alla settimana o più che occorreva al Centro per il Controllo e la Prevenzione delle Malattie per assemblare un grafico costruito sulla base delle relazioni degli ambulatori medici.

Nel 2012 le elezioni americane hanno reso palpabile la potenza dei Big Data. Obama ha vinto facendo leva su quella forza per capire gli umori degli elettori cercando di reagire,quasi in tempo reale, a cambiamenti del sentiment, trascinando gli indecisi alle urne.

1.3 Ciclo di vita dei Big Data

Di seguito sono riportate e descritte alcune fasi importanti del ciclo di vita dei big data:

- acquisizione
- persistenza
- integrazione
- analisi.

L’acquisizione dei Big Data può avvenire mediante diversi mezzi[2]:

- API messe a disposizione dalle fonti dati tramite le quali è possibile interfacciarsi. Due particolari esempi sono le Twitter API e le Graph API di Facebook che consentono, dopo aver eseguito una fase di autenticazione ed aver acquisito un Token, di esaminare tutti i tweet pubblici che rispettano determinati criteri nel primo caso e nel secondo caso di tutti i contenuti pubblic o accessibili tramite amicizia che rispondono ai criteri di ricerca desiderati.

- import di dati con strumenti ETL : i dati vengono estratti da sistemi sorgenti quali database transazionali, comuni file di testo o da altri sistemi informatici e subiscono un processo di trasformazione che ha l'obbiettivo di consolidare i dati che vengono successivamente memorizzati nelle tabelle del sistema di sintesi. Molti strumenti ETL sono già attrezzati per connettersi a sistemi quale Hadoop.
- Lettura di Stream di dati: La velocità con la quale si producono dati ha reso indispensabile tecnologie per il trasferimento dei dati in modo tale che essi siano processati in un flusso continuo. Le API di Twitter offrono un insieme di funzionalità a bassa latenza utili ad accedere in tempo reale al flusso globale di tweet, eliminando perciò la necessità di chiamare ripetutamente ad intervalli regolari il relativo end-point REST

  La persistenza dei dati, a causa della loro eterogeneità, è un punto di difficile risoluzione con le tecnologie tradizionali. Esistono diversi database per gestire la persistenza di dati eterogenei tra cui HBase e Cassandra che consentono di ottenere tempi di lettura e scrittura molto veloci anche su tabelle aventi molti miliardi di righe e colonne.

  L'integrazione con i dati presenti in database esterni è una fase che spesso risulta necessaria per preparare i dati alla fase di analisi.

  Infine la fase di analisi dei dati può avvenire con diversi strumenti ed in diverse modalità(batch,real-time), nel mondo di Hadoop è possibile utilizzare Hive, che consente di eseguire query ed analizzare grandi dataset in batch, al di fuori di Hadoop è possibile utilizzare strumenti quali Pig, R, Mahout o Knime.

1.4 L'emergere di nuove problematiche

Nonostante l'avvento dei Big Data, assuma un ruolo fondamentale nell'economia attuale, ha aperto dei dibattiti inerenti questioni etiche, morali e

sociali: sono sempre di più le persone che vedono questa raccolta di informazioni con profondo sospetto, come un'intrusione della loro privacy e nient'altro più[6]. Seppur l'evidenza, andrebbe forse spiegato in maniera più chiara che i dati in grande quantità possono avere un ruolo economico significativo a beneficio non solo del commercio privato, ma anche delle economie nazionali.

1.5 I Big Data in Italia

Una ricerca della SDA Bocconi School of Management realizzata in collaborazione con IBM, descrive la situazione relativa ai Big data in Italia. Tale ricerca è stata effettuata su un campione di 202 imprese appartenenti a diversi settori quali:

- Manifatturiero(31%)
- Pubblica Amministrazione locale e della Sanità(18%)
- Distribuzione e della logistica(18%)
- Finanza(11%)
- Servizi e utility(22%)

Dai risultati si evincono i seguenti punti:

Le fonti dei Big Data percepite come più significative sono social network e social media.

Le aziende mostrano maggiore interesse in ambiti quali l'analisi di prodotti e servizi più innovativi e l'analisi più accurata delle esigenze dei propri clienti.

Diversi sono i punti critici da valutare per l'adozione di una soluzione big data in un'azienda tra cui il budget disponibile, la valutazione del ROI dell'investimento, la presenza di competenze. La maggior parte delle aziende esprime un interesse per la problematica ma circa il 10% di esse ha in programma o ha già realizzato una soluzione basata sui big data. Quello che

ostacola la realizzazione basata sui Big Data è principalmente la mancanza di skill.

2. APACHE STORM

2.1 Panoramica di Storm

Storm è un sistema di computazione real-time, distribuito, free e Open Source attualmente incubato dall'Apache Software Foundation[7]

La potenza di Storm sta nel fatto che esso è in grado di risolvere un'enorme varietà di casi d'uso con un semplice insieme di primitive. Un benchmark ha rivelato che ogni nodo del sistema riesce a processare un milione di tuple al secondo. E' sulla base di queste peculiarità che molte, importanti società del settore IT hanno hanno integrato nei propri sistemi quello che si può definire uno Storm Cluster: Twitter, Groupon, Yahoo!, Flipboard, Alibaba sono solo 5 delle 55 società che ad oggi utilizzano Storm per effettuare analisi in real-time.[8]Storm è stato sviluppato dalla BackType, un'azienda che forniva supporto ad altre aziende tramite analisi social, per effettuare scelte di Marketing. Successivamente all'acquisto di tale azienda da Twitter, Storm è diventato il cuore di Twitter. Storm può essere utilizzato per i seguenti scenari:

- Stream Processing: Storm è usato per processare uno stream di dati e aggiornare diversi database in real time. La velocità di processamento del dato deve essere almeno pari alla velocità con cui arrivano i dati in input.
- Continuous Computation: interrogazioni continue e presentazione dei risultati in real-time. Un esempio è l'analisi dei trend dei topic su Twitter.
- Distributed RPC: Storm può essere utilizzato per parallelizzare un'intensa interrogazione *on the fly.*
- Real-time analytics: Analizzare e visualizzare dati che provengono da diversi data sources in real time.

2.2 Caratteristiche di Storm

Le peculiarità di Storm che lo rendono una soluzione perfetta per analizzare stream di dati in real time sono le seguenti:

Velocità: Un benchmark ha rivelato che ogni nodo del sistema riesce a processare un milione di tuple al secondo.

Orizzontalmente scalabile: Ogni nodo (WN o Worker Node) ha un limite di numeri di eventi che può processare in un secondo. Un nodo rappresenta una macchina, correttamente configurata, che esegue Storm. Essendo una piattaforma distribuita, Storm permette di aggiungere nodi al cluster e dunque di incrementare la capacità di processamento. E' linearmente scalabile il che significa che è possibile raddoppiare la capacità di processamento raddoppiando i nodi.

Fault Tolerant: Le unità di lavoro sono eseguite da processi worker(WP o Worker Process) in un cluster Storm. Quando un processo muore, Storm si occupa di riavviarlo. Se il nodo sul quale il processo worker eseguiva muore, Storm si occupa di riavviarlo su un altro nodo del cluster.

Affidabilità: Storm fornisce un meccanismo che permette di processare ogni messaggio una sola volta. Se il processamento di un messaggio fallisce, Storm, previa configurazione, si occuperà di processare di nuovo il messaggio.

Supporto a diversi linguaggi di programmazione: Sebbene la piattaforma di Storm viene eseguita su una Java Virtual Machine, le applicazioni possono essere scritte in diversi linguaggi di programmazione.

2.3 Componenti di Storm

Un cluster Storm segue un modello master-slave, dove le diverse comunicazioni vengono coordinate attraverso ZooKeeper.[9]

Il master in un cluster Storm è il nodo Nimbus. Esso è responsabile della distribuzione del codice applicativo sui vari nodi worker, di assegnare task sui diversi nodi, di monitorare task per vedere si vi sono dei fallimenti, e riavvia i task quando è necessario. Il nodo Nimbus è Stateless e salva tutti i suoi dati in ZooKeeper. Per un cluster Storm, esiste solo un nodo Nimbus. In caso di failure può essere riavviato senza avere alcun effetto sui task in esecuzione sui nodi worker(a differenza di Hadoop che quando il JobTracker muore, tutti i task in esecuzione vengono lasciati in uno stato inconsistente e necessitano dunque di una nuova esecuzione).

Un altro componente di Storm è il nodo Supervisor, questi nodi sono i worker in un cluster Storm. Ogni nodo supervisor esegue un supervisor daemon che è responsabile della creazione, avvio e stop dei processi worker che vengono eseguiti in un nodo. Il supervisor daemon, come Nimbus, sono stateless e salvano tutto il loro stato in Zookeeper(in questo modo possono essere riavviati senza problemi di inconsistenza). Normalmente, un supervisor daemon gestisce più processi worker su una macchina (Vedi Figura 2.1).

In molte applicazioni distribuite, diversi processi hanno bisogno di coordinarsi tra di loro e condividere delle informazioni di configurazione. Zookeeper è un'applicazione che fornisce tutti questi servizi in modo affidabile, anche Storm usa un cluster ZooKeeper per coordinare vari processi. Lo stato del cluster e i vari task sottomessi a Storm sono salvati in Zookeeper, il nodo Nimbus e i nodi supervisor non comunicano direttamente tra di loro, ma comunicano attraverso Zookeeper. Anche esso è fault tolerant, questo significa che può essere killato senza compromettere il cluster.[10] Di seguito vi è un diagramma relativo ad un cluster Storm.

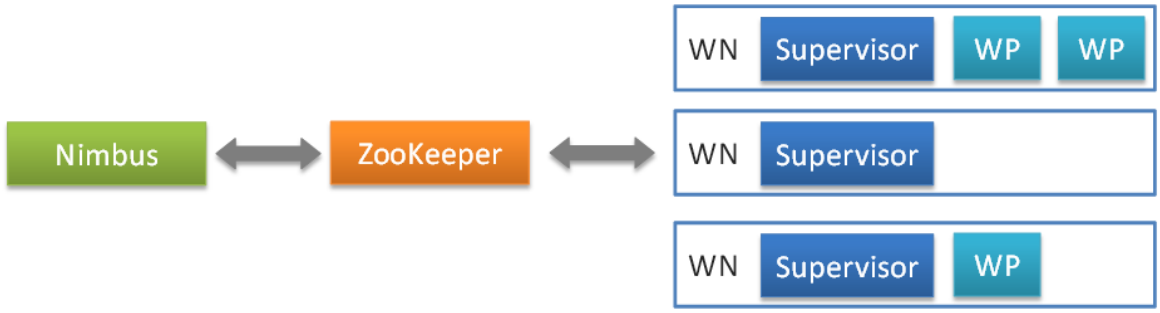


Figura 2.1: Architettura cluster Storm

2.4 Definizione di una Topologia e suoi Componenti

Una topologia può essere rappresentata da un grafo aciclico, dove ogni nodo fa delle operazioni e invia i risultati ad un altro nodo. Un esempio di Topologia è la seguente:

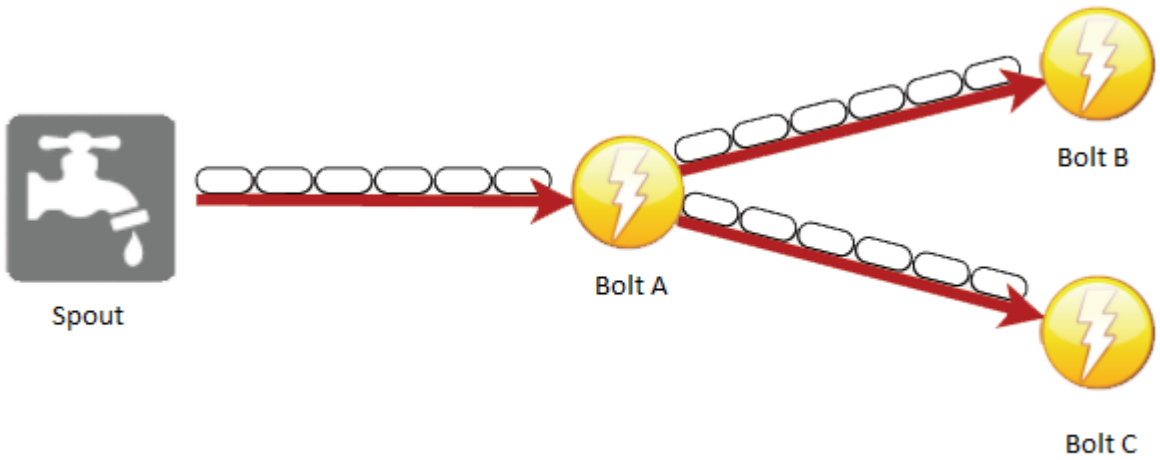


Figura 2.2: Esempio di topologia

I componenti di una topologia sono:

Stream: una sequenza infinita di tuple che possono essere processate in parallelo da Storm. Ogni stream può essere processato da uno o più Bolt. Una

tupla è l'unità base di dati che può essere processata da Storm, ognuna ha una lista predefinita di etichette che possono assumere diversi valori (tipi primitivi o classi Serializzabili).

Spout: la sorgente di tuple in una topologia. Esso è responsabile della lettura dei dati e di incapsularli in tuple da inviare ai Bolt. Se si sviluppi in Java si può implementare l'interfaccia IRichSpout. Quando una tupla viene emessa ad un Bolt, Storm ne tiene traccia per tutto il suo percorso, inviando allo Spout un ACK. Questo avviene solo se la tupla viene emessa insieme ad un MessageID, in caso contrario, se il MessageID è null, il comportamento di default è quello di non tenerne traccia. E' possibile definire, sempre inserendo un MessageID insieme alla tupla,anche un timeout, se la tupla non viene precessata con lo specifico timeout viene inviato un fail allo Spout. In termini di performance, considerando lo scenario in cui è ammessa qualche perdita di tupla, vi è un piccolo guadagno non utilizzando questo livello di affidabilità (Lasciando il MessageId null). I metodi principali di uno Spout sono il nextTuple() che è il metodo che chiama Storm per ricevere la tupla successiva. In questo metodo bisogna implementare la logica di lettura dai datasources, di incapsulare la tupla e di inviarla. Il metodo ack() che viene richiamato da Storm quando una tupla che è stata emessa con il MessageId è stata processata correttamente. Il metodo fail() richiamato quando una tupla non è stata processata nel giusto tempo o quando non è stata processata correttamente. Il metodo open() che viene richiamato una sola volta in fase di inizializzazione dello spout.

Bolt: si occupa di fare delle operazioni sulle tuple che gli arrivano in ingresso da uno Spout o da un altro Bolt. Idealmente ogni Bolt dovrebbe fare delle operazioni semplici su una tupla e, per trasformazioni più complesse, coordinarsi con altri Bolt. Se si sviluppa in Java è possibile implementare l'interfaccia IRichBolt. Un Bolt può sottoscriversi a più Stream di dati

appartenenti a diversi componenti. I metodi principali di un Bolt sono l'execute(Tuple input) che viene eseguito per ogni tupla proveniente dallo stream al quale si è sottoscritto, in questo metodo vengono fatte operazioni sulla tupla, persistenza della tupla in un database o invio della tupla elaborata ad altri Bolt. Se un MessageId è stato associato ad una tupla, il metodo execute pubblica un ack o un fail per permettere a Storm di sapere se quella tupla è stata processata correttamente.

Storm supporta due modalità di esecuzione di una topologia: Modalità locale, Modalità remota. Nella modalità locale, la topologia viene eseguita sulla macchina locale in una singola JVM. Questa modalità simula un cluster Storm in una singola JVM ed in genere viene utilizzata per effettuare testing e debugging di una topologia. Nella modalità remota, viene utilizzato un client Storm per sottomettere la topologia al master. Il nodo Nimbus si prenderà cura di distribuire il codice su tutti gli altri nodi.

2.5 Partizionamento delle tuple

Quando viene definita una topologia, uno stream di dati viene diviso in un numero di partizioni e suddiviso tra i diversi Bolt che sono interessati allo stream. Ogni Bolt prenderà un sottoinsieme di tuple dallo stream a cui si è sottoscritto. E' possibile definire come queste partizioni di tuple devono essere assegnate configurandolo nella topologia.

Esistono diversi tipi di Stream Grouping:[11]

Shuffle grouping: Le tuple vengono distribuite in maniera uniforme sui diversi Bolt, sarà processato lo stesso numero di tuple per ogni task.

Field Grouping: Permette di controllare in che modo le tuple sono inviate ai task di un Bolt, basandosi su uno o più campi della tupla stessa. Garantisce che

un dato set di valori per una certa combinazione di campi, sia sempre inviata allo stesso Bolt.

All Grouping Una singola copia di ogni tupla viene inviata a tutte le istanze di un Bolt. Generalmente, questo tipo di grouping è usato per inviare segnali ai Bolt. Ad esempio, se è necessario effettuare un refresh di una cache, è possibile inviare un refresh cache signal a tutte le istanze di un Bolt

Global Grouping: con questo grouping, non vi è nessuna partizione dello stream, ma l'intero stream viene inviato al Bolt avente numero di ID minore

Direct Grouping Con questo grouping, è la sorgente che decide quale componente riceverà la tupla. Può essere utilizzato ad esempio quando abbiamo uno stream di log e vogliamo processare una tipologia di log con un Bolt specifico.

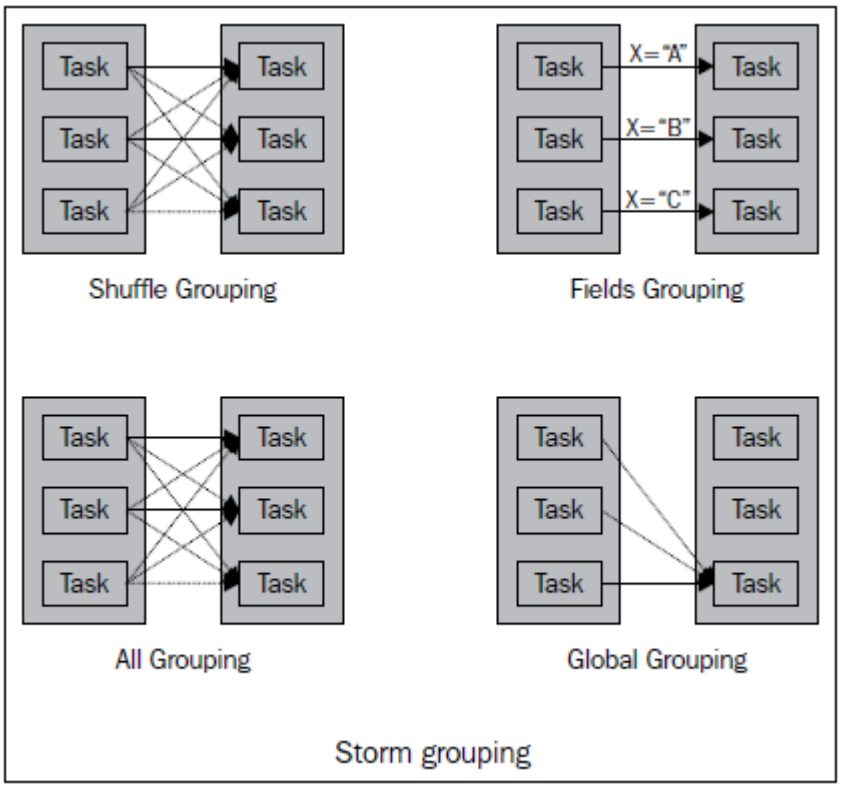


Figura 2.3: Storm grouping

2.6 Word Counter Storm

Per consolidare i concetti visti fino ad ora, in seguito viene riportato un semplice progetto per il conteggio delle parole in un file. Nella figura 2.4 viene mostrata la topologia del progetto.

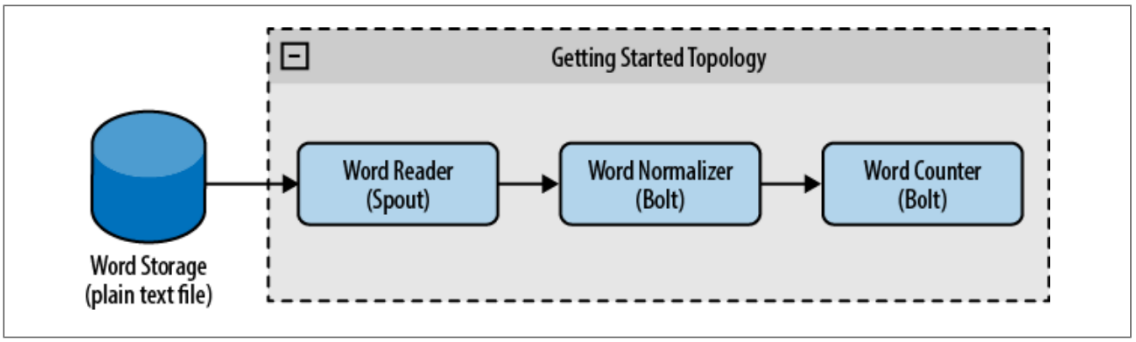


Figura 2.4: Topologia Word Counter

In questa topologia abbiamo uno Spout(Word Reader), che si occupa di leggere un file di testo, un Bolt (Word Normalizer) che si occupa di effettuare lo split di una frase ricevuta dallo Spout in singole parole, ed un Bolt(Word Counter) che si occupa di conteggiare tutte le parole che riceve dal Word Normalizer.

In seguito viene riportato lo Spout del progetto che si occupa di leggere tutte le righe di un file:

```
public class WordReader extends BaseRichSpout {
        private SpoutOutputCollector collector;
        private FileReader fileReader;
        private 27oolean completed = false;
        public void ack(Object msgId) {
                System.out.println(“OK:”+msgId);
        }

//se una tuple fallisce, il framework richiama il seguente metodo, è
//compito dello sviluppatore gestire l’evento.
        Public void fail(Object msgId) {
                System.out.println(“FAIL:”+msgId);
        }
```

```
//Storm richiama il seguente metodo sempre, anche quando il file è stato letto
        public void nextTuple() {
                if(completed){
                        try {
                                Thread.sleep(1000);
                        } catch (InterruptedException e) {
                                //Do nothing
                        }
                        return;
                }
                String str;
                BufferedReader reader = new BufferedReader(fileReader);
                try{
                        //legge tutte le righe del file
                        while((str = reader.readLine()) != null){
// per avere una maggior affidabilità, insieme alla tuple inseriamo un id per attivare gli ACK,
//in questo caso l'id è il messaggio stesso.
                                This.collector.emit(new Values(str),str);
                        }
                }catch(Exception e){
                        throw new RuntimeException("Error reading tuple",e);
                }finally{
                        completed = true;
                }}
//metodo richiamato in fase di inizializzazione dello Spout
        public void open(Map conf, TopologyContext context,
                        SpoutOutputCollector collector) {
        try {
          this.fileReader=newFileReader(conf.get("wordsFile").toString());
                } catch (FileNotFoundException e) {
                throw new RuntimeException("Error reading file     ["+conf.get("wordFile")+"]");
                }
                this.collector = collector;
        }
```

```
        //Questo metodo pone un'etichetta sulla tuple emessa.
        Public void declareOutputFields(OutputFieldsDeclarer declarer) {
                declarer.declare(new Fields("line"));
        }
}
```

In seguito viene riportata l'implementazione del Bolt che si occupa di normalizzare il testo riducendo la frase a lowercase e emettere una tupla per ogni parola contenuta in una frase

```
public class WordNormalizer extends BaseBasicBolt {
        public void execute(Tuple input, BasicOutputCollector collector) {
      String sentence = input.getString(0);
      String[] words = sentence.split("");
      for(String word : words){
        word = word.trim();
        if(!word.isEmpty()){
          word = word.toLowerCase();
          collector.emit(new Values(word));
} }}
        //Questo metodo pone un'etichetta sulla tuple emessa.
        Public void declareOutputFields(OutputFieldsDeclarer declarer) {
                declarer.declare(new Fields("word"));
        }}
```

In seguito è implementato il Bolt che si occupa di contare le occorrenze di ogni parola ricevuta dal WordNormalizzator

```
public class WordCounter extends BaseBasicBolt {
        Integer id;
        String name;
        Map<String, Integer> counters;
//questo metodo viene richiamato quando il cluster verrà spento, per //mostrare il conteggio delle parole
```

```
@Override
public void cleanup() {
        System.out.println("–Word Counter ["+name+"-"+id+"] –");
        for(Map.Entry<String, Integer> entry : counters.entrySet()){
                System.out.println(entry.getKey()+": "+entry.getValue());
        }}
//questo metodo viene richiamato solo in fase di inizializzazione
@Override
public void prepare(Map stormConf, TopologyContext context) {
        this.counters = new HashMap<String, Integer>();
        this.name = context.getThisComponentId();
        this.id = context.getThisTaskId();
}
@Override
public void declareOutputFields(OutputFieldsDeclarer declarer) {}

@Override
public void execute(Tuple input, BasicOutputCollector collector) {
        String str = input.getString(0);
        if(!counters.containsKey(str)){
                counters.put(str, 1);
        }else{
                Integer c = counters.get(str) + 1;
                counters.put(str, c);}}}
```

Infine, in seguito viene mostrata l'implementazione della Topologia

```
public class TopologyMain {
        public static void main(String[] args) throws InterruptedException {

    //Topology definition
                TopologyBuilder builder = new TopologyBuilder();
                builder.setSpout("word-reader",new WordReader());
```

```
//nel modo seguente, agganciamo il Bolt Normalizer alla sorgente. Da notare che nei Bolt e nello
Spout non è stato fatto alcun riferimento sul percorso che le tuple debbano seguire.
                Builder.setBolt(“word-normalizer”, new WordNormalizer())
                        .shuffleGrouping(“word-reader”);
//agganciamo il Word Counter al Word Normalizer.
                Builder.setBolt(“word-counter”, new WordCounter(),1)
                        .fieldsGrouping(“word-normalizer”, new Fields(“word”));

                Config conf = new Config();
                conf.put(“wordsFile”, args[0]);
                conf.setDebug(false);
        //Topologia lanciata in Locale.
                Conf.put(Config.TOPOLOGY_MAX_SPOUT_PENDING, 1);
                LocalCluster cluster = new LocalCluster();
                cluster.submitTopology(“Getting-Started-Toplogie”, conf,
          builder.createTopology());

                /* per lanciare una topologia su un cluster remote
                StormSubmitter cluster = new StormSubmitter();
                cluster.submitTopology(“Getting-Started-Toplogie“, conf,
          b.createTopology());
*/
                Thread.sleep(1000);
                cluster.shutdown();
          }
}
```

2.7 Interfaccia grafica di Storm

L’interfaccia grafica di Storm fornisce informazioni importanti come:
Informazioni sul cluster: è possibile visualizzare la versione di Storm utilizzata dal cluster, il numero di nodi worker disponibili, numero di worker utilizzati etc. Quando lanciamo una topologia su un cluster, dobbiamo assicurarci che vi

siano worker disponibili altrimenti la topologia starà in uno stato di wait fino a quando non vi saranno nuovi worker liberi.

Informazioni sul nodo Nimbus: è possibile visualizzare le configurazioni del nodo nimbus.

Informazioni sui nodi supervisori: è possibile vedere delle informazioni sui nodi supervisori come l'id, l'host, slot liberi e slot utilizzati.

Informazioni sulla topologia: è possibile visualizzare la lista delle topologie che il cluster sta eseguendo con il numero di id, il numero di worker assegnati alla topologia, numero di Bolt etc. Una Topologia può avere i seguenti stati: ACTIVE, INACTIVE, KILLED. E' possibile passare da uno stato all'altro attraverso l'interfaccia grafica. L'interfaccia grafica mostra altre importanti informazioni come il numero emesso di tuple da ogni Spout e Bolt, il numero di ack e di fail riscontrati, la latenza di una tupla ovvero la differenza di tempo tra quando lo spout emette la tupla ed il tempo in cui viene inviato l'ack.

Per default, l'interfaccia grafica di Storm è in esecuzione sulla porta 8080, per potervi accedere bisogna accedere al seguente indirizzo: http://nimbus-node:8080 dove nimbus-node è l'indirizzo IP dell'unica macchina Nimbus presente nel cluster.

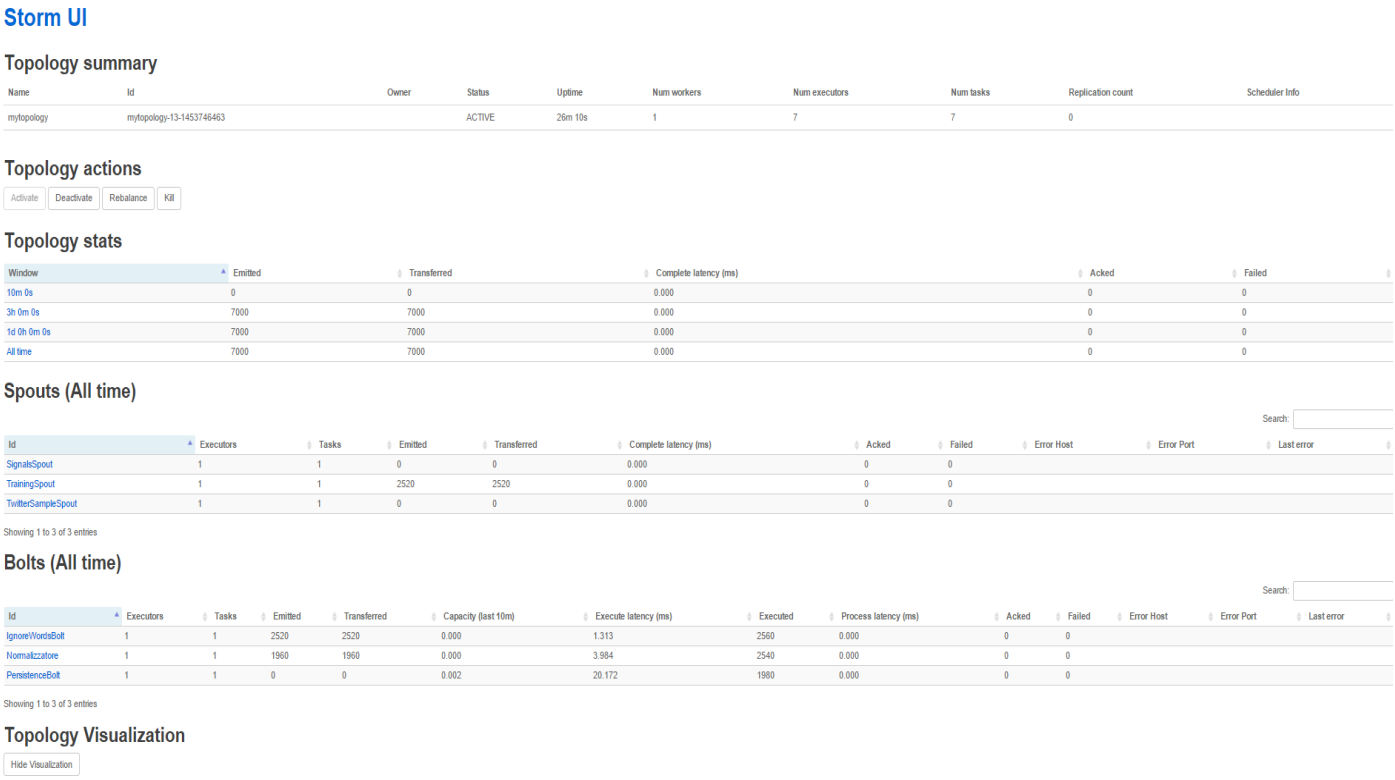


Figura 2.5: Storm UI

2.8 Integrazione di Kafka con Storm

Apache Kafka è un message broker open source, distribuito e ad alte prestazioni[11] sviluppato da Linkedin. Esso può svolgere il ruolo importante di buffer di messaggi che devono essere trattati da Storm. Kafka, integrato con Storm, è emerso come uno dei componenti importanti per l'elaborazione dei dati in tempo reale[10]. Kafka può anche essere utilizzato per inviare i dati uscenti da una topologia ad una coda. L'architettura interna di Kafka differisce in modo sostanziale da altri servizi di messaggistica. Esso è un sistema peer-to-peer dove ogni nodo viene chiamato broker, i diversi broker comunicano tra di loro mediante l'aiuto di ZooKeeper

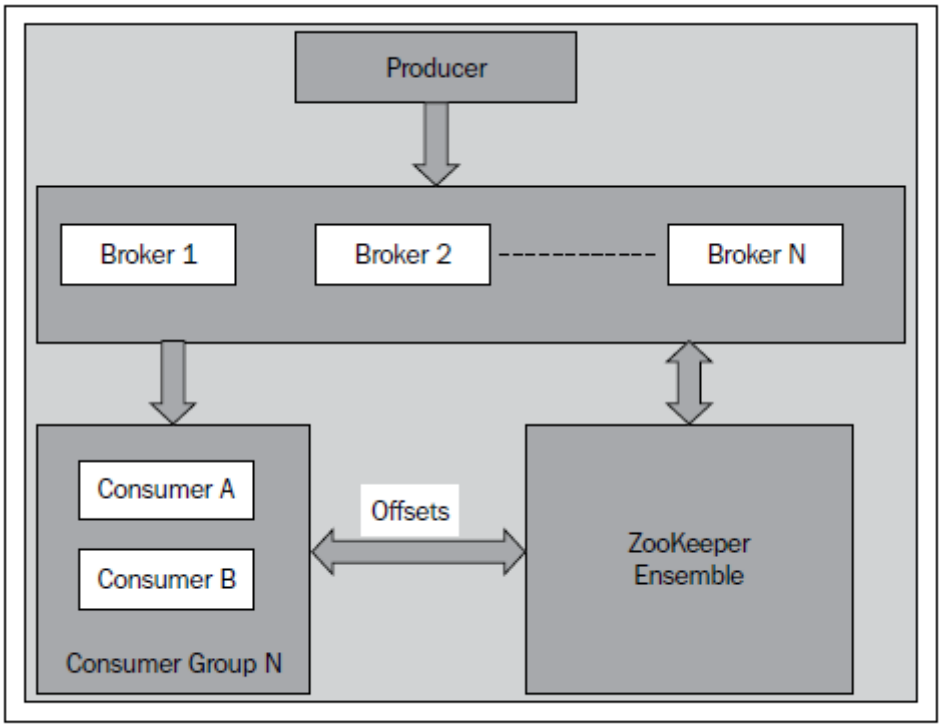


Figura 2.6: Architettura di un cluster Kafka

I componenti principali di Kafka sono i seguenti:

- **Produttore:** invia i messaggi su uno o più Topic(una coda che può essere consumata di più consumatori). Per il parallelismo, un Topic può essere suddiviso in più partizioni e più partizioni possono essere lette o scritte in parallelo. I dati di ogni partizione sono salvati in directory diverse, è possibile salvare directory diverse su dischi diversi per superare i limiti di I/O di un singolo disco. Ogni messaggio nella partizione ha un numero di sequenza univoco chiamato offset.

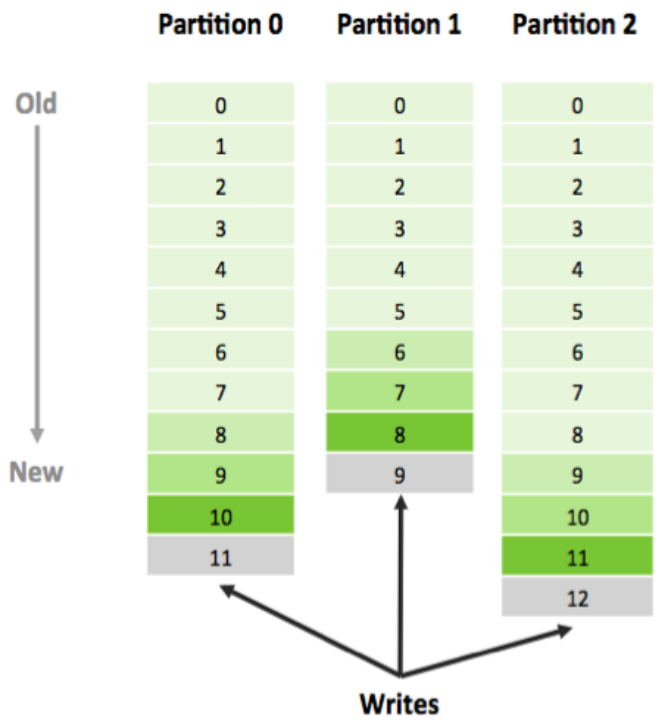


Figura 2.7: Partizioni di Kafka

- **Replicazione:** Kafka supporta la replicazione di una partizione per essere fault tolerance. Automaticamente viene gestita la replicazione di una partizione, cercando di replicarla su broker diversi.
- C**onsumatore:** Un consumatore legge un range di messaggi da un broker. Ogni consumatore ha un groupId, più consumatori con lo stesso groupId, verranno visti dal punto di vista logico come un singolo consumatore. I messaggi non vengono eliminati dal broker una volta consumati, è responsabilità del consumatore tener traccia di quanti messaggi ha consumato.
- **Broker:** I Broker ricevono messaggi dal produttore e si occupano anche della persistenza del messaggio sul disco. Per ogni topic, il broker crea delle directory su disco. Il broker Kafka è molto leggero, esso gestisce solo le partizioni, la loro persistenza e le connessioni TCP.
- **Conservazione dei dati:** E possibile impostare il limite massimo di tempo per un messaggio all'interno di una partizione, allo scadere di questo tempo

il messaggio nella partizione sarà cancellato. Inoltre è possibile settare un limite sulla dimensione che deve avere una partizione prima che i messaggi scaduti siano cancellati dalla partizione.

2.9 Overview sui framework per il processamento dei dati in real time

Oltre ad Apache Storm trattato in precedenza, esistono diversi framework per il processamento di dati in tempo reale[16]. Di seguito vi sono i due, che con Storm, hanno avuto maggior successo.

- Apache Spark
- IBM InfoSphere

2.9.1 Apache Spark

Spark è un framework open-source per l'analisi di grandi quantità di dati su cluster, nato per essere veloce e flessibile. Supporta numerosi linguaggi di programmazione e numerosi concetti come ad esempio MapReduce, processamento di dati in memoria, stream processing, graph processing o machine learning. Spark può essere utilizzato anche on top di Hadoop. Esso è attualmente usato da più di 500 organizzazioni mondiali, tra cui eBay Inc, IBM, NASA, Samsung e Yahoo!. Un sondaggio indica che il 13% dei data scientists usavano Spark nel 2014, con un ulteriore 20% intenzionati ad usarlo nel 2015 e il 31% che lo stavano prendendo in considerazione [14]. Tale framework, date le sue potenzialità e la sua semplicità di utilizzo, sta quindi crescendo esponenzialmente. Con i suoi 465 contributori, è stato il progetto di Big Data open source più attivo del 2013.

Come viene mostrato di seguito, Storm è una buona scelta se si ha la necessità di avere una latenza inferiore al secondo e nessun dato perso. Spark Streaming è

migliore se si necessita la garanzia che ogni evento venga processato esattamente una sola volta a patto di accettare una latenza superiore. [15]

	Storm	Spark Streaming
Origin	BackTupe,Twitter	UC Berkeley
Implemented in	Clojure	Scala
Language api	Java	Scala,Java
Processing model	Record at a time	Mini batches
Latency	Sub-second	Few seconds
Fault tolerance-every record processed:	At least once(may be duplicates)	Exactly once

Tabella 1: Confronto tra Storm Spark.

2.9.2 IBM InfoSphere Streams

InfoSphere Streams è un framework per il processamento dei dati in tempo reale, fault tolerance altamente scalabile.L' IDE è basato su eclipse ed offre un'interfaccia per lo sviluppo e la configurazione come la seguente.

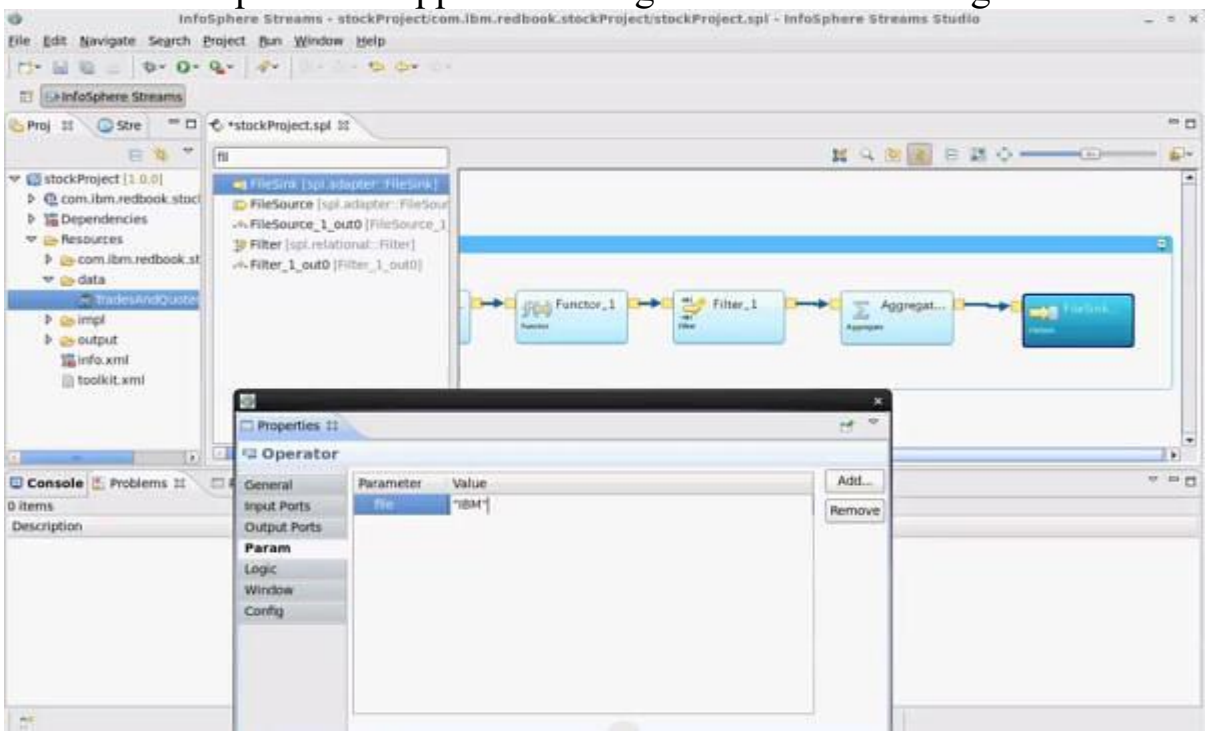


Figura 2.7: UI IBM Streams

Dei ricercatori dell'IBM hanno effettuato uno studio utilizzando un benchmark su Stream e su Storm mettendo in evidenza quanto InfoSphere Stream sia migliore in termini di performance di Apache Storm[17]. IBM per tale studio ha utilizzato un dataset di email pubbliche di 1.5GB, contenente circa mezzo milione di messaggi. I ricercatori hanno effettuato delle operazioni su queste email con Apache Storm e con IBM Streams. Il flow chart che le email hanno seguito è il seguente

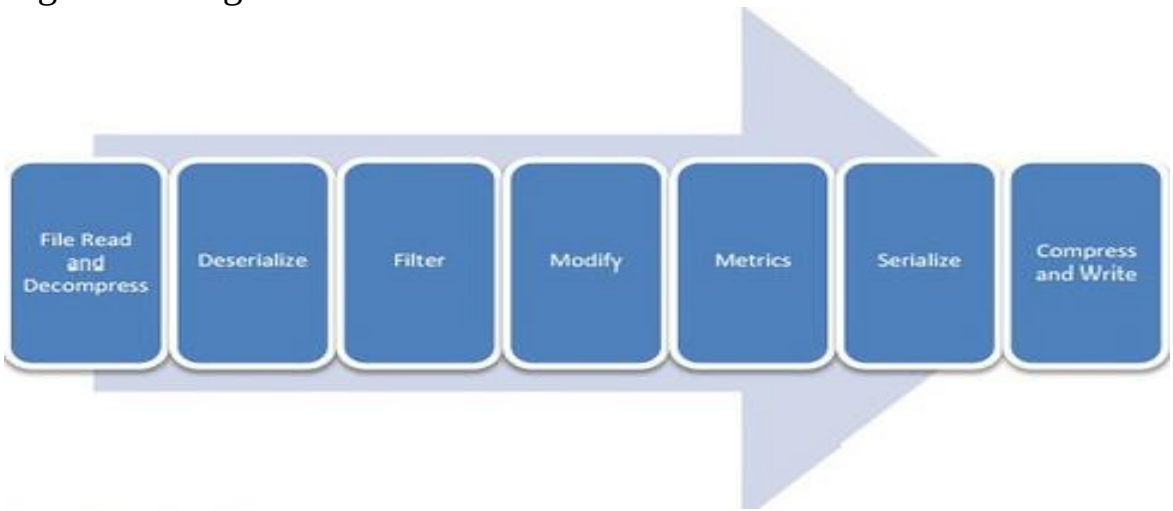


Figura 2.8: FlowChart dataset di email

Dopo aver eseguito questa prova, i ricercatori hanno trovato che il consumo di CPU di Storm è maggiore del consumo di CPU di Streams di un range che varia tra le 4.5 e 13.2 volte, e che il throughput di Streams era maggiore di un range che varia tra 1.6 e 11.3 volte quello di Storm. Di seguito vi sono i risultati del test:

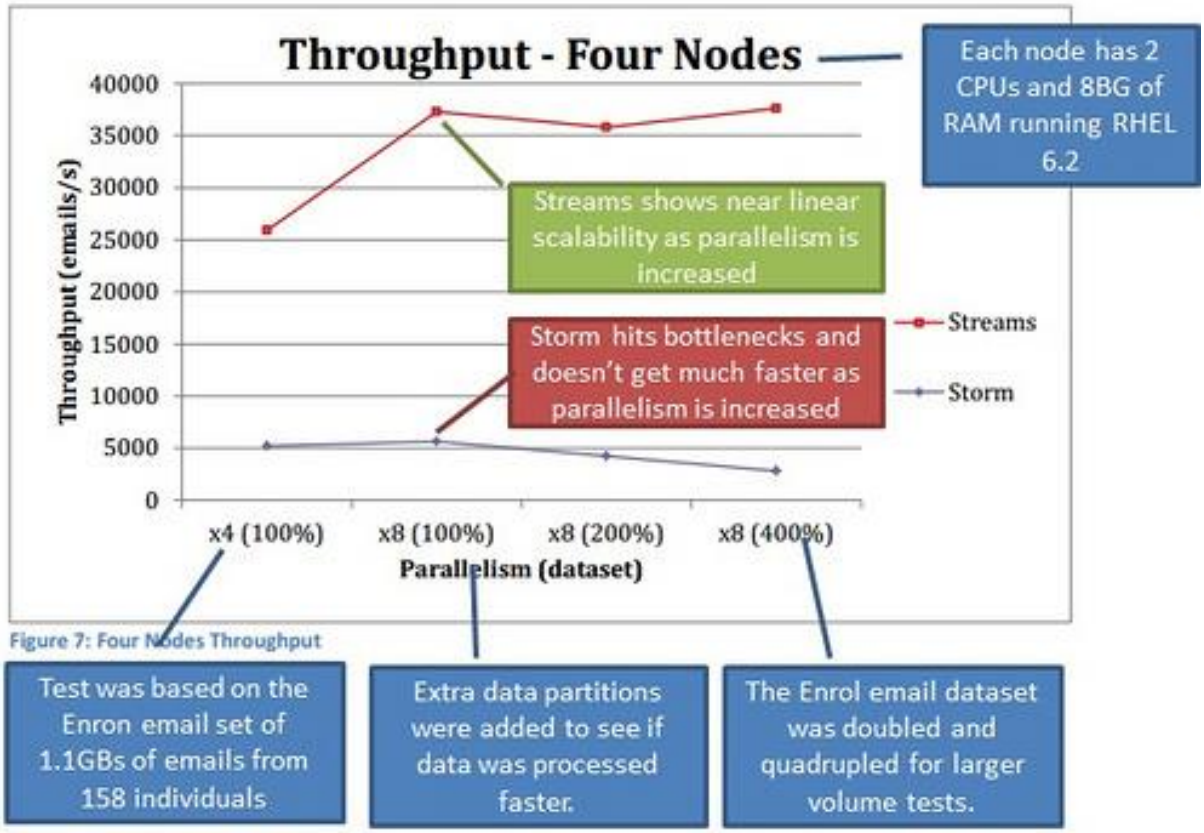


Figura 2.9: Throughput Storm e Streams.

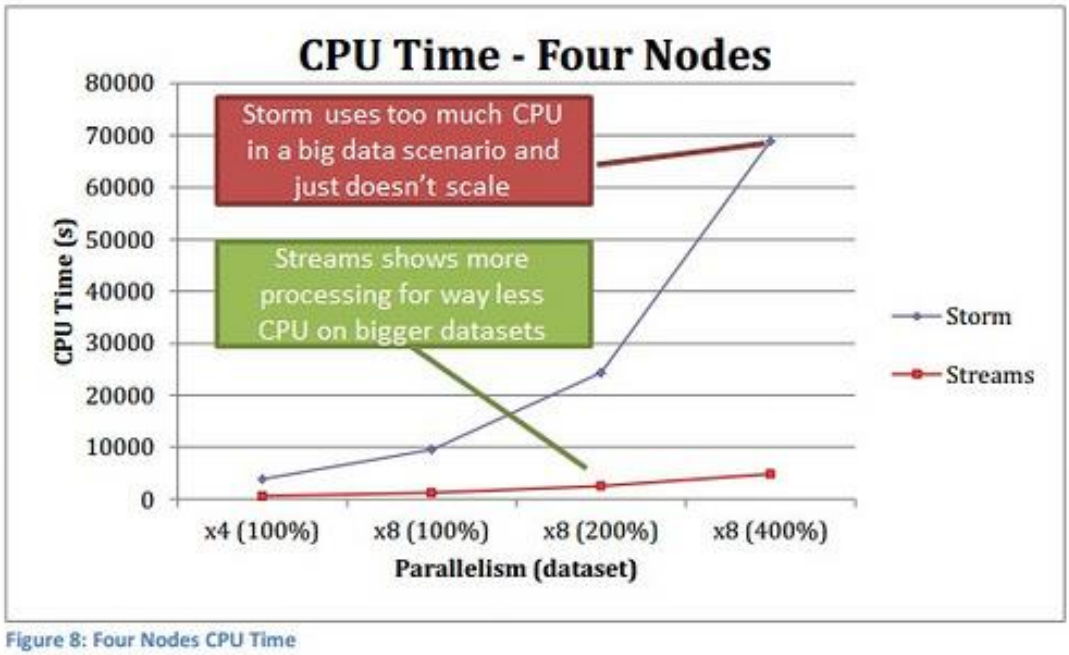


Figura 2.10: Tempo CPU Storm e Streams

I risultati condotti da IBM fanno capire che in alcuni scenari, non si ha alcun vantaggio economico non acquistando la licenza di IBM Streams, utilizzando

un framework open-source, se in seguito bisogna acquistare hardware più performante per avere performance come quelle di Streams.

3. Database NoSQL

3.1 La filosofia del NoSQL

Le difficoltà nell'uso dei DBMS relazionali in certi settori ha richiesto lo sviluppo di soluzioni alternative, comunemente raggruppate sotto il termine NoSQL. NoSQL è l'acronimo di Not only SQL ed è usato per identificare tutti quei database che non fanno uso di un modello di dati relazionale[26].

La filosofia del NO-SQL si può riassumere nei seguenti punti:

- I database relazionali sono troppo costosi e spesso, quelli che svolgono bene il loro lavoro, sono commerciali. NO-SQL abbraccia totalmente la filosofia open-source;
- Il paradigma di programmazione è ad oggetti, NO-SQL è semplice da usare e non occorre uno specialista di DBMS;
- I dati sono altamente portabili su sistemi differenti.
- Non definisce uno schema "rigido" (schemaless) e non occorre tipare i campi, per cui non esistono limiti o restrizioni ai dati memorizzati nei database NO-SQL
- Velocità di esecuzione, di interrogazione di grosse quantità di dati e possibilità di distribuirli su più sistemi eterogenei (replicazione dei dati), con un meccanismo totalmente trasparente all'utilizzatore;
- I DBMS NO-SQL si focalizzano su una scalabilità orizzontale e non verticale come quelli relazionali dove per consentire di contenere più informazioni c'è bisogno di hardware sempre più performante e costoso. I database NoSql permettono di utilizzare commodity hardware evitando dunque di utilizzare hardware particolarmente costoso.

Sebbene i punti di forza riassunti in precedenza, NO-SQL non garantisce i requisiti ACID su cui si basano i sistemi relazionali,ciò può sembrare una grave mancanza in grado di generare incoerenze nelle base di dati. Nel mondo NoSql, un database non garantisce la consistenza in ogni istante di tempo, ma solo quando le attività di modifica ai dati cessano.

Nel 2000 Eric Brewer presenta il teorema CAP, il quale afferma l'impossibilità per un sistema informatico distribuito di fornire simultaneamente tutte e tre le seguenti garanzie [27]:

- Consistency: Dopo ogni operazione il sistema si trova in uno stato consistente: tutti i nodi vedono gli stessi dati nello stesso momento.
- Availability: L'eventuale fallimento di un nodo non blocca il sistema.
- Partition tolerance: Il sistema continua ad operare anche in seguito ad un partizionamento della rete.

  Sulla base di questo teorema è possibile classificare i database distribuiti in tre categorie: CA, CP e AP. Il grafico seguente mostra una panoramica su cosa possono offrire i diversi database in termini di consistenza, disponibilità e partizionamento.

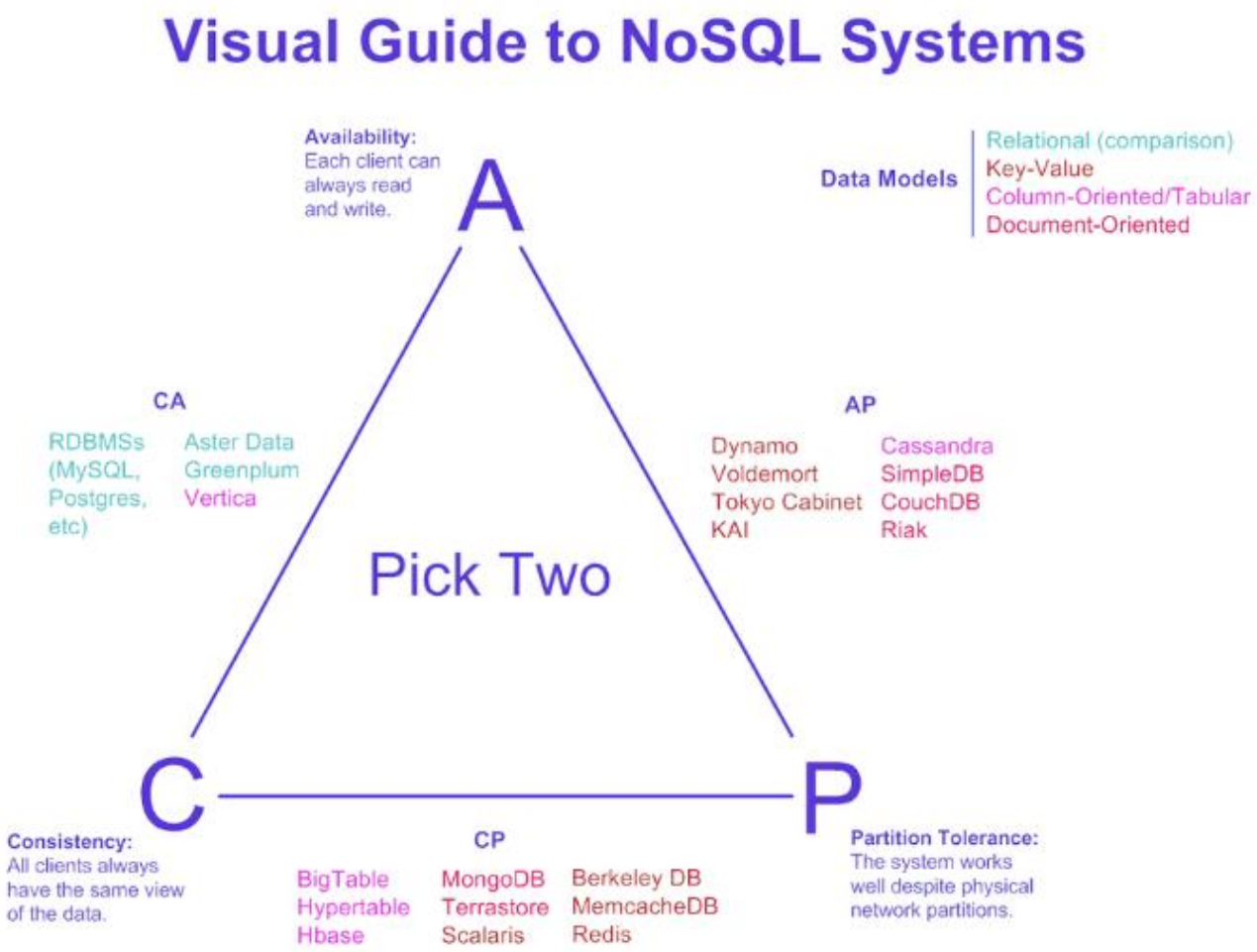


Figura 3.0: Teorema CAP

3.2 Categorizzazioni dei database NoSql

Negli ultimi anni sono state fornite diverse categorizzazioni dei database NoSQL[2][28],

- key/value

-document-oriented

-column-oriented

-graph oriented

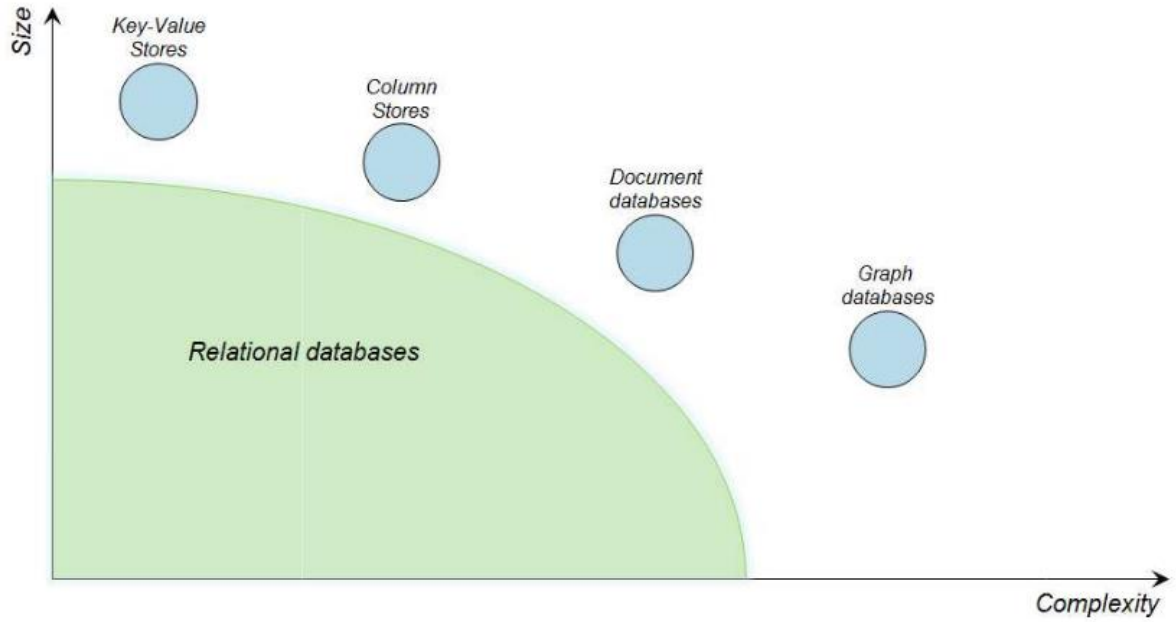


Figura 3.1: Categorizzazione DB

I **database key-value** rappresentano il modello di dati più semplice in quanto sono formati da una tabella associativa(map) che permette, data la chiave, di ottenere il rispettivo valore. In seguito vi sono le operazioni che sono possibili con una tabella associativa, sono molto semplici ma hanno la caratteristica di garantire tempi di esecuzione costanti.

-add: operazione che consente di aggiungere un elemento all'array

-Remove: Operazione che consente di eliminare un elemento dall'array.

-Find: Operazione che consente di ricercare un valore nell'array tramite la chiave

-Modify: Operazione che consente di cambiare il valore associato ad una chiave.

Spesso questi tipi di database non consentono di effettuare operazioni più complesse come aggregazioni e join. Trattandosi di semplici associazioni chiave-valore, questi tipi di database sono i più semplici! Questa semplicità si paga con un maggior numero di dati da inserire per descrivere le relazioni

I **database column-oriented** organizzano i dati per colonna, invece che per riga.

I punti di forza consistono nell'ottimizzare lo spazio evitando di inserire, dove la colonna non presenta un valore, il valore null(nei database con organizzazione per riga il valori nulli occupano spazio). Il modello è molto flessibile perché ogni riga può contenere valori di una o più colonne.

I **database** di tipo **document-oriented** sono strumenti per la gestione di dati semi-strutturati. Questi database sono simili a delle tabelle hash, hanno un unico campo per l'identificazione e valori che possono essere di qualsiasi tipo. I documenti possono contenere strutture nidificate, come ad esempio, documenti, liste o liste di documenti. Tali database sono particolarmente utilizzati in situazioni in cui i dati hanno una struttura dinamica o presentano un gran numero di campi opzionali. Tra questi, il più famoso è MongoDB.

I **database graph-oriented** è costituito da nodi e relazioni tra nodi, essi nascono per rappresentare e navigare in maniera efficiente nei grafi, non hanno uno schema rigido e si adattano facilmente a cambiamenti nelle strutture dei dati. Tra i punti di forza i database graph-oriented consentono di realizzare in maniera efficiente operazioni quali il calcolo del percorso più breve tra due nodi e l'individuazione dei nodi collegati direttamente o tramite altri nodi(la funzione che permette di vedere gli amici in comune con un altra persona sui social Network).

3.3 Una breve introduzione a HBase, MongoDB e Cassandra

3.2.1 HBase

HBase è uno dei database column-oriented la cui realizzazione ha preso spunto da BigTable, infatti sfrutta la memorizzazione di dati distribuiti fornita da Google File System oltre che al concetto di tabella e di column family[29]. Hbase è open source, scalabile e distribuito, caratterizzato da una gestione strutturata dei dati sotto forma di tabelle di grandi dimensioni. Una o più

colonne formano una riga avente un id univoco (Row Key). Tutte le righe sono sempre ordinate in maniera lessicografica in base alla row key come nella figura seguente:

```
hbase(main):001:0> scan 'table1'
ROW                           COLUMN+CELL
row-1                          column=cf1:, timestamp=1297073325971 ...
row-10                         column=cf1:, timestamp=1297073337383 ...
row-11                         column=cf1:, timestamp=1297073340493 ...
row-2                          column=cf1:, timestamp=1297073329851 ...
row-22                         column=cf1:, timestamp=1297073344482 ...
row-3                          column=cf1:, timestamp=1297073333504 ...
row-abc                        column=cf1:, timestamp=1297073349875 ...
7 row(s) in 0.1100 seconds
```

Figura 3.2: Scan Hbase

Una tabella può contenere una o più column family che a loro volta possono contenere una o più colonne. Il nome di una colonna è dato da un prefisso della famiglia (parte fissa) e da un qualificatore (parte variabile) [30]. HBase consente di aggiungere o rimuovere colonne alle famiglie a Runtime in modo tale da adattarsi all'eterogeneità dei dati. Hbase organizza i propri dati in regioni,dove vengono salvate un insieme di righe di una tabella. Inizialmente c'è solo una regione per una tabella, quando una tabella diventa troppo grande per essere contenuta in un server, può essere partizionata orizzontalmente in più regioni. Ogni regione è contenuta esattamente in un region server ed ogni region server può contenere una o più regioni. La figura 3.3 mostra che relazioni vi sono tra i dati, le region e i region server.

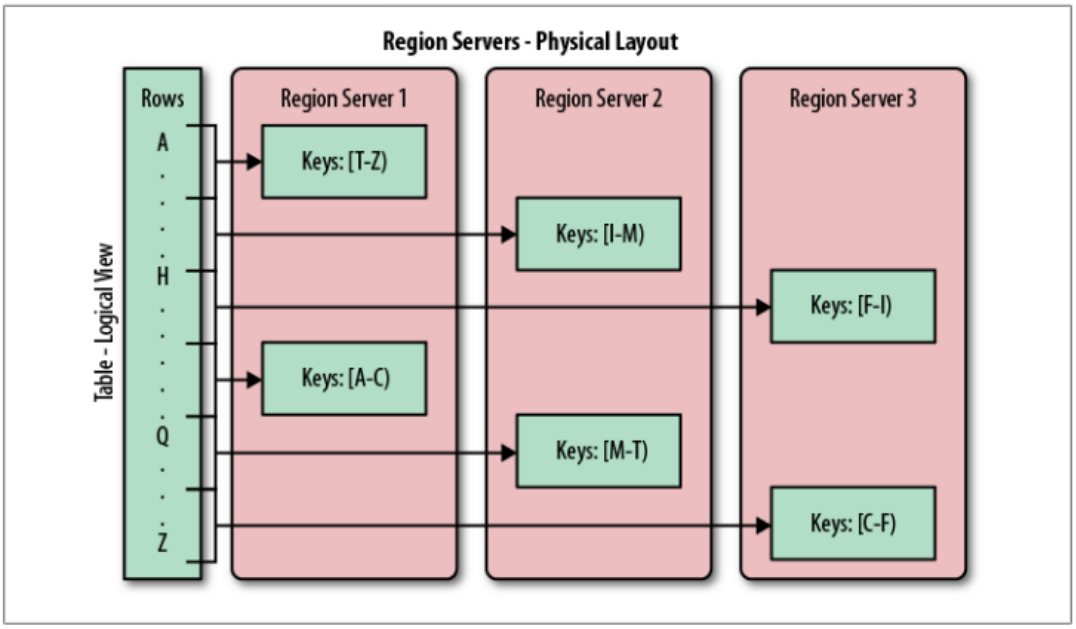


Figura 3.3: Organizzazione delle rows nei Region Server

Da un paper di Google su Big Table del 2006, si evince che il numero di regioni contenute in ogni server varia da 10 a 1000 ed ogni regione ha dimensioni pari a 100 200 MB. Per HBase il numero di regioni contenute in un server varia da 10 a 1000 ed ogni regione ha dimensione pari a 1-2Gb [2]. Ogni regione può essere spostata da un Region Server ad un altro per effettuare un bilanciamento del carico o in seguito ad un failure. HBase si appoggia sul file System HDFS di Hadoop, l'architettura interna è mostrata nella figura 3.4:

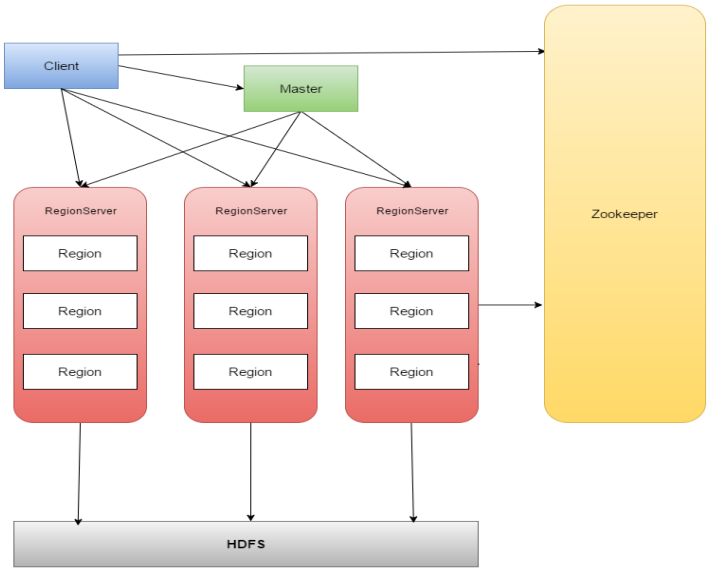


Figura 3.4:Architettura HBase

L’amministrazione di HBase avviene mediante l’uso di strumenti da riga di comando, anche se è disponibile un’interfaccia web attraverso la quale è possibile eseguire operazioni di monitoraggio.
HBase è utilizzato in numerosi scenari in cui la mole di dati raggiunge svariate centinaia di milioni di record per tabella. Facebook utilizza HBase per salvare centinaia di miliardi di messaggi al mese[25]. Esistono diversi progetti di driver JDBC basati sulle librerie native di HBase in grado di utilizzare linguaggio SQL per accedere ai dati memorizzati in tabelle HBase, uno di questi è Apache Phoenix, un progetto incubato da Apache a fine 2013, che ha una bassa latenza nell’esecuzione di query[35]. Esso ottimizza le prestazioni rispetto all’accesso con job MapReduce attraverso l’utilizzo di filtri sui dati, vengono eseguite le clausole WHERE il prima possibile in modo tale da ridurre i dati su cui il resto della query deve lavorare.

3.2.2 Apache Cassandra

Apache Cassandraè un database column-oriented, fault tolerant, eventually consistent e che ha come punto di forza che le performance di lettura e scrittura aumentano aggiungendo nuovi nodi al cluster.
Cassandra rispetta le proprietà di availability e partition tolerance ma non quella della consistency, tuttavia attraverso la configurazione del sistema, è possibile aumentare il livello di consistency, aumentando però anche la latenza delle operazioni. Il modello dati ha un organizzazione che parte dal cluster come entità di più alto livello e scende fino alla colonna. Gli elementi che fanno parte del modello dati di Cassandra sono i seguenti[29]:

- **column**: sono il più basso livello di organizzazione dei dati. Questi elementi base contengono tuple composte da un timestamp, un nome e un valore. Il

timestampserve a Cassandra per capire qual è il valore più recente e quindi per gestire iconflitti. Tipicamente le colonne vengono rappresentate con la notazione JSON.

- **super column:** sono delle colonne nelle quali i rispettivi valori sono altre colonne, una specie di array ordinato di colonne.
- **column family:** è un contenitore di colonne, analogo alla tabella in un sistema relazionale. Ogni column family è salvata in un file separato e i valori sono ordinati per chiave della riga.
- **keyspace:**è un namespace per le column family.
- **cluster:** l'insieme dei server che costituiscono l'istanza di Cassandra. Il cluster può contenere uno o più keyspace.

Cassandra offre un tool scritto in Python per eseguire query con un linguaggio chiamato CQL, che supporta tutte le operazioni SQL eccetto le JOIN.

Cassandra è caratterizzato da scritture molto veloci che, in alcuni casi, sono più performanti delle operazioni di lettura. Questo è un motivo per cui il suo utilizzo è adatto in scenari in cui sia richiesto effettuare molte scritture. Nella tabella sottostante vengono mostrate le performance su un database MySql e un database Cassandra di dimensioni pari a 50Gb[31]:

Cassandra	MySql
0.12 ms in scrittura	300 ms in scrittura
0.15 ms in lettura	350 ms in lettura

Tabella 2

3.2.3 MongoDB

MongoDbè un database NoSQL sviluppato in C++ open source, altamente performante e scalabile linearmente, le sue caratteristiche principali sono le seguenti[2] [32]:

Document-oriented storage: i dati vengono archiviati sotto forma di document in formato BSON, una rappresentazione binaria simile a JSON, con alcuni tipi dato aggiuntivi, facilmente mappabile sia su oggetti JSON sia su strutture tipiche dei linguaggi di programmazione ad oggetti.

Full Index Support: indicizzazione di qualsiasi attributo

Replication & High Availability: facilità nella replicazione dei dati attraverso la rete e alta scalabilità;

Auto-Sharding: scalabilità orizzontale senza compromettere nessuna funzionalità;

Query document-based;

Map/Reduce: aggregazione flessibile e data processing

GridFS: memorizzazione di file di qualsiasi dimensione senza appesantimenti dello stack

Commercial Support: Enterprise support, corsi di apprendimento e consulenze online

Un Mongo system contiene un set di database. Un database contiene un set dicollections, le quali hanno al loro interno un set di documents. Un documento è un set di campi (fields), ognuno dei quali è una coppia chiave-valore dove il valore può contenere, a sua volta, un altro documento o un array di documenti, oltre che a tipi di dato base. Ogni documento ha un campo predefinito, il campo _id e possono avere una dimensione massima di 16 Mb, a meno che non si utilizzi l'estensione di MongoDb GridFS, che supera questo limite suddividendo il file in più chunk. E' possibile interagire con MongoDB tramite

l'utilizzo di una shell o tramite l'utilizzo di un driver. Di seguito vi è un piccolo confronto sulle performance di MongoDb e di MySql.Su un pc a 64 bit, i tempi di risposta delle query di inserimento, a parità di dati, sono i seguenti.

Dati inseriti	Query MongoDB	Query MySQL
100	109 ms	0.06 sec
500	202 ms	0.09 sec
1.000	374 ms	0.12 sec
5.000	1.79 sec	0.23 sec
10.000	3.48 sec	0.41 sec
50.000	17.52 sec	2.01 sec
100.000	34.30 sec	9.73 sec
200.000	58.12 sec	14.76 sec
300.000	1 min 43.12 sec	32.23 sec
500.000	2 min 49.53 sec	2 min 16.33 sec
1.000.000	5 min 51.56 sec	5 min 21.05 sec
2.000.000	13 min 56.06 sec	21 min 10.33 sec
3.000.000	19 min 24.55 sec	1 h 2 min 23.69 sec

Tabella 3: Confronto MongoDB e MySql

Dai risultati mostrati si evince che MongoDB è sicuramente più performante di MySQL per l'inserimento e l'interrogazione di grosse quantità di dati. Su piccole quantità di dati, presenta una latenza trascurabile, guadagnando esponenzialmente nei tempi quando si interrogano milioni di record. Tuttavia, al crescere dei dati, le operazioni di lettura e inserimento di MongoDB subiscono un notevole peggioramento delle prestazioni.

3.4 HBase Cassandra e MongoDB a confronto

Nelle operazioni di inserimento e aggiornamento, HBase e Cassandra si dimostrano più veloci ed elaborano un numero maggiore di documenti rispetto a MongoDB. Tuttavia, la struttura a "colonne" e "chiave-valore" è meno adatta a memorizzare strutture dati complesse, come i grafi, sebbene possano trattare una quantità di informazioni superiore rispetto ai database "document oriented".

Nelle tabelle4 e 5[29] vengono messi a confronto e vengono riassunti gli scenari ideali di utilizzo di questi DB NoSql.

	HBase	Cassandra	MongoDB
Ideale quando:	Quando c'è la necessità di un accesso in lettura e scrittura, random e in tempo reale alla grande quantità di dati.	Quando si scrive più che leggere (logging).	Se si necessita di query dinamiche. Se si preferisce lavorare con gli indici rispetto ad usare algoritmi map/reduce. Buono se si ha bisogno di velocità quando si lavora con grandi moli di dati.
Per esempio:	il database di messaggi in Facebook	Sistemi bancari e industria finanziaria.	tutte le situazioni in cui si vorrebbe usare MySQL o PostgreSQL senza avere colonne definite a priori.

Tabella 4: Confronto tra Database NoSQL (a)

	Cassandra	Hbase	Mongo
Persistence	Yes	Yes	Yes
Replication	Yes	Yes	Yes
High Availability	Distribuited	Distribuited	Distribuited
Transactions	Eventually consistent	Locally(row-level)consistent	Strongly consistent
Rack-locality awareness	Yes	Yes	Yes

Implementation Language	Java	Java	C++
Influences/ sponsor	Dynamo and BigTable,Facebook/Digg	BigTable,Facebook	MetLife,Craigslist, New York Times, Sourceforge
License type	Apache	Apache	Apache

Tabella 5: Confronto tra Database NoSql (b)

4. Caso Studio: il progetto Web-Reputation

4.1 Obiettivi

Il seguente elaborato, svolto nell'azienda Imola Informatica con la collaborazione di TEIA Technologies, si pone l'obiettivo di analizzare in tempo reale dati provenienti da fonti eterogenee, classificarli, renderli persistenti e visualizzarli in tempo reale attraverso dei grafici, al fine di costituire un valido strumento di supporto per i processi decisionali.

4.2 Architettura

Nella figura 4.0 viene mostrata l'architettura di questo progetto.

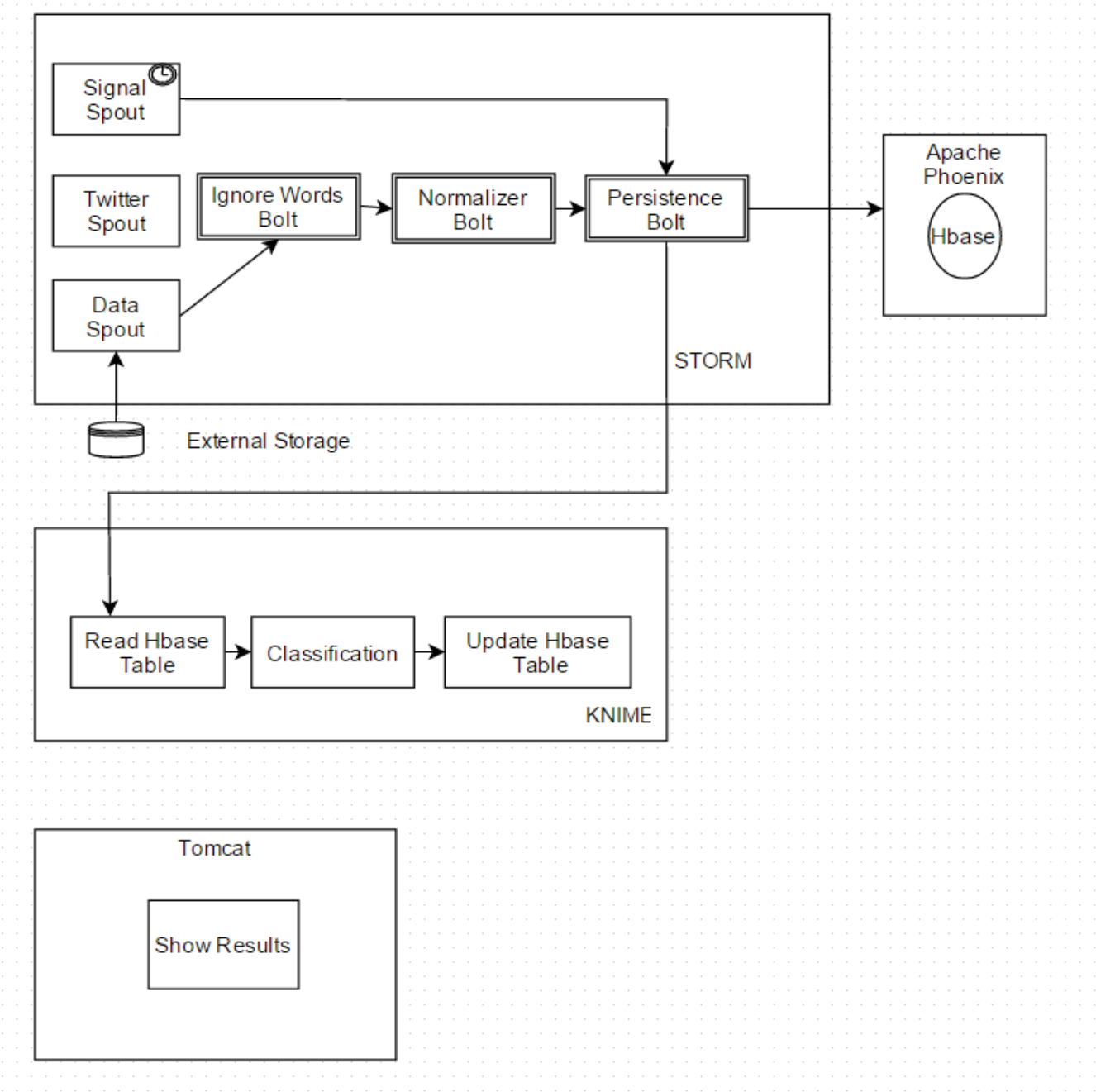


Figura 4.0: Architettura

Vi sono tre fasi fondamentali:

1. Processamento dei dati provenienti da diverse sorgenti
2. Classificazione
3. Visualizzazione dei risultati

Di seguito vengono descritte nel dettaglio queste tre fasi.

4.3 Processamento dei dati provenienti da diverse sorgenti

In questa fase vengono acquisiti dal sistema dati provenienti da Twitter e dati locali (dati da classificare e training set). Twitter è una piattaforma gratuita di social network e microblogging che mette a disposizione delle API che consentono di stabilire una connessione permanente con i server e di raccogliere il flusso di tweet *in tempo reale* fintanto che la connessione è mantenuta; la richiesta può specificare, ovviamente, determinate parole chiave, ma anche utenti, lingua, posizione; è inoltre possibile raccogliere un campione rappresentativo del flusso, non filtrato per alcun parametro [30]. Le parole d'interesse da monitorare insieme ai Path relativi ai training set(uno per ogni keyword di interesse) e ai Path relativi ai dati in locale devono essere inserite in un file di configurazione. Nel Twitter Spout è stata implementata tutta la logica per monitorare il flusso di dati provenienti da Twitter. Ogni Tweet contiene oltre al campo testuale, molte altre informazioni aggiuntive come l'utente che ha postato quel tweet, il numero di volte che il messaggio è stato retweettato, la lingua del messaggio, le coordinate geografiche del luogo da cui il messaggio è stato postato, data di creazione del messaggio.

Nell'applicazione realizzata è stato scelto di utilizzare solo le seguenti voci:

- Id univoco del Tweet(Url)
- Testo
- Lingua

Queste informazioni vengono passate come parametro ad un Oggetto serializzabile chiamato MessageText ed inviate al Bolt IgnoreWords. Tale oggetto oltre ad essere serializzabile per poter essere convertito in una tupla e per poter essere inviato agli altri Bolt, deve essere registrato nella Topologia tramite il seguente metodo:config.registerSerialization(MessageText.class);

In questa prima fase vengono scartati tutti i tweet il cui contenuto non contiene nessuna keyword inserita nel file di configurazione. Per ogni parola di un tweet, viene controllato se vi è un match con la lista di parole relative agli argomenti da monitorare, in caso positivo viene creata un'istanza di MessageText con i suddetti parametri, e viene invocato il metodo su tale istanza ogni volta che viene rilevata una keyword in un singolo tweet addWord(String keyword). E' possibile che un tweet contenga più keyword e quindi deve essere classificato con più training set diversi. Un altro Spout è il DataSpout che si occupa di caricare i TrainingSet e dati salvati in locale. Ogni messaggio caricato viene passato come parametro al metodo setText(String tweet) dell'oggetto MessageText. Tale oggetto viene convertito in una tupla sulla quale viene apposta un'etichetta che rappresenta la natura del messaggio(tweet del training o tweet da classificare). Come ultimo Spout vi è il SignalSpout il quale invia una tupla al PersistenceBolt ad intervalli di tempo predefiniti apponendo su di essa un'etichetta. Tale tupla serve per avviare Knime in batch ed avviare la classificazione dei dati ancora non classificati.

Il Bolt IgnoreWords la prima volta che viene istanziato, carica una lista di Stopword, una lista per ogni lingua. Ogni parola contenuta in questa lista viene eliminata dal tweet. Questa tecnica permette di ridurre il bag of words eliminando tutte le parole che non sono rilevanti per la comprensione di una frase (Es congiunzioni, preposizioni) [24]. Le stopword sono dipendenti dalla lingua, è possibile trovare una lista di stopword per le diverse lingue qui [18]. Il tweet senza le stopword viene passato come parametro all'istanza del MessageText (la stessa ricevuta in ingresso del Bolt IngoreWords)la quale viene convertita in una tupla ed inviata al Normalizer Bolt.

In questo Bolt vengono effettuate le seguenti operazioni:

- Riconoscimento della lingua del tweet

- Stemming
- Rimozione degli URL
- Rimozione caratteri speciali
- LowerCase del testo

Viene effettuato un riconoscimento della lingua del tweet perché le informazioni fornite da Twitter in merito alla lingua del testo, sono spesso affette da errore. Twitter si basa sulla provenienza del messaggio e non su una analisi accurata del testo per determinare la lingua del medesimo. Nell'applicazione sviluppata viene utilizzata per determinare la lingua del messaggio la libreria UberLanguageDetector che riconosce ben 70 lingue[21]. Tale libreria utilizza dei modelli serializzati per classificare e fornisce dei metodi per crearne nuovi per riconoscere dei testi scritti in lingua ancora non supportata. In base alla lingua predetta viene effettuata la fase di stemming che consente di portare alla radice una parola (Es.Abbassando, abbassare, abbassai in abbass) riducendo ulteriormente il BagOfWords. Da sempre l'informatica si confronta con lo stemming, questo processo è ad esempio parte integrante dei motori di ricerca che lo utilizzano per ampliare ed espandere le interrogazioni e riuscire così ad elaborare il linguaggio utilizzato. Per effettuare lo stemming esistono diversi algoritmi tra cui [23]:

- Porter: senza dubbio lo stemmer più comunemente usato, anche il meno aggressivo. Uno dei pochi stemmer che ha un supporto per Java, anche se è quello computazionalmente più intensivo degli altri algoritmi. E' il primo algoritmo che è nato per lo Stemming.
- Snowball: Questo Stemmer è un miglioramento dello stemmer di Porter. Risulta migliore in termini di velocità ed ha un supporto per Java[23]. È lo stesso Porter che ammette che lo Snowball stemmer è migliore del suo algoritmo originale.

- Lancaster: questo stemmer risulta essere molto aggressivo. Spesso effettua lo stemming in modo errato. Mentre con l'algoritmo di Porter e la sua versione migliorata Snowball le parole che vengono portate alla radici sono comprensibili dall'essere umano, le parole ridotte con lo Stemmer Lancaster sono difficilmente interpretabili; Es. applicando tale stemmer alla parola "things", il risultato è "th", dovrebbe essere "thing", lo stemming di living dovrebbe essere liv, invece il risultato è l.

  Per evitare di eliminare informazioni utili, nell'applicazione sviluppata non si è optato per l'algoritmo di Lancaster, ma si è scelto di utilizzare lo Snowball Stemmer in quanto risulta essere meno aggressivo, più efficace di Porter e compatibile con Java[23].

  Le API utilizzate consentono di effettuare lo stemming su testo scritto in dieci lingue diverse. Essendo fortemente dipendente dalla lingua del testo, bisogna dichiarare la lingua del messaggio prima di avviare il processo di stemming. Nella figura 4.1 vi è un esempio di messaggio a cui sono state eseguite le due tecniche descritte in precedenza.

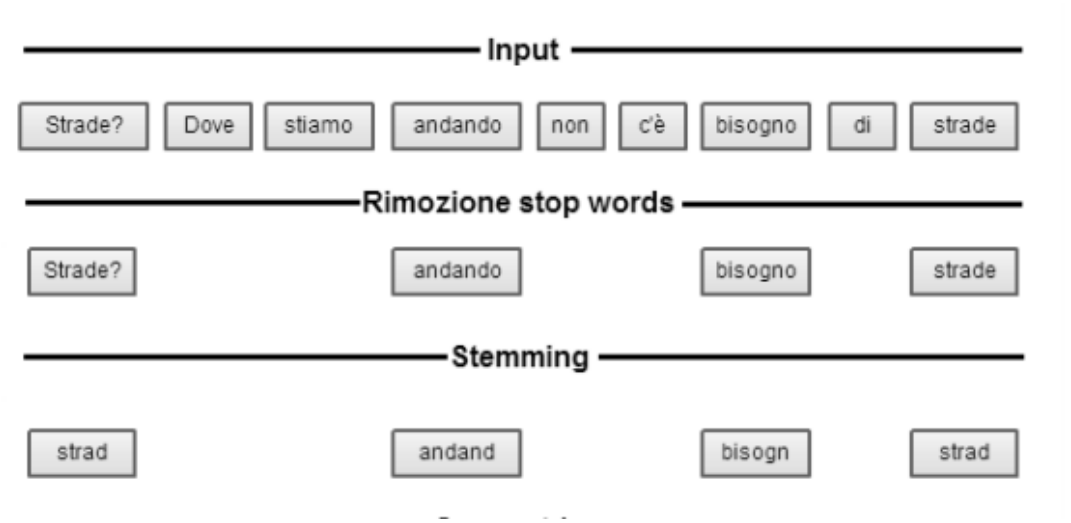


Figura 4.1: Preprocessing del testo

Come terza fase vengono rimossi tutti gli URL, caratteri speciali e viene fatto il Lower case del testo, queste operazioni sono necessarie per diminuire il Bag Of Words [33]. Inoltre vengono rimosse le parole che hanno meno di tre caratteri in quanto non sono determinanti per la comprensione del messaggio. Per rimuovere gli URL è stata utilizzata un'espressione regolare.

Le espressioni regolari rappresentano uno strumento molto potente per lavorare sulle stringhe. La sintassi di questo pseudo-linguaggio è molto flessibile. Java supporta le espressioni regolari con un package java.util.regex composto dalle classi Pattern e Matcher che permettono di validare una stringa, o ricercare un testo al suo interno, a partire da un'espressione regolare[34]. Di seguito vi è il metodo utilizzato nell'applicazione:

```
private String removeUrl(String tweet) {
        Pattern p = Pattern.compile("(https?|ftps?|file)://[^\\s]+",
                Pattern.CASE_INSENSITIVE);
        Matcher m = p.matcher(commentstr);
        if (m.find()) {
            return m.replaceAll("");
        }
return commentstr;}
```

Dopo aver effettuato la "pulizia" del tweet, il messaggio viene passato come parametro ad un metodo dell'oggetto MessageText il quale viene convertito ed inviato al Persistence Bolt. Tale Bolt legge l'etichetta appostasulla tupla ricevuta, sono possibili diversi scenari:

- Il mittente è il SignalSpout
- Il mittente è il TwitterSpout.
- Il mittente è il DataSpout.

Nel primo caso viene avviato uno script per avviare KNIME in batch per eseguire la classificazione. È possibile avviare un Flow Chart di Knime senza interfaccia grafica, direttamente da riga di comando. Per tale motivo è stato

scritto uno script che accetta dei parametri quali il percorso del flow chart, il nome della tabella HBase dei dati da classificare, il nome della tabella del training set, il percorso dove salvare i file di output. Per avviare lo script viene utilizzato il seguente codice:

```
String s;
Process p;
p = Runtime.getRuntime().exec("/usr/hdp/current/knime-full_1.11.1/script.sh "
                                    + PropertyReader.PATH_KNIME + ""+ currentPath + ""
                                    + name.toUpperCase() + ""+ PropertyReader.PATH_OUTPUT + name
                                    + ".csv");
BufferedReader br = new BufferedReader(
        new InputStreamReader(p.getInputStream()));
                while ((s = br.readLine()) != null)
        System.out.println("line: " + s);
        p.waitFor();
        System.out.println("exit: " + p.exitValue());
        p.destroy();
```

Negli ultimi due casi,viene caricata la parola o la lista di parole passate come parametro all'oggetto MessageText nello Spout(parole che hanno fatto match con le keyword inserite in fase di configurazione). Nel caso in cui il mittente sia il TwitterSpout, per ogni parola caricata viene visto se esiste una tabella in Hbase che abbia come nome della tabella la parola stessa. Nel caso in cui non esistono tabelle viene creata una tabella utilizzando Apache Phoenix il quale permette di utilizzare un database NoSql come se fosse Sql. Si è scelto di utilizzare Apache Phoenix in quanto risulta essere più performante per risolvere le query; attraverso l'utilizzo di filtri sui dati, vengono eseguite le clausole WHERE il prima possibile in modo tale da ridurre i dati su cui il resto della query deve lavorare [35] [2]. La tabella creata ha la seguente struttura:

Url	Type	Text	Original Text	Language	Author	Prediction

Tabella 6: Schema dati da classificare

Se esiste una tabella invece viene aggiunta una nuova entry. L'URL è la chiave primaria in quanto identifica univocamente un tweet, il Type sta ad indicare la fonte del messaggio(Twitter o locale), il Text indica il tweet "pulito", il campo Original Text indica il messaggio senza nessun tipo di pre-processing, Language indica la lingua del messaggio che è stata predetta nel Normalizer Bolt, Author l'autore del messaggio, Prediction viene inserita una stringa vuota la quale verrà aggiornata in seguito da KNIME. Nel caso in cui il mittente sia il DataSpout, viene analizzata un'ulteriore etichetta apposta dal DataSpout stesso, per capire se il messaggio ricevuto sia un messaggio di training o un messaggio che si vuole classificare. Nel primo caso, per ogni parola caricata viene visto se esiste una tabella in HBase che abbia come nome della tabella la medesima parola più la stringa "TRAINING". In caso negativo viene creata una tabella che ha la seguente struttura:

Text	Original Text	Language	Prediction

Tabella 7: Schema dati training

In caso positivo viene aggiunta una nuova entry nella tabella. Il caso in cui il MessageText inviato dal DataSpout contenga un messaggio salvato in locale che si vuole classificare, viene gestito come se fosse stato inviato dal TwitterSpout.

4.4 Classificazione

La classificazione testuale o text categorization è un'attività che consiste nell'assegnare in maniera automatica etichette predefinite a testi scritti in linguaggio naturale. Per realizzare ciò si utilizzano solitamente degli approcci di apprendimento automatico di tipo supervisionato, che prendono in input un set di testi e restituiscono in output un modello generale per la classificazione di nuovi testi. Esistono tuttavia altri approcci, come quello non supervisionato o semi-supervisionato, ma solitamente con risultati peggiori.

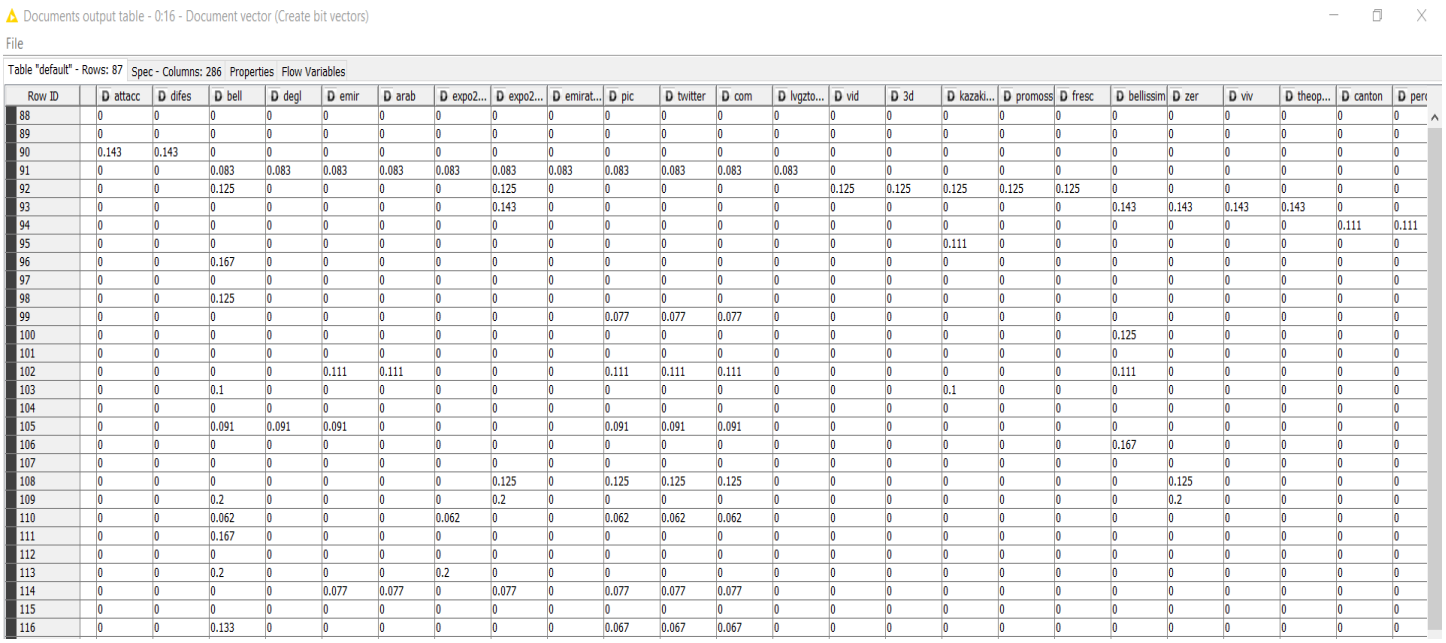
Frasi di un testo non possono essere direttamente interpretate dagli algoritmi di classificazione come quando lavorano con features di tipo numerico. La traduzione di un testo da classificare in un vettore di attributi è quindi un passo necessario e fondamentale per la prestazione del lavoro di classificazione. Il processo che trasforma una frase in un insieme di termini comprensibili da un classificatore è chiamato Feature extraction. Esistono diversi metodi di feature extraction come ad esempio la tecnica del Bag of Word. Nel BoW si considerano tutte le parole principali contenute in un documento scritto e le si memorizzano in un dizionario, costruito attraverso una fase di training. Nel BoW un documento è visto come un punto (vettore) nello spazio delle parole del dizionario (**feature**):$\mathbf{D_i = (w_{i,1}, w_{i,2} , w_{i,3} , \dots , w_{i,n})}$.

Ogni termine $w_{i,k}$ è il peso della parola **k** nel documento **i**: è possibile assegnare un valore a $w_{i,k}$in diversi modi [36]:

1. $w_{i,k} = \begin{cases} 1 & \text{se il termine k appare nel documento i} \\ 0 & \text{altrimenti} \end{cases}$
2. $w_{i,k} = n_i$ numero di volte cheil termine k appare nel documento i
3. tf.idf: $w_{i,k} = \dfrac{\text{frequenza del termine k nel documento i}}{\text{frequenza del termine k nell'intera collezione}}$
4. …altri

Figura 4.2: Tecniche per l'assegnamento di un peso per un attributo del BoW

TF-IDF associa maggiore importanza ai termini meno frequenti nel corpus del documento ma più rilevanti per la ricerca. Le parole comuni sono molto frequenti ma, essendo anche presenti in tutti i documenti, hanno una ridotta importanza nel processo di selezione delle risorse [19][42] ed è proprio per questo motivo che nell'applicazione svolta è stata scelta tale tecnica. La figura 4.3 riporta un esempio di applicazione di tf idf realizzato con KNIME dove nella prima colonna vi sono tutti i tweet di un dataset, in tutte le altre colonne vi sono tutti i termini trovati nel dataset ed il valore di ogni cella corrisponde al valore del tf-idf calcolato con la tecnica 3 di figura 4.1.


Documents output table - 0:16 - Document vector (Create bit vectors)

File

Table "default" - Rows: 87 Spec - Columns: 286 Properties Flow Variables

Row ID	attacc	difes	bell	degl	emir	arab	expo2...	expo2...	emirat...	pic	twitter	com	lvgzto...	vid	3d	kazaki...	promoss	fresc	bellissim	zer	viv	theop...	canton	per
88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
89	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
90	0.143	0.143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
91	0	0	0.083	0.083	0.083	0.083	0.083	0.083	0.083	0.083	0.083	0.083	0.083	0	0	0	0	0	0	0	0	0	0	0
92	0	0	0.125	0	0	0	0	0.125	0	0	0	0	0	0.125	0.125	0.125	0.125	0.125	0	0	0	0	0	0
93	0	0	0	0	0	0	0	0.143	0	0	0	0	0	0	0	0	0	0	0.143	0.143	0.143	0.143	0	0
94	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.111	0.111
95	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.111	0	0	0	0	0	0	0	0
96	0	0	0.167	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
97	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
98	0	0	0.125	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
99	0	0	0	0	0	0	0	0	0	0.077	0.077	0.077	0	0	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.125	0	0	0	0	0
101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
102	0	0	0	0	0.111	0.111	0	0	0	0.111	0.111	0.111	0	0	0	0	0	0	0.111	0	0	0	0	0
103	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0
104	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
105	0	0	0.091	0.091	0.091	0	0	0	0	0.091	0.091	0.091	0	0	0	0	0	0	0	0	0	0	0	0
106	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.167	0	0	0	0	0
107	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
108	0	0	0	0	0	0	0	0.125	0	0.125	0.125	0.125	0	0	0	0	0	0	0	0.125	0	0	0	0
109	0	0	0.2	0	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0.2	0	0	0	0
110	0	0	0.062	0	0	0	0.062	0	0	0.062	0.062	0.062	0	0	0	0	0	0	0	0	0	0	0	0
111	0	0	0.167	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
112	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
113	0	0	0.2	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
114	0	0	0	0	0.077	0.077	0	0.077	0	0.077	0.077	0.077	0	0	0	0	0	0	0	0	0	0	0	0
115	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
116	0	0	0.133	0	0	0	0	0	0	0.067	0.067	0.067	0	0	0	0	0	0	0	0	0	0	0	0

Figura 4.3: Un esempio di BoW adottando la tecnica Tf-ifd

Spesso capita che, una volta finito il processo di estrazione delle features e di creazione del dizionario, l'insieme dei termini sia troppo esteso; questo fatto può causare sprechi computazionali dell'algoritmo e problemi di overfitting con

una conseguente cattiva classificazione [37]. Per ovviare a questo problema nell'applicazione svolta, prima di effettuare la fase di feature extraction vengono effettuate tecniche come lo stemming e la rimozione delle stopword:
Una volta eseguita la fase di feature extraction si passa alla fase della classificazione. Esistono numerose tecniche di classificazione:

- Metodi basati sugli alberi di decisione
- Metodi basati su regole
- Ragionamento basato sulla memoria
- Reti Neurali
- Bayes "Naive" e Bayesian Belief Networks
- Support Vector Machines

Di seguito ci soffermiamo sugli alberi decisionali, SVM e classificatori Bayesani.

4.3.1 Alberi decisionali

Il metodo degli alberi decisionali viene spesso scelto per la capacità di generare regole chiare e comprensibili. Gli algoritmi impiegati negli alberi decisionali posizionano l'attributo che offre migliori probabilità di segmentazione al nodo radice dell'albero stesso. Per ogni livello dell'albero viene fatto un test su un attributo e viene preso come nodo l'attributo che ha Information Gain maggiore.
Gli alberi decisionali si suddividono in due tipologie:

- **Alberi di classificazione:** Assegnano un'etichetta ai recordo e quindi alle classi appropriate; possono anche indicare il livello di confidenza sulla classificazione effettuata. In questo casa l'albero di classificazione

fornisce la probabilità della classe, cioè il livello di confidenza di appartenenza ad una classe

- **Alberi di regressione:** Stimano il valore di una variabile target che assume valori numerici.

Entrambe le tipologie presentano la stessa struttura. Quando un modello ad albero vieni applicato ai dati, ogni record viene analizzato dall'albero lungo un percorso stabilito da una serie di test finché il record non raggiunge un nodo terminale dell'albero. Qui viene assegnata un'etichetta di classe, oppure, nel caso degli alberi di regressione, viene assegnato un valore numerico. [20]

4.3.2 Classificatori Bayesani

Il classificatore Bayesiano esegue una classificazione di tipo statistica improntata al calcolo della probabilità delle cause (o probabilità a posteriori), cioè, avvenuto un certo evento si determina la probabilità di quale causa lo abbia scatenato. Il classificatore deriva direttamente dal teorema introdotto dal matematico Thomas Bayes. Questo tipo di classificatore si indica come naive perché è basato sull'assunzione esemplificativa che tutti gli attributi (features) che descrivono una certa istanza sono tra loro indipendenti data la categoria a cui appartiene l'istanza. Questa affermazione viene detta assunzione del Naive Bayes. L'assunzione fatta è molto forte, più gli attributi sono correlati tra loro, tanto peggiore saranno le classificazioni fatte. Nonostante questa assunzione sia violata nella maggior parte dei problemi reali come, ad esempio, nella categorizzazione del testo, il Naive Bayes si comporta molto bene e risulta essere molto efficace.[38]

4.3.3 Support Vector Machine

Una SVM è un classificatore binario che apprende il confine fra esempi appartenenti a due diverse classi. Funziona proiettando gli esempi in uno spazio multidimensionale e cercando un iperpiano di separazione in questo spazio. L'iperpiano di separazione massimizza la sua distanza (il “margine”) dagli esempi di training più vicini. Tale classificatore gode di alcune proprietà come la capacità di gestire dati con molte caratteristiche descrittive, compattamento dell'informazione contenuta nel data set in input, improbabile overfitting.

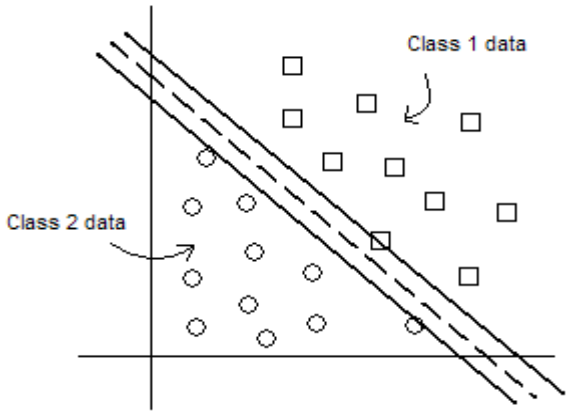


Figura 4.4: Funzionamento del SVM

Dati che non sono linearmente separabili vengono mappati in uno spazio di dimensione superiore utilizzando una funzione di mapping in cui essi siano linearmente dipendenti. Uno spazio di dimensioni maggiore causa però seri problemi di calcolo perché l’algoritmo di apprendimento deve lavorare con vettori di grandi dimensioni [22].

4.5 Valutazione dei classificatori

Per classificare e valutare l’accuratezza del classificatore, è importante capire come stabilire teste training set. Il processo di Bernoulli ci indica la probabilità che si ripeta un evento dopo l’avvenuta successione di eventi binari, e ci serve proprio per capire come realizzare training e test set.

Uno dei metodi più usati è il cosiddetto hold-out [38], per cui dal data set a disposizione si ricavano sia il training che il test set (in generale come rapporto 2/3,1/3). Tuttavia, questo sistema risulta essere sicuramente poco efficiente, poiché vi può essere l'alto rischio che fra training e test set le classi non siano rappresentate nella stessa maniera (una classe potrebbe essere sotto rappresentata). Per questo si usa l'hold-out stratificato per rappresentare le classi in maniera uguale fra i due set. Si utilizza il metodo della cross validation: si partiziona l'insieme dei dati, e si tiene una sola partizione per il test, mentre il resto viene utilizzato come training. Si itera quindi k volte, tante quante sono le diverse partizioni (in generale si assume come k 10), realizzando quindi k valutazioni diverse; in questo modo poi si valuta una qualità media, e per il classificatore finale si utilizza l'intero data set di dati etichettati. Una particolare tecnica consiste nel assumere k come il numero degli oggetti: in questo caso si parla di leave oneout, che è un caso estremo; non effettua campionamenti casuali[38].

L'accuratezza è definita come il numero di campioni correttamente classificati rispetto al numero totale di campioni classificati e la formula è la seguente: (TP+TN)/(TP+TN+FP+FN) dove TP indica le il numero delle istanze positive classificate correttamente, TN il numero delle istanze negative classificate correttamente, FP il numero dei falsi positivi, FN il numero dei falsi negativi;

Tasso di errore: 1-Accuratezza.

Nell'applicazione svolta, per calcolare l'accuratezza si è utilizzata la tecnica hold-out senza reimbussolamento (le istanze che fanno parte di una partizione vengono tolte dal data set, per cui non possono far parte di nessun'altra partizione), dal data set a disposizione sono state prese in maniera random il 75% delle istanze per il training set ed il 25% delle istanze per il test set. Come

ulteriore verifica si è scelto di utilizzare un testset etichettato manualmente. I Risultati sono riportati nel capitolo 5.

4.6 Piattaforme per la classificazione

Per classificare i dati sono state studiate due piattaforme: WEKA e KNIME.

Weka è un ambiente software interamente scritto in Java. Un semplice metodo per utilizzare questo software consiste nell'applicare dei metodi di apprendimento automatici ad un set di dati, e analizzarne il risultato [40].

KNIME è una piattaforma modulare per l'esplorazione dei dati, che consente graficamente di creare data flow, eseguire analisi, e ispezionare i risultati[39]. Knime è basato su Eclipse, e nella sua versione base comprende numerosi nodi di I/O, di analisi, di data mining e nodi per permettere di scrivere delle proprie funzioni in diversi linguaggi come Java, Python, C; Inoltre Knime fornisce dei nodi per utilizzare funzioni di Weka. Data la potenza espressiva (Database Reader/Writer, Joiner, Java Snippet), nell'applicazione svolta si è scelto di utilizzare Knime. Di seguito viene riportato il flow chart in Knime per la classificazione.

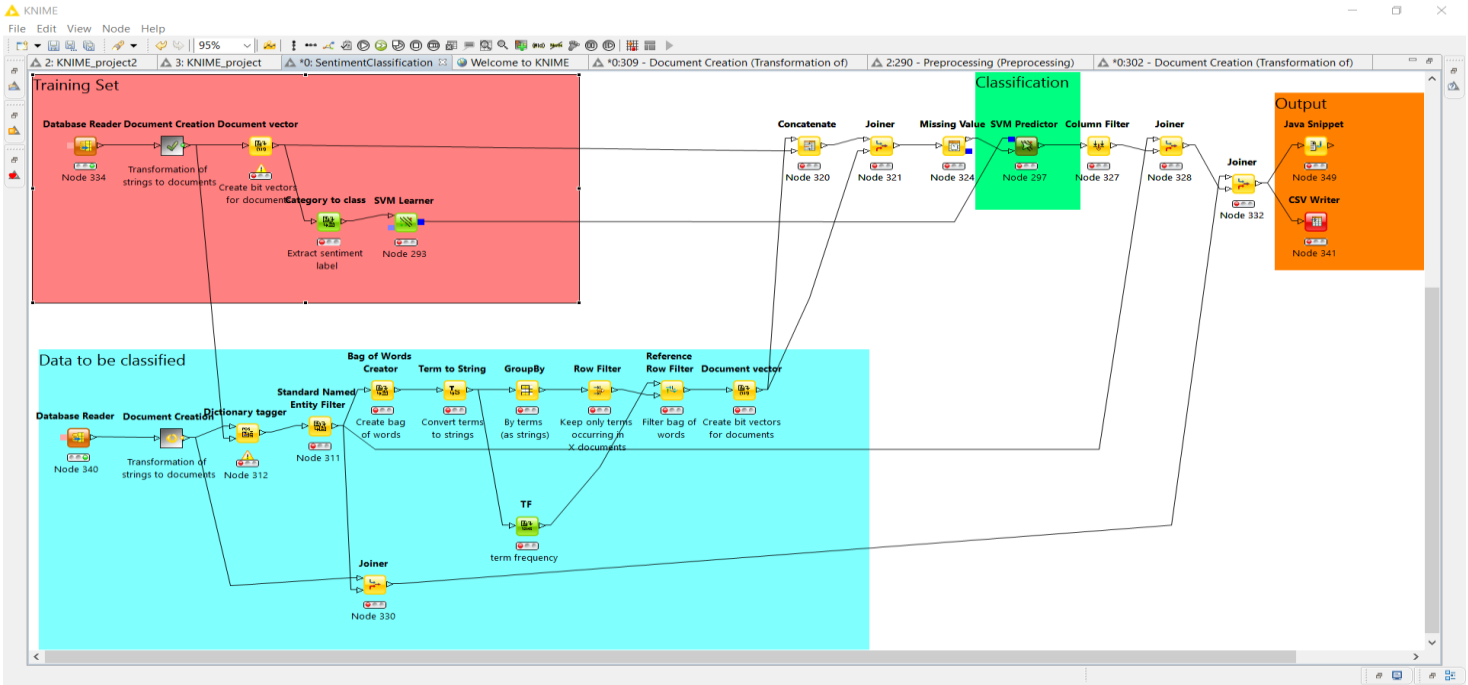


Figura 4.5: Flow Chart Knime

In ingresso vi sono i nodi relati all'input. Sono stati utilizzati i driver jdbc di apache phoenix per interfacciarsi con Hbase. Il nome della tabella viene passato dinamicamente da Apache Storm ai nodi Database Reader quando viene invocato lo script ad intervalli di tempo predefiniti. Tramite il nodo Bag of Words Creator, viene creato il Bag of Word, un vettore che ha per attributi l'insieme di tutte le parole trovate nei tweet. Tutte le parole che sono contenute nei tweet da classificare e non sono presenti nel training set vengono scartate, questa operazione è stata fatta tramite il nodo Dictionary tagger ed il nodo Standard Named Entity Filter,il primo crea un dizionario avente tutti i termini del training set ed appone un'etichetta sui termini dei tweet da classificare che non sono contenuti in tale dizionario, il secondo filtra tutti i termini che hanno l'etichetta messa nel nodo precedente. Per tale motivo è indispensabile avere un training set che abbia quasi tutte le parole importanti per un dominio. Ad esempio, se si classificano i tweet relativi ad un telefono, è indispensabile avere un training set che comprenda istanze che abbiano parole che con buona probabilità vi sono anche nei dati da analizzare, che possono essere fotocamera buona, batteria pessima, il telefono si blocca etc. Quante più parole nuove vi sono nel training set, tante meno parole scartate si avranno in fase di analisi garantendo così una maggior accuratezza.

4.7 Trasformazione della dimensionalità dei feature vector

Per utilizzare il SVM è risultato indispensabile trasformare la dimensionalità del feature vector del tweet da classificare, che chiameremo FV1, nella dimensionalità del feature vector del training set, che chiameremo FV1. Se si ha un FV1 di 1000 attributi (un attributo corrisponde ad una nuova parola) e un FV2 di 100 attributi, vengono scartati tutti gli attributi che sono presenti solo in FV2; in FV2 saranno aggiunti tutti gli attributi che sono esclusivamente FV1.

Di seguito viene riportato un esempio: Per semplificare tale esempio non viene utilizzata la tecnica Tf-Idf ma viene posto a 1 il valore di un termine nel vettore se il tweet contiene quel termine, 0 altrimenti.

Tweet training

Tweet 1: telefono buono, prestazioni buone; Positivo

Tweet 2: telefono pessimo, prestazioni pessime; Negativo

Tweet da classificare

Tweet 3: prestazioni pessime, fotocamera buona, sconsigliato

Bag of Words Training

	telefono	buono	prestazioni	pessime	pessimo	Buone
Tweet1	1	1	1	0	0	1
Tweet2	1	0	1	1	1	0

Tabella 7 Applicazione del Bag of Words al Training Set

Bag of Words dei dati da classificare prima di riportare entrambi i vettori alla stessa dimensionalità

	Prestazioni	pessime	fotocamera	buona	sconsigliato
Tweet3	1	1	1	1	1

Tabella 8: Applicazione del Bag of Words al Dataset da classificare

Di seguito vengono scartate le parole: fotocamera, pessima, sconsigliato perché non sono presenti nel primo Bag of Words, e vengono aggiunte le parole che sono solo nel primo Bag of Words: telefono, buono, pessimo,buone. Il risultato è il seguente:

	telefono	buono	prestazioni	pessime	pessimo	Buone
Tweet3	0	0	1	1	0	0

Tabella 9: Trasformazione di dimensionalità del feature vector

Da questo esempio si evince che è importante avere un training set che abbia quante più parole che sono contenute nei tweet da classificare.

Questa fase è stata fatta tramite un nodo concatenate che ha per ingresso i BoW del Training set e del dataset da classificare,un nodo join ed un nodo missing value. Dopo questa fase il tweet è pronto per esser classificato con il nodo SVM il quale riceve il feature vector in ingresso ed in uscita fornisce la classe di appartenenza. L'uscita del nodo SVM viene collegata ad un nodo column filter per eliminare tutte le colonne contenente i termini; L'uscita del column filter viene collegata in serie a due nodi join per visualizzare una tabella che ha le stesse colonne della tabella di partenza più un'ulteriore colonna relativa alla predizione della classe appena fatta. Questa tabella sarà l'ingresso del nodo Java Snippet e del nodo Csv writer; Il primo è un nodo custom in cui si è previsto di fare l'update della tabella HBase tramite Apache Phoenix, in quanto come è stato riscontrato in fase di sviluppo, il nodo previsto da KNIME Database Update non risulta compatibile con la sintassi di Phoenix, il secondo salva il risultato in formato CSV nel path inserito nel file di configurazione di Apache Storm. È possibile avviare un Flow Chart di Knime in batch, senza interfaccia grafica, direttamente da riga di comando. Per tale motivo è stato scritto uno script che accetta dei parametri quali il percorso del flow chart, il

nome della tabella HBase dei dati da classificare, il nome della tabella del training set, il percorso dove salvare i file di output.

4.8 Visualizzazione dei risultati

Per visualizzare i risultati ottenuti è stata realizzata una servlet che mostra i grafici relativi a tutte le tabelle. Vengono generati dei grafici a torta che mostrano in maniera rapida come sono partizionate le classi di una tabella. Per creare i grafici è stata utilizzata la libreria open source JFreeChart[41] che fornisce una serie di metodi per creare diversi tipi di grafici. Per interfacciarsi con HBase tramite Apache Phoenix si sono riscontrati problemi di conflitto di classi con JBoss e con Tomcat, per cui è stato necessario una fase di detection dei conflitti ed una fase di risoluzione dei medesimi, creando dunque una nuova libreria contenente tutte le classi necessarie per un corretto interfacciamento con HBase. Nella figura 4.6 viene riportato un esempio di visualizzazione di dati che sono stati classificati con l'applicazione sviluppata.

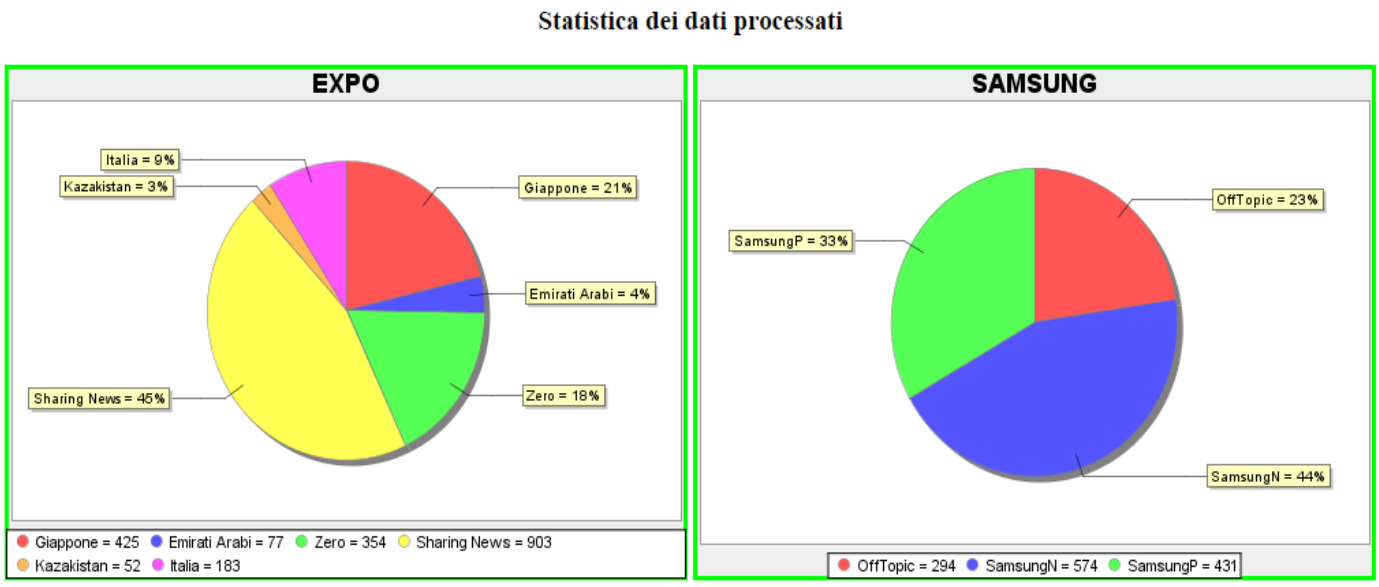


Figura 4.6: Visualizzazione dei risultati della classificazione

5. Risultati ottenuti e valutazione delle prestazioni

Questo capitolo illustra i risultati degli esperimenti finalizzati a valutare la classificazione e le prestazioni dell'applicazione sviluppata, ottenute in scenari reali. I test sono stati condotti su una macchina virtuale dotata di 8 Gb di ram, 50 Gb Hard Disk, processore 4700MQ con frequenza variabile da 1.4 GHz a 2.4 GHz. I test si sono svolti in due fasi: nella prima fase vengono valutate le prestazioni di Apache Storm prima e dopo il tuning dei parametri di deploy, nella seconda fase viene valutata l'accuratezza dei dati classificati con SVM, Naive Bayesian e alberi decisionali.

5.1 Valutazione delle performance di Apache Storm

Per valutare le performance di Apache Storm, si sono adottate come metriche di valutazione il throughput ed il tempo impiegato per processare e per persistere 100000 messaggi di testo lunghi 140 caratteri. Nel DataSpout ogniqualvolta che Storm invoca il metodo nextTuple() viene inviato il messaggio all'IgnoreWordsBolt, solo per la prima invocazione viene salvato un timestampche servirà in seguito per calcolare il tempo che ha impiegato Storm per processare i suddetti messaggi. Quando il PersistenceBolt riceverà e processerà l'ultimo messaggio viene calcolato il tempo in secondi. I risultati rilevati prima e dopo aver effettuato il tuning dei parametri vengono riportati nella tabella 10.

N°	Thread IgnoreWordsBolt	Thread NormalizerBolt	Thread PersistenceBolt	N°Messaggi	Tempo esecuz.	Throughput
1	1	1	1	100000	3min 29sec	480 t/sec
2	1	2	2	100000	1min 51sec	898 t/sec
3	1	4	4	100000	2min 42sec	619 t/sec
4	1	6	6	100000	Fail	-

Tabella 10:Prestazioni Storm .

È stato scelto di aumentare il numero dei thread dei Bolt che eseguono più operazioni ovvero il Bolt Normalizer che si occupa di effettuare il preprocessing del testo ed il Bolt Persistence che si occupa di persistere il messaggio e di avviare la classificazione. Non si è ritenuto necessario aumentare il numero dei Thread degli Spout in quanto, come dimostra la UI di Storm alla voce Execute Latency, il NormalizerBolt e il PersistenceBolt rappresentano il collo di bottiglia del sistema. Nel Test N°2, raddoppiando il numero di Thread dei Bolt che eseguono più operazioni, si è avuto un incremento medio del Throughput del 92%. Non si è scelto di aumentare il numero di Task per ogni Thread in quanto i Task vengono eseguiti in modo sequenziale e singolarmente da un Thread [10] quindi non aumenta il livello di parallelismo. Nel Test N°3, aumentando ulteriormente il numero dei Thread a quattro per il NormalizerBolt e a 4 per il PersistenceBolt, le prestazioni sono aumentate del 27% rispetto al primo test (Single Thread) e sono peggiorate del 31% rispetto al test N°1.La motivazione di tale decremento delle performance è dovuta, come dimostra Collectl, al fatto che la RAM si satura velocemente. Collectl è uno strumento per benchmark e monitoraggio delle prestazioni di un sistema Linux. È possibile scegliere di monitorare qualsiasi elemento di un vasto insieme di sottosistemi che, attualmente, include: buddyinfo, CPU, disco, inode, infiniband, lustre, memoria, rete, nfs, processi, quadrics, slab, socket e TCP[43]. Per visualizzare la memoria disponibile viene utilizzato il comando: collectl –sM, tale comando stampa a video ogni secondo la memoria Ram utilizzata/libera, e la memoria swap utilizzata/libera. Nel test N°4 si è provato ad aumentare il numero dei Thread totali a 12 di cui 6 per il Normalizer Bolt, 6 per il Persistence Bolt, come risultato finale si è ottenuto che Storm ha lanciato un’eccezione in quanto non è stato in grado di creare i Thread a causa della non disponibilità di memoria libera.

Adottando la configurazione del test N°2, si è valutato il tempo medio per elaborare ed emettere una tupla del Normalizer Bolt e del Persistence Bolt. Tale informazione ci viene fornita direttamente dalla UI di Storm, infatti selezionando i suddetti Bolt nell'interfaccia, sotto la voce executed latency vi è il tempo medio che una tupla trascorre all'interno del metodo execute. Nella tabella 11 viene riportato il tempo che impiega ogni Bolt ad elaborare una tupla

IgnoreWordsBolt	NormalizerBolt	PersistenceBolt
0.952ms	1.141ms	1.186ms

Tabella 11

Da tale tabella si evince che il NormalizerBolt ed il PersistenceBolt impiegano più tempo rispetto all'IgnoreWordsBolt. La configurazione adottata nel Test N°2 non può essere ulteriormente migliorata in quanto per migliorarla bisognerebbe aumentare contemporaneamente il numero dei Thread del Normalizer e del Persistence Bolt per non far sì che uno dei due diventi il nuovo collo di bottiglia, ma come è stato visto nel test N°4, aumentando il numero dei Thread Storm solleva l'eccezione `java.lang.OutOfMemoryError.`

5.2 Valutazione delle performance della classificazione e di KNIME

Nella figura 5.0 viene mostrato il flowchart di KNIME dove vengono messi a confronto tre algoritmi di classificazione.

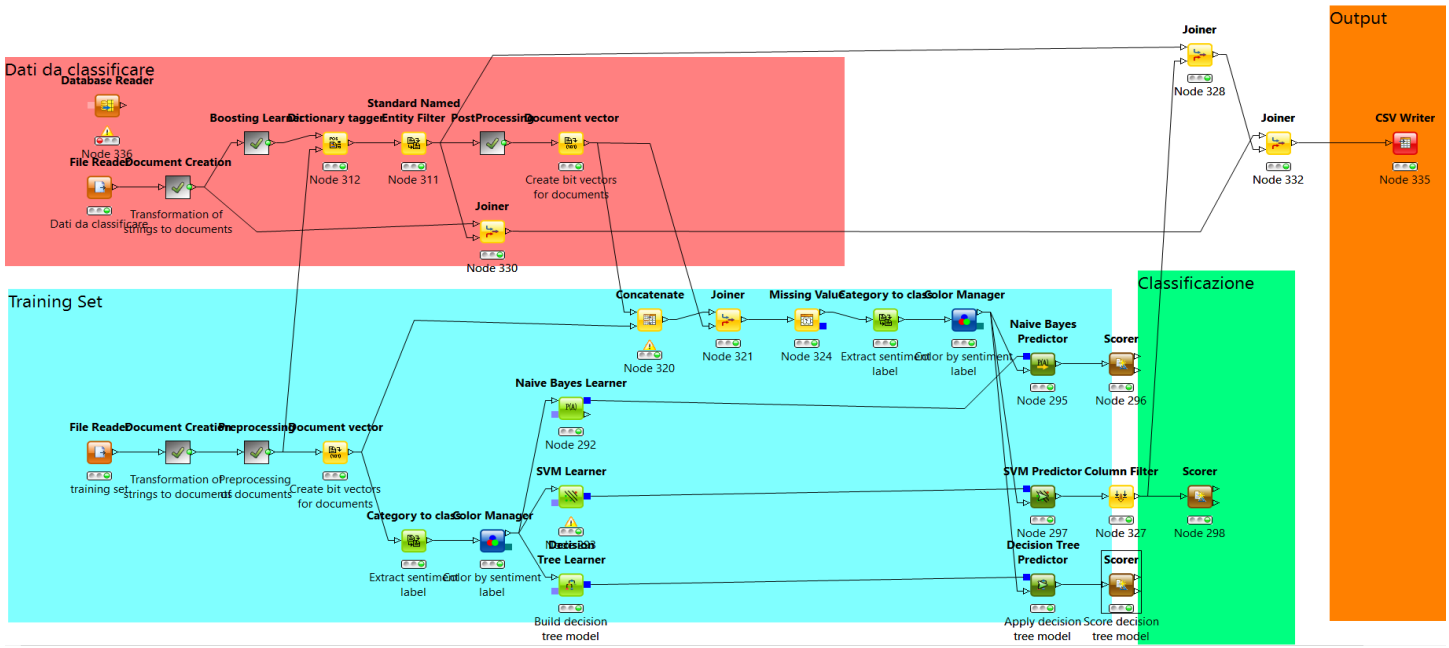


Figura 5.0: FlowChart di KNIME con i metodi di classificazione a confronto.

Per valutare l'accuratezza del SVM, Naive Bayesian e Alberi decisionali si è utilizzato un Dataset di X-Factor avente un Training set avente 12 classi di 400 istanze(circa 33 istanze per classe)ed un Test Set di 8708 istanze, entrambi forniti da TEIA Technologies.

KNIME mette a disposizione un nodo chiamato Scorer che, posto all'uscita di un classificatore, calcola il numero di istanze classificate correttamente, l'errore, l'accuratezza e infine permette di visualizzare la matrice di confusione; Tale matrice è una matrice di dimensione $k * k$, con k numero di classi nella quale sulle colonne si hanno il numero reale di records appartenenti a ciascuna classe e sulle righe il numero previsto di records appartenenti ad una data

classe. In questo modo i valori presenti sulla diagonale principale sono quelli che rappresentano il numero di casi classificati correttamente dall'algoritmo, mentre ogni valore fuori dalla diagonale principale rappresenta un errore di classificazione. Di seguito vengono riportate le tre matrici di confusione dei classificatori suddetti con le relative accuratezze. Nella figura 5.1 viene riportata la matrice di confusione del classificatore bayesiano

Confusion matrix - 0:296 - Scorer

File

Table "spec_name" - Rows: 12 | Spec - Columns: 12 | Properties | Flow Variables

Row ID	"Mara_P"	"Fedez e Mika_P"	"Elio_P"	"Mika_P"	"Skin_P"	"Neutral"	"Fedez_P"	"Fedez_N"	"Mika_N"	"Morgan_P"	"Elio_N"	"Skin_N"
"Mara_P"	1103	1	0	1215	1	2	0	3	0	0	0	0
"Fedez e Mi...	1	1865	1	1759	4	83	91	4	16	0	1	0
"Elio_P"	1	0	206	338	2	0	0	0	0	0	38	0
"Mika_P"	0	51	0	1356	1	106	0	1	29	0	1	0
"Skin_P"	0	0	0	0	41	0	0	0	0	0	0	0
"Neutral"	0	0	0	0	0	32	0	0	0	0	0	0
"Fedez_P"	0	0	0	37	0	4	57	0	0	0	0	0
"Fedez_N"	0	0	0	1	0	0	0	25	0	0	0	0
"Mika_N"	0	12	0	70	0	8	0	0	81	0	0	0
"Morgan_P"	0	0	0	1	0	0	0	0	0	19	0	0
"Elio_N"	0	0	0	0	0	0	0	0	0	0	20	0
"Skin_N"	0	0	0	0	0	0	0	0	0	0	0	20

Figura 4.1:Matrice di confusione utilizzando Naive Bayesian

Da tale matrice si evince che il classificatore sbaglia di molto quando deve classificare correttamente i tweet positivi relativi a Mika. L'accuratezza di tale algoritmo di classificazione è risultata essere la più bassa infatti come si evince dalla figura 5.2, vengono classificate correttamente 4825 istanze su 8708 con un'accuratezza dunque del 55%.

Accuracy statistics - 0:296 - Scorer

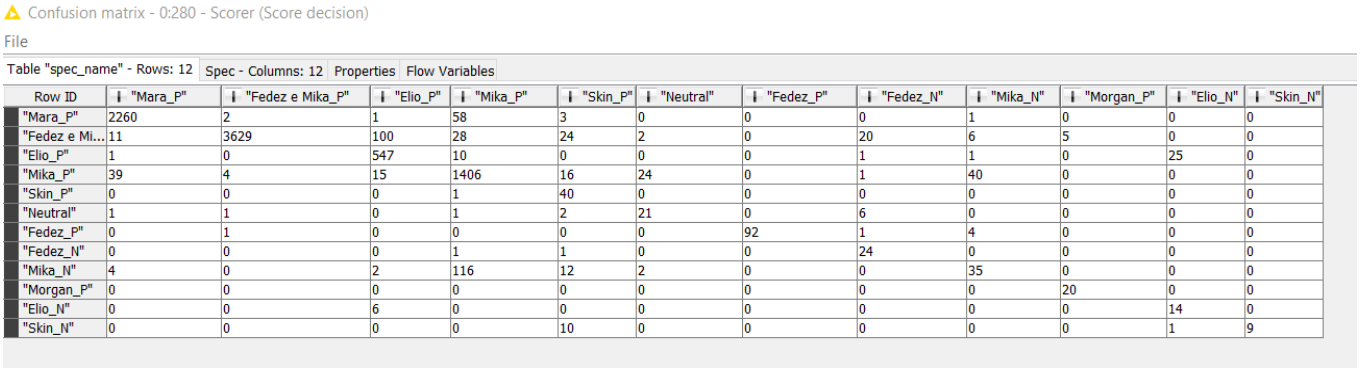
File

Table "default" - Rows: 13 | Spec - Columns: 11 | Properties | Flow Variables

I...	Own...	Name	Value
0	0:296	Cohen's kappa	0.420329896913579
0	0:296	#False	3883
0	0:296	#Correct	4825
0	0:296	Error	0.4459118052365641
0	0:296	Accuracy	0.554088194763436
0		knime.worksp...	C:\Users\Angelo\Desktop\KNIME_2.12.1\knime-w...

Figura 5.2: Accuratezza Naive Bayesian

Nella figura 5.3 viene mostrata la matrice di confusione degli alberi decisionali.


Confusion matrix - 0:280 - Scorer (Score decision)

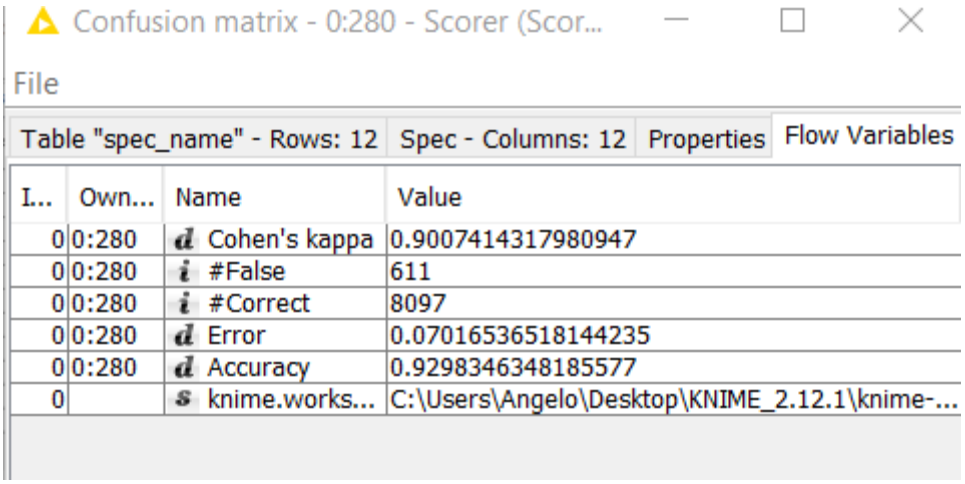
File

Table "spec_name" - Rows: 12 | Spec - Columns: 12 | Properties | Flow Variables

Row ID	"Mara_P"	"Fedez e Mika_P"	"Elio_P"	"Mika_P"	"Skin_P"	"Neutral"	"Fedez_P"	"Fedez_N"	"Mika_N"	"Morgan_P"	"Elio_N"	"Skin_N"
"Mara_P"	2260	2	1	58	3	0	0	0	1	0	0	0
"Fedez e Mi...	11	3629	100	28	24	2	0	20	6	5	0	0
"Elio_P"	1	0	547	10	0	0	0	1	1	0	25	0
"Mika_P"	39	4	15	1406	16	24	0	1	40	0	0	0
"Skin_P"	0	0	0	1	40	0	0	0	0	0	0	0
"Neutral"	1	1	0	1	2	21	0	6	0	0	0	0
"Fedez_P"	0	1	0	0	0	0	92	1	4	0	0	0
"Fedez_N"	0	0	0	1	1	0	0	24	0	0	0	0
"Mika_N"	4	0	2	116	12	2	0	0	35	0	0	0
"Morgan_P"	0	0	0	0	0	0	0	0	0	20	0	0
"Elio_N"	0	0	6	0	0	0	0	0	0	0	14	0
"Skin_N"	0	0	0	0	10	0	0	0	0	0	1	9

Figura 5.3: Matrice di confusione utilizzando gli Alberi decisionali

Tale classificatore riesce a classificare meglio i tweet positivi relativi a Mika rispetto al classificatore precedente. L'accuratezza, come viene mostrato nella figura 4.4 è del 93% classificando correttamente 8097 istanze su 8708 classificando in modo errato 611 messaggi.


Confusion matrix - 0:280 - Scorer (Scor...

File

Table "spec_name" - Rows: 12 | Spec - Columns: 12 | Properties | Flow Variables

I...	Own...	Name	Value
0	0:280	Cohen's kappa	0.9007414317980947
0	0:280	#False	611
0	0:280	#Correct	8097
0	0:280	Error	0.07016536518144235
0	0:280	Accuracy	0.9298346348185577
0		knime.works...	C:\Users\Angelo\Desktop\KNIME_2.12.1\knime-...

Figura 5.4: Accuratezza Alberi decisionali

Nella figura 5.5 viene mostrata la matrice di confusione del SVM

Confusion matrix - 0:298 - Scorer

File

Table "spec_name" - Rows: 12 | Spec - Columns: 12 | Properties | Flow Variables

Row ID	"Mara...	"Fedez e Mika_P"	"Elio_P"	"Mika_P"	"Skin_P"	"Neutral"	"Fedez_P"	"Fedez_N"	"Mika_N"	"Morgan_P"	"Elio_N"	"Skin_N"
"Mara_P"	2293	2	2	5	19	0	1	0	0	3	0	0
"Fedez e Mi...	8	3765	2	19	1	2	1	20	2	4	1	0
"Elio_P"	7	0	530	1	4	0	0	1	0	0	42	0
"Mika_P"	7	4	0	1456	6	23	0	1	43	5	0	0
"Skin_P"	0	0	0	0	41	0	0	0	0	0	0	0
"Neutral"	0	0	0	0	0	32	0	0	0	0	0	0
"Fedez_P"	0	0	0	0	0	0	94	0	0	0	0	4
"Fedez_N"	0	0	0	0	0	0	0	26	0	0	0	0
"Mika_N"	0	0	0	110	4	6	0	0	51	0	0	0
"Morgan_P"	0	0	0	0	0	0	0	0	0	20	0	0
"Elio_N"	0	0	0	0	0	0	0	0	0	0	20	0
"Skin_N"	0	0	0	0	0	0	0	0	0	0	0	20

Figura 5.5: Matrice di confusione SVM

Tale matrice di confusione nei test effettuati, è quella che ha un maggior numero di elementi sulla diagonale maggiore infatti, come si evince dalla figura 5.6 tale classificatore ha un’accuratezza del 96%.

Accuracy statistics - 0:298 - Scorer

File

Table "default" - Rows: 13 | Spec - Columns: 11 | Properties | Flow Variables

I...	Own...	Name	Value
0	0:298	Cohen's kappa	0.9410933780672498
0	0:298	#False	360
0	0:298	#Correct	8348
0	0:298	Error	0.04134129536058796
0	0:298	Accuracy	0.958658704639412
0		knime.works...	C:\Users\Angelo\Desktop\KNIME_2.12.1\knime-...

Figura 5.6: Accuratezza SVM

Per ogni test effettuato, si è riscontrato un aumento dell’accuratezza di circa il 12% utilizzando la tecnica dello stemming, ed un aumento dell’accuratezza di circa il 4% rimuovendo tutte le stop word.

Il modello di classificazione svolto, è stato confrontato con il classificatore Crimson Hexagon[44] e presi 130 posts categorizzati in due modi diversi, risulta che:

- 29 sono giusti per il classificatore realizzato.
- 18sono giusti per il classificatore Crimson
- 83 nessuno dei due classificatori ha classificato in modo corretto

KNIME per classificare il dataset di 8097 istanze ha impiegato 2min e 37 secondi riscontrando un utilizzo medio di 1.8Gb di Ram, e di circa il 73% di utilizzo della CPU.

Per migliorare le performance dell'applicazione sviluppata, si può creare un nuovo Bolt che per ogni parola, trova una lista di parole ordinate in modo decrescente in base ai sinonimi più frequentemente utilizzati in una determinata lingua; Se la parola per la quale viene creata la lista non si trova alla prima posizione della lista stessa (il caso in cui non è la parola più frequente) viene rimpiazzata con la prima entry della lista. Ad esempio data la parola "autoveicolo", viene creata una lista contenente: macchina, auto, automobile, autoveicolo. La parola presa in considerazione "autoveicolo" viene rimpiazzata nel tweet da "macchina". In questo modo si riduce il Bag Of Words del DataSet aumentando l'accuratezza della classificazione[46].

Nella figura 5.7 viene mostrata l'architettura dell'applicazione con l'aggiunta del suddetto Bolt.

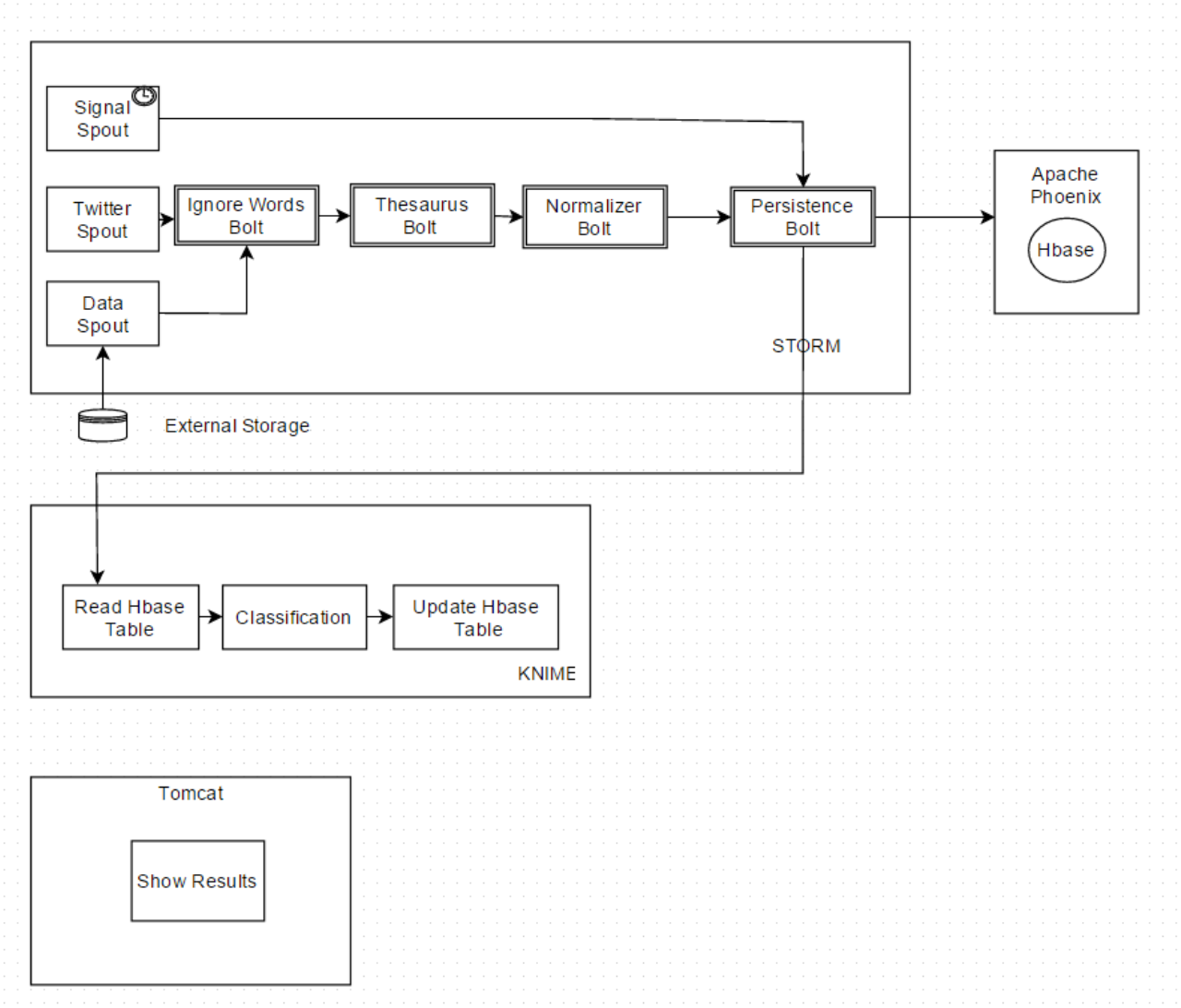


Figura 5.7: Architettura ottimizzata con il Bolt Thesaurus

Conclusioni

L’obiettivo di questo lavoro di tesi è stato quello di analizzare in tempo reale dati provenienti da fonti eterogenee, classificarli, renderli persistenti e visualizzarli in tempo reale attraverso dei grafici, al fine di costituire un valido strumento di supporto per i processi decisionali.

Attraverso il monitoraggio dei social network, le aziende e le organizzazioni interessate possono conoscere in tempo reale eventuali *minus* e *plus* dei propri prodotti, servizi, progetti o eventi, in quanto gli utenti hanno ormai l'abitudine di recensire on line e di condividere con altri utenti, mediante l’utilizzo di social, le proprie opinioni sulle cose che stanno più a cuore. Uno dei più mezzi più proficui di scambio è Twitter, sorgente generosa di dati pronti per esser analizzati.

In questo lavoro di tesi si sono analizzati diversi framework per il processamento di dati in tempo reale e per la persistenza di tali dati utilizzando Database NoSQL. Inoltre si sono analizzate diverse piattaforme per la classificazione dei dati e diverse tecniche di classificazione. Relativamente al processamento dei dati in tempo reale è stato utilizzato Apache Storm testando diverse configurazioni di deploy e scegliendo quella più performante in termini di throughput ed in termini di tempo impiegato per classificare un dataset di 100000 istanze.

Per quanto riguarda la persistenza dei dati si è utilizzato HBase attraverso Apache Phoenix. Relativamente alla classificazione dei dati è stato utilizzato KNIME, ed effettuando una serie di operazioni, si è utilizzato come algoritmo di classificazione il SVM in quanto come si evince nei test effettuati nel capitolo 6, è stato il classificatore con accuratezza maggiore.

È stato utilizzato un Dataset di X-Factor avente un Training set di 400 istanze ripartite in maniera equa su 12 classi (circa 33 istanze per classe) ed un Test Set

di 8708 istanze, entrambi forniti da TEIA Technologies riscontrando un'accuratezza del 96% classificando correttamente 8348 istanze e classificando in modo errato 360 istanze. Il classificatore sviluppato, è stato confrontato con la piattaforma di Web Reputation Crimson Hexagon e, come viene mostrato nel capitolo 6, utilizzando lo stesso dataset si è avuta un'accuratezza maggiore.

Per migliorare le performance dell'applicazione sviluppata, si può creare un nuovo Bolt che per ogni parola, trova una lista di parole ordinate in modo decrescente in base ai sinonimi più frequentemente utilizzati in una determinata lingua; Se la parola per la quale viene creata la lista non si trova alla prima posizione della lista stessa (il caso in cui non è la parola più frequente) viene rimpiazzata con la prima entry della lista. In questo modo si riduce il Bag Of Words del DataSet aumentando l'accuratezza della classificazione[46].

[1] CHANDARANA, Parth; VIJAYALAKSHMI, M. Big Data analytics frameworks. In: *Circuits, Systems, Communication and Information Technology Applications (CSCITA), 2014 International Conference on*. IEEE, 2013. p. 430-433.

[2]Rezzani,Big data. Architettura, tecnologie e metodi per l'utilizzo di grandi basi di dati

[3] Zikopoulos, Paul, and Chris Eaton. *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

[4] YIN, Shen; KAYNAK, Okyay. Big data for modern industry: challenges and trends. *Proc. IEEE*, 2014.

[5]https://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_fact_sheet_final.pdf

[6] MAYER-SCHÖNBERGER, Viktor; CUKIER, Kenneth. *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, 2012.

[7]http://storm.apache.org/

[8] ANTONIO, Murgia. Monitoring di Sistemi Eterogenei attraverso l'utilizzo di Tecnologie Linked e Semantic Data e di Apache Storm Realtime Processing System. 2014.

[9] YANG, Wenjie, et al. Big Data Real-Time Processing Based on Storm. In: *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*. IEEE, 2012. p. 1784-1787.

[10]Learning Storm:Create real-time stream processing applications with Apache Storm, Ankit Jain Anand Nalya

[11] LEIBIUSKY, Jonathan; EISBRUCH, Gabriel; SIMONASSI, Dario. *Getting started with storm*. " O'Reilly Media, Inc.", 2011.

[12]TOSHNIWAL, Ankit, et al. Storm@ twitter. In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2013. p. 147-155.

[13] ROY, STREAMING ANALYTICS WITH IBM STREAMS Analyze More, Act Faster, and Get Continuos Insights

[14] Popolarità di Spark, https://databricks.com/blog/2015/01/27/ big-data-projects-are-hungry-for-simpler-and-more-powerful-tools-survey-validates-apache-spark-is-gaining-developer-traction. html

[15] http://xinhstechblog.blogspot.in/2014/06/storm-vs-spark-streaming-side-by-side.html

[16] http://www.infoq.com/articles/stream-processing-hadoop

[17]NABI, Zubair, et al. Of Streams and Storms. *IBM White Paper*, 2013.

[18]http://www.ranks.nl/stopword

[19] Baeza Yates, Modern Information Retrieval, Addison-Wesley Longman Publishing Co., 1999,

[20] Micheal J. A. Berry Gordon S.Linoff, Data Mining L'azienda intelligente e la gestione strategica delle informazioni.

[21] https://github.com/optimaize/language-detector

[22] Burges, C.J.C. (1998), "A tutorial on support vector machines for pattern recognition," Data Mining and Knowledge Discovery, vol. 2

[23]Willett, Peter. "The Porter stemming algorithm: then and now." Program 40.3 (2006).
[24] C. Silva, and B. Ribeiro. The importance of stop word removal on recall values in text categorization. Neural Networks, 2003
[25] Aiyer, Amitanand S., et al. "Storage Infrastructure Behind Facebook Messages: Using HBase at Scale." IEEE Data Eng. Bull. 34.2 (2012): 4-12.
[26] Han, Jing, et al. "Survey on NoSQL database." Pervasive computing and applications (ICPCA), 2011 6th international conference on. IEEE, 2011.
[27] Pritchett, Dan. "Base: An acid alternative." Queue 5.3 (2008): 48-54.
[28] Tauro, Clarence JM, Baswanth Rao Patil, and K. R. Prashanth. "A Comparative Analysis of Different NoSQL Databases on Data Model, Query Model and Replication Model." Proceedings of the International Conference on Emerging Research in Computing, Information, Communication and Applications ERCICA. 2012.
[29] Tudorica, Bogdan George, and Cristian Bucur. "A comparison between several NoSQL databases with comments and notes." Roedunet International Conference (RoEduNet), 2011 10th. IEEE, 2011.
[30]https://dev.twitter.com/streaming/overview
[31] The Cassandra Distribuited Database - Eric Evans - February 7, 2010.
[32] https://docs.mongodb.org/manual/
[33]Pak, Alexander, and Patrick Paroubek. "Twitter as a Corpus for Sentiment Analysis and Opinion Mining." *LREC*. Vol. 10. 2010.
[34] http://sbrinz.di.unipi.it/peppe/MaterialeCorsi/CorsoJavaCDC/12_1EspressioniRegolari.pdf
[35] https://phoenix.apache.org/performance.html
[36]Phan, Xuan-Hieu, Le-Minh Nguyen, and Susumu Horiguchi. "Learning to classify short and sparse text & web with hidden topics from large-scale data collections." *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008.
[37]Aghdam, Mehdi Hosseinzadeh, Nasser Ghasem-Aghaee, and Mohammad Ehsan Basiri. "Text feature selection using ant colony optimization." *Expert systems with applications* 35.3 (2009): 6843-6852.
[38]http://bias.csr.unibo.it/golfarelli//DataMining/MaterialeDidattico/DMISI-Classificazione%20-%20Bayes.pdf
[38]http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/tdm/tdm_dm_16_aprile_07.pdf
[39]https://tech.knime.org/documentation
[40]http://uweb.deis.unical.it/tagarelli/wp-content/uploads/2013/12/Weka-Introduzione-e-Preprocessing.pdf
[41]http://www.jfree.org/jfreechart/
[42]Ramos, Juan. "Using tf-idf to determine word relevance in document queries."*Proceedings of the first instructional conference on machine learning*. 2002.
[43] https://packages.debian.org/it/sid/collectl
[44] http://www.crimsonhexagon.com/platform/
[45] http://www.malinga.me/reading-and-understanding-the-storm-ui-storm-ui-explained/
[46]Bollegala, Danushka, David Weir, and John Carroll. "Cross-domain sentiment classification using a sentiment sensitive thesaurus." *Knowledge and Data Engineering, IEEE Transactions on* 24.8 (2013): 1719-1731.