Università degli studi di Trento

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea triennale in INFORMATICA

Tesi Finale

Clustrify: clustering non supervisionato di tweets

Relatore
Prof. Alberto Montresor

Laureando
Mattia Larentis

Anno accademico 2013-2014

Dedicata a Michela

Contenuti

Prefazione

> "È evidente che l'uomo sia un essere sociale più di ogni ape e più di ogni animale da gregge. Infatti, la natura non fa nulla, come diciamo, senza uno scopo: l'uomo è l'unico degli esseri viventi a possedere la parola."
>
> — Aristotele, Politica

L'uomo, in quanto essere sociale, per natura è portato al bisogno di comunicare. Dall'invenzione del telegrafo all'utilizzo di Internet è cambiato solamente il mezzo di divulgazione, ma non l'atto di voler condividere pensieri con il resto dell'umanità.

Questi pensieri vengono talvolta espressi sotto forma di testi, di dimensione variabile, che un individuo scrive nella speranza che da altri vengano letti. La grande mole di messaggi può essere elaborata per ricavarne informazioni utili, potenzialmente per la risoluzione di alcuni problemi.

Clusterify è un'applicazione web che mira alla possibilità di risolvere più problemi contemporaneamente: dividere un ammasso di tweets in insiemi distinti, dove ognuno è caratterizzato da un forte legame che le proprie componenti hanno tra di loro, e da un legame debole nei confronti delle altre. Questo processo può aprire le porte a molteplici risultati, come la possibilità di filtrare testi per una macro-area interessata, piuttosto che caratterizzare persone interessate ad un'attività per capire come meglio investire nel prossimo semestre.

Il mio progetto ha l'intento di mostrare la potenzialità del clustering non supervisionato per la suddivisione di testi. Nella fattispecie si ha intenzione di applicare degli algoritmi di clustering per definire questi sottogruppi ed, una volta composti, utilizzarli come strumento per filtrare gli stessi messaggi. Supponendo che l'utente medio di Twitter non parli solamente dell'argomento per cui è stato seguito, è possibile che questo scriva di materie al nostro utente interessanti o, al contrario, ininfluenti.

Clusterify, così facendo, permette di leggere solamente tweets per lui interessanti.

Chapter 1

Background

1.1 Twitter API

Twitter Inc. è un social network che permette agli utenti di scrivere e di leggere *micropost*, messaggi lunghi al più 140 caratteri, comunemente chiamati *tweets*.

Questa piattaforma, creata nel 2006 da Jack Dorsey, Evan Williams, Biz Stone e Noah Glass, ha spopolato fino a raggiungere nel 2013 più di 200 milioni di utenti attivi e più di 400 milioni di tweets al giorno [1].

Ad oggi, Twitter, è l'undicesimo sito internet visitato ogni giorno [2].

Twitter possiede, come la maggior parte dei social network, un servizio interrogabile via API per ottenere e per caricare dati.

Nel primo caso si possono richiedere i tweets che compongono la *timeline* di un utente, i messaggi privati che due utenti si scambiano oppure alcuni dati relativi all'account di un utente come nome e cognome, nickname, immagine profilo e via dicendo.

Nel secondo caso, invece, si può scrivere un nuovo tweet oppure modificare dati che Twitter ha salvato nel propri databases.

In entrambi i casi serve che l'utente interessato *firmi* un consenso che permette al programmatore di interagire con i propri dati; una volta permesso, sarà possibile agire come l'utente stesso.

1.1.1 REST API v1.1

Il modo per interagire con i dati di Twitter è appunto la REST API.

Con REST (Representational State Transfer) si indica un tipo di architettura software basato sull'idea di utilizzare la comunicazione tra macchine per mezzo di richieste HTTP.

Le applicazioni basate su questo tipo di architettura vengono nominate RESTful ed utilizzano, appunto, HTTP per tutte le operazioni di *CRUD*: Create, Read, Update e Remove.

Twitter mette a disposizione, nella documentazione per i programmatori [3], una lunga lista di possibili richieste – `https://dev.twitter.com/docs/api/1.1` – e per ognuna di queste, una pagina dedicata dove, dopo una breve descrizione, vengono elencati i dati richiesti in input, i possibili filtri applicabili, la struttura dell'output che questa richiesta genererà ed, infine, un esempio richiesta/risposta per chiarire le idee.

1.1.2 Autenticazione e autorizzazione: OAuth

OAuth, il protocollo di autenticazione che utilizza Twitter [4], permette agli utenti di approvare un'applicazione che agisca al loro posto, senza il bisogno di condividere password. Gli sviluppatori possono così pubblicare e interagire con dati protetti, come ad esempio i tweets; i service provider, al contempo, proteggono le credenziali dei loro utenti [5].

1.2 dataTXT

DataTXT è un insieme di API semantiche sviluppate da SpazioDati S.r.l. che mirano a estrarre significato da testi scritti in diverse lingue. Questo prodotto si diversifica dagli altri dello stesso genere in quanto è stato pensato e ottimizzato per lavorare su testi molto corti, come i tweets.

DataTXT, essendo appunto una famiglia di strumenti, esegue molti compiti: può ad esempio estrarre da un testo delle *entità* oppure categorizzare dei documenti in categorie stabilite dall'utente stesso.

1.2.1 Grafo di dataTXT

DataTXT per poter eseguire i suoi algoritmi ha bisogno di basarsi su di un grafo pre-calcolato; questo ha $n + r$ nodi, dove n sono gli snippet $S = \{S_1, ..., S_n\}$ e r sono i topic $T = \{t_1, ..., t_r\}$.

Dato uno snippet s e un topic t, denotiamo come $p(s, t)$ lo score che dataTXT assegna all'annotazione di s con il topic t. Questo valore rappresenta l'importanza che un topic t ha su di uno snippet s e viene utilizzato come valore per l'arco da va da s a t nel grafo stesso.

Dati due topic t_a e t_b, per esempio due pagine di Wikipedia, si può misurare la loro *relatedness* $rel(t_a, t_b)$ utilizzano la seguente funzione che si basta sul numero di citazioni e co-citazioni delle due pagine di Wikipedia:

$$rel(t_a, t_b) = \frac{log(|in(t_a)|) - log(|in(t_a) \cap in(t_b)|)}{log(W) - log(|in(t_b)|)}$$

dove $in(t)$ è il set di archi entranti nella pagina t e W è il numero totale delle pagine di Wikipedia [6].

1.2.2 dataTXT NEX

DataTXT NEX è un servizio che permette di fare *Named Entity eXtraction* da un testo.

1.2.2.1 Named entity extraction

Named entity extraction, anche conosciuto come Named entity recognition, è un processo che cerca di localizzare e, successivamente, di classificare elementi presenti in un testo in categorie predefinite, come ad esempio persone, organizzazioni, luoghi, espressioni di tempo, ecc [7].

1.2.2.2 Esempio di richiesta

```
https://api.dandelion.eu/datatxt/nex/v1/?lang=en&text=The%20doctor%
20says%20an%20apple%20is%20better%20than%20an%20orange&include=type,
abstract,categories,lod&$app_id=YOUR_APP_ID&$app_key=YOUR_APP_KEY
```

1.2.2.3 Esempio di risposta

Nell'*header* sono presenti molte informazioni utili quali il numero di crediti utilizzati, quelli rimanente e la data in cui questi torneranno al valore iniziale.

```
Connection: keep-alive
Content-Length: 2748
Content-Type: application/json;charset=UTF-8
Date: Wed, 21 Oct 2015 16:29:37 GMT
Server: Apache-Coyote/1.1
X-DL-units: 1
X-DL-units-left: 999
X-DL-units-reset: 2015-10-22 00:00:00 +0000
```

Nel *body*, invece, sono presenti i dati veri e propri. *Annotations* è la lista composta dagli elementi trovati, con vari dettagli; questi elementi sono filtrati da alcuni parametri che possono essere passati nella richiesta, ad esempio la *min_confidence*.

```
{
  "timestamp": "2015-10-21T16:29:37",
  "time": 2,
  "lang": "en",
  "annotations": [{
      "...": "...",
  }, {
    "abstract": "The apple is the fruit of the apple tree...",
    "id": 18978754,
    "title": "Apple",
```

```
    "start": 19,
    "categories": [
      "Apples",
      "Malus",
      "Plants described in 1803",
      "Sequenced genomes"
    ],
    "lod": {
      "wikipedia": "http://en.wikipedia.org/wiki/Apple",
      "dbpedia": "http://dbpedia.org/resource/Apple"
    },
    "label": "Apple",
    "types": [
      "http://dbpedia.org/ontology/Eukaryote",
      "http://dbpedia.org/ontology/Plant",
      "http://dbpedia.org/ontology/Species"
    ],
    "confidence": 0.7869,
    "uri": "http://en.wikipedia.org/wiki/Apple",
    "end": 24,
    "spot": "apple"
  }, {
    "...": "...",
  }]
}
```


1.2.3 dataTXT REL

DataTXT REL è un servizio che, ad oggi, non è ancora presente sul mercato, ma che permetterà di capire quando un topic t_a è correlato ad un topic t_b, ricavando questo valore dalla formula utilizzata per la creazione degli archi pesati all'interno del grafo di dataTXT.

1.2.4 dataTXT *

DataTXT è un progetto vivo ed in continua evoluzione, con l'uscita periodica di nuove funzionalità; su `https://dandelion.eu/products/datatxt/` si possono ottenere informazioni circa le novità.

Chapter 2

Clusters e clustering

Il *clustering* è una tecnica di raggruppamento di oggetti in insiemi, secondo determinate regole, con l'intento che gli appartenenti allo stesso *cluster* siano più simili di quelli contenuti negli altri gruppi.

Queste regole tendono a minimizzare la *distanza* interna a ciascun gruppo ed a massimizzare quella tra i gruppi stessi: questa distanza viene quantificata per mezzo di misure di similarità.

La distanza è una qualsiasi funzione $d : X \times X \to \mathbb{R}$ che soddisfa:

$$
\begin{aligned}
&d(x, y) \geq 0 \\
&d(x, y) = 0 \iff x = y \\
&d(x, y) = d(y, x) \\
&d(x, y) \leq d(x, z) + d(z, y)
\end{aligned}
$$

Non esiste un solo algoritmo per raggruppare in modo corretto, bensì un solo compito da portare a termine. Questo obiettivo può essere raggiunto in più modi, basandosi su qualsivoglia tipo di nozione col solo fine di raggruppare in modo efficiente e sensato gli oggetti dati.

Esistono molteplici nozioni a cui ci si può ispirare per l'atto del raggruppamento; la creazione di gruppi caratterizzati da una piccola distanza tra i propri membri, piuttosto che l'utilizzo di particolari distribuzioni statistiche.

Seppur non esista non solo modo per fare clustering, molti di questi algoritmi godono di alcuni tratti in comune che definiscono i così detti *modelli*.

I modelli di clustering tipici sono:

- Clustering partizionale
- Clustering gerarchico
- Clustering density-based

2.1 Clustering partizionale

Gli algoritmi di clustering di questa famiglia creano una partizione delle osservazioni minimizzando la seguente funzione di costo:

$$\sum_{j=1}^{k} E(C_j)$$

ove k è il numero dei cluster richiesti in output, C_j è il j-esimo cluster e $E : C \to R^+$ è la funzione di costo associata al singolo cluster.

Questa tipologia di algoritmi solitamente richiede all'utente di specificare k, il numero di cluster distinti che si vogliono raggiungere a processo terminato, e mira ad identificare i gruppi naturali presenti nel dataset, generando una partizione composta da cluster disgiunti la cui unione ritorna il dataset originale.

2.1.1 Algoritmi conosciuti

Gli algoritmi più famosi appartenenti questa categoria sono:

- k-means
- k-medoids
- CLARANS

2.2 Clustering gerarchico

Gli algoritmi di clustering gerarchico, invece, creano una rappresentazione gerarchia ad albero dei cluster. Le strategie per il clustering gerarchico sono tipicamente di due tipi:

- Agglomerativo
- Divisivo

2.2.1 Metodo agglomerativo

Il metodo agglomerativo segue un approccio *bottom up* al problema dove, inizialmente, si ha un cluster per ogni oggetto e, successivamente, si procede all'unione di questi cluster, basando la selezione dei cluster da unire ad una *funzione di similarità*.

2.2.2 Metodo divisivo

Il metodo divisivo, invece, segue un approccio *top down* al problema dove, inizialmente, si ha un unico cluster contente tutti gli oggetti e, via via, viene suddiviso in più sotto-cluster, basando la selezione del cluster da dividere ad una *funzione di similarità*. Solitamente si impone un numero minimo di elementi che ogni cluster deve contenere alla fine del processo.

2.2.3 Dissimilarità tra cluster

Nella maggior parte dei metodi di clustering gerarchico si fa uso di metriche specifiche che quantificano la distanza tra coppie di elementi e di un criterio di collegamento che specifica la dissimilarità di due insiemi di elementi (cluster) come funzione della distanza a coppie tra elementi nei due insiemi.

2.2.4 Metriche

La scelta della metrica influenza la forma dei cluster, poiché alcuni elementi possono essere più vicini utilizzando una data distanza e più lontani utilizzandone un'altra.

Le metriche comuni sono le seguenti:

- Distanza euclidea
- Distanza di Manhattan

2.2.5 Criteri di collegamento

Il criterio di collegamento specifica la distanza tra insiemi di elementi come funzione di distanze tra gli elementi negli insiemi. I criteri di collegamento comuni sono i seguenti:

- Complete linkage: calcola la distanza tra i due cluster come la distanza massima tra elementi appartenenti ai due clusters
- Minimum o single-linkage: calcola la distanza tra i due cluster come la distanza minima tra elementi appartenenti a cluster diversi
- Average linkage: calcola la distanza tra i due cluster come la media delle distanze tra i singoli elementi

2.2.6 Algoritmi conosciuti

Gli algoritmi più famosi appartenenti questa categoria sono:

- SLINK *(single-linkage)*
- CLINK *(complete-linkage)*

2.3 Clustering density-based

Negli algoritmi di clustering density-based il raggruppamento avviene analizzando l'intorno di ogni punto dello spazio, connettendo regioni di punti con densità sufficientemente alta.

2.3.1 Algoritmi conosciuti

Gli algoritmi più famosi appartenenti questa categoria sono:

- DBSCAN

2.4 Clustering distribution-based

Questa tipologia di algoritmi è quella più vicina alla statistica. I cluster possono essere definiti come l'insieme di oggetti che appartengono, probabilmente, alla stessa distribuzione.

Anche se dal punto teorico questi modelli sono eccellenti, soffrono di un problema chiave chiamato *overfitting*.

2.4.1 Overfitting

Si parla di overfitting quando un modello statistico si adatta ai dati osservati utilizzando un numero eccessivo di parametri. Un modello sbagliato può adattarsi perfettamente se è abbastanza complesso rispetto alla quantità di dati disponibili.

Questo concetto è presente anche nell'apprendimento automatico e nel data mining. Un algoritmo che prevedere apprendimento viene allenato usando un training set e si assume che quest'algoritmo raggiungerà uno stato in cui sarà in grado di prevedere gli output per tutti gli altri esempi. Tuttavia, è possibile che un apprendimento svolto in modo sbagliato, il modello potrebbe adattarsi a caratteristiche che sono specifiche del training set, ma che non hanno riscontro nel resto dei casi.

2.4.2 Algoritmi conosciuti

Gli algoritmi più famosi appartenenti questa categoria sono:

- Expectation–maximization

Chapter 3

Clusterify

Clusterify è un'applicazione web che offre agli utenti la possibilità di leggere i propri tweet suddivisi in insiemi definiti dinamicamente e in modo non supervisionato da un algoritmo di clustering precedentemente scelto.

Clusterify, come tutte le applicazioni web, è composta da due parti: *frontend* e *backend*.

3.1 Frontend

Frontend

3.2 Backend

Clusterify è un'applicazione Django [8] costruita con l'idea di essere pluggabile, ovvero con la possibilità di inserire, con poco lavoro, nuovi *reader* e nuovi algoritmi di clustering. Si può vedere il tutto come una struttura a *Lego* dove si possono cambiare dei blocchi in modo da ottenere il risultato migliore, senza necessariamente modificare il codice del *core*. Una volta stabiliti questi blocchi, clusterify farà il resto per calcolare i clusters dai dati dati in input.

3.2.1 Reader

Un reader è un oggetto che permette di caricare all'interno del workflow un insieme di testi da analizzare.

Per mezzo di questo oggetto è possibile leggere dati da qualsiasi fonte, che sia un social network, come nel nostro caso, da un file in memoria, piuttosto che da un database, implementando solamente il metodo $\_texts()$.

Questo metodo deve ritornare una lista di ennuple; ognuna di queste deve contenere testo, url (se esiste) e l'autore (se esiste).

3.2.1.1 TwitterReader

TwitterReader, come dice il nome, è il reader che è stato implementato per poter ottenere i dati da Twitter; $\_texts()$, in questo caso, fa una richiesta a Twitter e, una volta ricevuti, formalizza i dati.

3.2.2 Algoritmi di clustering

Un algoritmo di clustering, come visto in precedenza, è un algoritmo che permette di raggruppare oggetti all'interno di un insieme, mantenendo uniti quelli che secondo alcune regole risultano più coesi.

Come per il *reader*, anche questi algoritmi possono essere estesi con nuovi codici che implementano regole diverse, in modo da poter soddisfare qualsiasi richiesta. Clusterify nasce con tre algoritmi di clutering nel core, utili per valutare quale performava meglio nella suddivisione di topic definiti utilizzando dataTXT.

3.2.2.1 K-Means

K-means è il nome di un algoritmo di clustering che punta a creare k gruppi distinti di uguale varianza, minimizzando l'inerzia del gruppo. Questo algoritmo richiede che il numero di cluster sia definito a priori.

L'algoritmo punta a scegliere k centroidi C che minimizzano lo scarto quadratico medio con un dataset X di n elementi $(x_1, ..., x_n)$, creando k insiemi $S = S_1, ..., S_K$:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} ||x_j - \mu_i||^2$$

L'algoritmo in se è composto da tre passi:

1. Scelta dei centroidi iniziali;
2. Assegnamento di ogni componente;
3. Creazione di nuovi centroidi;

Dopo aver scelto i primi k centroidi dalle componenti del dataset X, l'algoritmo continua ad eseguire due operazioni finché i centroidi proposti possono essere considerati corretti, ovvero che la propria posizione non venga modificata con il passare delle iterazioni [9].

L'assegnamento di ogni componente ad un centroide si esegue utilizzando la distanza euclidea al quadrato, questa sarà assegnata al punto medio più vicino; si può calcolare la distribuzione delle componenti in questo modo:

$$S_i^{(t)} = \left\{ x_p : ||x_p - m_i^{(t)}||^2 \leq ||x_p - m_j^{(t)}||^2 \ \forall j, 1 \leq j \leq k \right\}$$

dove ogni x_p è assegnato a esattamente un $S^{(t)}$, anche se idealmente potrebbe essere assegnato a più di un centroide.

La creazione dei nuovi centroidi utilizza, invece, il valore medio di ogni componente appartenente ad ogni centroide precedentemente definito.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

Per quanto riguarda la scelta dei centroidi ci si può basare sul caso, scegliendo in modo *random* i punti, con l'idea che con il passare delle iterazioni queste stime si sistemeranno; la seconda scelta vede il posizionamento dei centroidi in modo equidistante dagli tra loro, col fine di cercare di massimizzare l'area coperta e minimizzare il numero di iterazioni necessarie per giungere nella situazione finale di stallo [10].

3.2.2.2 Spectral

spectral

3.2.2.3 Affinity Propagation

Affinity Propagation è il nome di un algoritmo di clustering basato sul concetto di "passaggio di messaggi" tra le varie componenti. Una caratteristica fondamentale di quest'algoritmo è la mancanza di necessità di determinare a priori la quantità di cluster che saranno necessari [11].

Un dataset è descritto utilizzando degli esemplari identificati come le componenti più rappresentative dell'insieme. I messaggi trasmessi tra coppie sono utilizzati per calcolare l'idoneità che una componente ha di essere l'esemplare dell'altra, la quale sarà successivamente aggiornata in risposta al valore delle altre coppie. Questo processo viene ripetuto iterativamente finché il tutto non converge, dando vita al cluster [12].

Sia $(x_1, ..., x_n)$ l'insieme dei punti da clusterizzare e sia S una funzione che permette di definire la similarità di due punti.

L'algoritmo procede alternando due passi di "passaggio di messaggi", aggiornando due matrici:

- La matrice di responsabilità R ha valori $r(i, k)$ che quantificano quanto x_k sia ben situato nel cluster contenente x_i, prendendo in considerazione tutti gli altri elementi contenuti in questo insieme;
- La matrice di disponibilità A contiene i valori $a(i, k)$ rappresentanti quanto sia appropriato per x_i scegliere x_k come prossimo elemento del cluster a cui lui stesso appartiene.

Entrambe queste matrici sono inizializzate a 0 e, successivamente, saranno aggiornate in modo iterativo con le seguenti espressioni:

$$r(i,k) = s(i,k) - \max_{k' \neq k} \left\{ a(i,k') + s(i,k') \right\}$$

$$a(i,k) = \min \left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k)) \right) for i \neq k$$

$$a(k,k) = \sum_{i' \neq k} \max(0, r(i',k))$$

Il processo terminerà quando il tutto convergerà [13].

3.3 Scelta dell'algoritmo di clustering

Ho scelto di usare {} perché...

3.4 Workflow

Clusterify non è altro che un sistema di *pipe*, dove un comando viene eseguito prendendo come input l'output del precedente.

Come si è visto nel sezione relativa al *processo di caricamento* nel capitolo *frontend*, il workflow si può suddividere in tre parti:

1. Acquisizione dei testi
2. Annotazione dei testi
3. Clustering dei testi

3.4.1 Acquisizione dei testi

La parte di acquisizione testi, come detto in *reader* è una delle due parti pluggabili di Clusterify.

Questa fase si occupa del prendere i dati da una qualsivoglia fonte e di salvarli in modo conferme allo standard creato. All'interno di questa parte è possibile implementare, se necessario, un livello di cache in modo da impedire di richiedere più volte le stesse informazioni.

In Clusterify, *TwitterReader* utilizza *Twython*, una libreria Python che offre un accesso facile ai dati presenti su Twitter [14]. In altre parole, Twython offre un *wrapper* alle API di Twitter, permettendo al programmatore di ottenere dati senza preoccuparsi della gestione errori, dell'aggiornamento delle API e quant'altro.

3.4.2 Annotazioni dei testi

La seconda fase del workflow si occupa di annotare i testi per mezzo di dataTXT.

Quest'operazione è fondamentale per un buon risultato finale in quanto questa può far incorrere nel riconoscimento di entità sbagliate; oppure, nel caso contrario, ci si può imbattere nell'estrazione di poche entità, ma molto precise. Avere troppe annotazioni sbagliate possono condurre nell'avere dei dati non veritieri; il contrario potrebbe portare a non avere nemmeno un entità estratta per testo analizzato e quindi, in fase di clustering, questi verrebbero persi.

Al programmatore è quindi chiesto di scegliere molto attentamente i parametri che verranno passati all'estrattore, per non finire in uno dei due casi.

3.4.3 Clustering dei testi

L'ultima fase la si può pensare suddivisa a sua volta in tre sotto fasi:

1. Creazione matrice delle adiacenze
2. Esecuzione dell'algoritmo di clustering
3. Uniformazione output

3.4.3.1 Creazione matrice di adiacenza

La maggior parte degli algoritmi di clustering necessita di un grafo pesato su cui operare. Questo è rappresentato per mezzo di una matrice delle adiacenze $N \times N$, ove N è la cardinalità dell'insieme composto da tutte le entità, e viene costruita in questo modo:

$$Adj(a,b) = \begin{cases} rel(a,b), & \text{se } a \neq b \\ 0, & altrimenti \end{cases}$$

ove Adj è la nostra matrice delle adiacenze, a e b sono i due topic e $rel(a,b)$ è la chiamata all'API dataTXT-REL che permette di capire quanto due entità sono correlate tra di loro.

Si deduce che Adj è una matrice speculare, con valori reali compresi tra 0 e 1, e che presenta 0 sulla diagonale.

3.4.3.2 Esecuzione dell'algoritmo di clustering

Una volta generata la matrice delle adiacenze si può eseguire un algoritmo di clustering. Come già detto Clusterify permette di estendere gli algoritmi già presenti, quali *Star*, *Spectral*, *Affinity Propagation*. Una volta stabilito

l'algoritmo da utilizzare, Cluterify passerà a questo i vari dati e aspetterà l'esecuzione e il successivo output.

3.4.3.3 Uniformazione output

Dato l'output dell'algoritmo di clustering, i dati saranno processati nuovamente attraverso una funzione votata a due principali funzioni:

- Rendere l'output conforme con il modello prestabilito
- Aggiungere informazioni ad ogni componente del cluster

La prima si occuperà di portare la struttura dati uscente dall'algoritmo in quella stabilita da Clusterify; la seconda, invece, permette di aumentare l'informazione che questa contiene, aiutandosi con dati calcolati in precedenza o sul posto.

Conclusioni e sviluppi futuri

Clusterify è un prodotto pronto all'uso e già utilizzato da SpazioDati per pubblicizzare uno dei molteplici casi in cui dataTXT può essere utilizzato. Nato dall'idea di mostrare come la forza del clustering può essere utilizzata anche su testi molto corti, è stata soggetto di molti studi nel team di SpazioDati per poter creare un ulteriore prodotto vendibile al pubblico pagante.

Clusterify nasce però come *proof of concept* per mostrare queste potenzialità. In futuro potrebbe essere esteso con l'implementazione di un *workflow* totalmente dinamico, dove il processo di caricamento, annotazione e clustering dei testi possa essere di tipo incrementale. In questo scenario l'utente potrebbe abbandonare l'interfaccia tipica offerta da Twitter per avere il proprio stream aggiornato in tempo reale su Clusterify.

Questa composizione presenta tutti gli aspetti legati al *core* di Clusterify, i suoi possibili miglioramenti e le sue limitazioni. Si può anche affermare che è comunque un buon punto di partenza per nuovi programmatori che vorrebbero contribuire al progetto, portando nuove idee e nuova forza.

Penso che Clusterify sia uno strumento ben pensato per essere successivamente esteso e migliorato e che sia molto importante per i futuri clienti di SpazioDati; spero, infine, che questo progetto riesca a svilupparsi nel futuro prossimo.

Ringraziamenti

Anche se appare solamente il mio nome sulla copertina di questa tesi, questa non sarebbe mai esistita senza l'aiuto di molte persone: è un piacere e un dovere render loro grazie.

Ringrazio il mio supervisore Alberto Montresor per il suo incoraggiamento e il supporto fornitomi durante la stesura di questa composizione.

Un ringraziamento speciale va riservato a Stefano Parmesan ed a Federico Vaggi, i quali hanno prestato molta pazienza e dedizione nel seguirmi ed impegno nell'aiutarmi.

Ho acquisito molte conoscenze circa lo sviluppo web durante le mie attività lavorative e alla passione che i miei colleghi sono stati in grado di trasmettermi; un grazie al team di SpazioDati e ai trenta3dev per il loro appoggio e per i bei momenti vissuti con loro.

Ringrazio mamma e papà in quanto mamma e papà.

Un grazie è tutto per Michela e per Carlotta che mi hanno spinto a dare del mio meglio anche quest'anno.

Infine vorrei ringraziare Michele Pittoni e tutti i miei compagni di corso che mi hanno supportato in questi tre anni di studio.

Bibliografia

[1] Karen Wickre. Celebrating twitter7.
https://blog.twitter.com/2013/celebrating-twitter7.

[2] Alexa. twitter.com site overview.
http://www.alexa.com/siteinfo/twitter.com.

[3] Twitter. Documentation | twitter developers.
https://dev.twitter.com/docs.

[4] Twitter. Authentication & authorization | twitter developers.
https://dev.twitter.com/docs/auth.

[5] Twitter. Oauth faq | twitter developers.
https://dev.twitter.com/docs/auth/oauth/faq.

[6] Ugo Scaiella, Paolo Ferragina, Andrea Marino, and Massimiliano Ciaramita. Topical clustering of search results. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 223–232, New York, NY, USA, 2012.

[7] Wikipedia. Named-entity recognition.
http://en.wikipedia.org/wiki/Named-entity_recognition.

[8] Django. Django project.
https://www.djangoproject.com/.

[9] scikit learn. K means - 2.3. clustering - scikit-learn 0.14 documentation.
http://scikit-learn.org/stable/modules/clustering.html#k-means.

[10] Wikipedia. K-means algorithm.
http://en.wikipedia.org/wiki/K-means_clustering#Algorithms.

[11] Delbert Dueck Brendan J. Frey. Clustering by passing messages between data points.
http://www.psi.toronto.edu/affinitypropagation/FreyDueckScience07.pdf.

[12] scikit learn. Affinity propagation - 2.3. clustering - scikit-learn 0.14 documentation.
http://scikit-learn.org/stable/modules/clustering.html#affinity-propagation.

[13] Wikipedia. Affinity propagation algorithm.
http://en.wikipedia.org/wiki/Affinity_propagation#Algorithm.

[14] ryanmcgrath. Twython.
https://github.com/ryanmcgrath/twython.