Report

Rocco Caruso

March 15, 2016

Chapter 1

Stato dell'Arte

1.1 Event detection nei media tradizionali

L'attività di event-detection è stata a lungo utilizzata per individuare eventi da stream testuali derivanti dai media più tradizionali come giornali o radio, infatti l'event-detection è stata per molto tempo oggetto di ricerca del programma di *Topic Detection and Tracking TDT* [1], un' iniziativa promossa dalla DARPA [1], con lo scopo di organizzare stream di notizie testuali sulla base degli eventi di cui discutono. Secondo il TDT, l'obiettivo dell'attività di *event detection*, è scoprire nuovi eventi o eventi precedentemente non noti, a partire da stream di notizie testuali derivanti dai media tradizionali come notiziari o newswire, dove ciascun evento è definito come segue:

definzione 1 (Evento)**.** qualcosa, non banale, che accade in un luogo e tempo specifico

Le tecniche di event-detection possono essere classificate in due macro categorie: *document-piovot* e *feature piovot* a seconda che utilizzino feature dei documenti o feature temporali delle singole keywords presenti nei documenti. La prima scopre eventi effettuando un clustering dei documenti sulla base di una qualche funzione di distanza fra i documenti stessi [14], mentre nella seconda si studia la distribuzione delle singole parole e scoprono nuovi eventi raggruppando le parole [6] Come evidenziato da [14] infatti, l'event detection può essere ricondotto al problema della scoperta di pattern in uno stream testuale, quindi il modo più naturale per scoprire nuovi eventi, è quello di usare un algoritmo di clustering. Il task di event-detection si può suddividere in tre fasi principali: data preprocessing, data rappresentation, data organization o clustering. Nella fase di preprocessing vengono applicate

[1] Agenzia di ricerca agenzia per i progetti di ricerca avanzata per la difesa

al testo delle classiche tecniche di NLP come la rimozione di stopwords, tokenizzazione e stemming. I modelli di rappresentazione di dati più utilizzati per l'event detection sono *il modello vettoriale* e *il modello bag of words*, i cui elementi saranno diversi da zero, se il termine corrispondente è presente nel documento. A ciascun termine nel vettore, è assegnato un peso secondo lo schema *tf-idf* [12] che valuta quanto è importante una parola per un documento all'interno di un corpus. Questo modello di rappresentazione non prende in considerazione l'ordine temporale delle parole ne le caratteristiche sintattiche o semantiche del testo come il part of speech tag o named entities. Per questa ragione utilizzando questo modello, ad esempio, sarebbe difficile distinguere due eventi simili ma accaduti ad un mese di distanza fra loro. Nel lavoro di [14] il task di scoperta di nuovi eventi da uno stream testuale di news è suddiviso in due fasi principali: Retrospective Event Detection (RED), New Event Detection (NED). La prima fase (RED) comporta la scoperta di eventi da una collezione già nota di documenti, mentre nella seconda si cerca di identificare gli eventi dallo stream di notizie in tempo reale. Per il RED è stato utilizzato un algoritmo di clustering gerarchico: *Group Average Clustering GAC*, che consente anche di descrivere gli eventi identificati con diversi livelli di granularità. Per la fase di New Event Detection, invece, solitamente viene adottato un agoritmo di clustering incrementale single-pass [1, 14] che consente di suddividere i documenti nei vari cluster non appena arrivano dallo stream. In particolare ciascun documento viene elaborato sequenzialmente e viene assorbito dal cluster più simile, o verrà creato un nuovo cluster se la similarità è al di sotto di una soglia prestabilita. In un ambiente di detection on-line (NED) un forte vincolo è costituito dal fatto che non si può hanno informazioni di eventi futuri,ovvero non è possibile utilizzare dati provenienti da documenti successivi, cronologicamente, a quello corrente. Utilizzando un modello di rappresentazione vettoriale, questo vincolo pone delle problematiche su come gestire la crescita del vocabolario dei termini quando vengono aggiunti nuovi documenti al corpus e come modificare delle statistiche inerenti l'intero corpus come l'IDF. La soluzione suggerita da [14] è quella di modificare il vocabolario dei termini in maniera incrementale e modificare l'IDF ogni qual volta viene aggiunto un nuovo documento.

$$idf_t(w) = log_2 \left(\frac{N_t}{df_t(w)} \right) \tag{1.1}$$

dove N_t è il numero di documenti fino al tempo t e $df_t(w)$ è la document frequency della keyword w fino al tempo t. In pratica questi approcci NED, tendono a divenire molto costosi sia in termini di risorse computazionali che di tempo richiesto, e in taluni casi addirittura irrealizzabili se non utilizzando delle tecniche che ne migliorino l'efficienza. Una possibile tecnica per

ridurre i costi è quella di utilizzare una *time window* [8, 9] per limitare il numero vecchi documenti da analizzare quando si prende in considerazione un nuovo documento. Utilizzare una finestra temporale non solo riduce i costi, ma permette anche di limitare lo scope degli eventi scoperti, consentendo di identificare eventi simili ma che accadono in uno slot temporale diverso [14]. Tutte queste tecniche per il TDT si basano sull'assunzione che tutti i documenti siano rilevanti e contengono informazioni di eventi, poichè lavorano su stream di informazioni affidabili, assunzione che è chiaramente violata per quanto riguarda lo stream di Twitter. Nelle tecniche feature-pivot un evento viene invece modellato come una attività che presenta picchi di frequenza (burst), ovvero un evento è rappresentato dall'insieme di keywords che presentano un burst [1]. L'assunzione fatta da queste tecniche è che alcune parole avranno un incremento di utilizzo repentino quando accade un evento. Nel lavoro di [6] viene utilizzato un'automa a stati infiniti per poter identificare i burst delle keyword all'interno dello stream testuale. Gli stati dell'automa corrispondono alla frequenze delle singole parole, mentre le transizioni fra a gli stati identificano i burst che corrisopondo a un cambiamento significativo nella frequenza. A differenza delle tecniche document-pivot, in questo caso si cercano di identificare eventi raggruppando (ovvero effettuando il clustering) quelle keyword che presentano un burst, piuttosto che i documenti. Nel lavoro di [1] la frequenza delle parole viene modellata tramite una distribuzione bionomiale, dpoi vengono idnvidiuate le bursty-keywords sulla base di una soglia euristica, per poi raggrupparle al fine di identificare gli eventi.

1.2 Twitter Event Detection

L'attività di Event Detection nei microblogs come Twitter, è concettualmente molto simile all' Event Detection nei media tradizionali. In entrambi i casi, viene dato in input al sistema uno stream di documenti testuali e l'obiettivo è quello di scoprire degli eventi raggruppando i documenti o le singole parole contenute nei documenti stessi. L'unica differenza che hanno è il tipo e il volume di documenti dello stream che devono analizzare, in pratica tuttavia, questa unica differenza si riflette in una serie di nuove sfide per il task dell'event detection. Innanzitutto il volume di documenti nel caso dei microblogs come twitter è di diversi ordini di grandezza più grande rispetto ai media tradizionali, ma soprattutto nel caso di stream derivanti dai media tradizionali, tutti i documenti hanno una qualche rilevanza rispetto ad un avvenimento, una notizia. Nel caso dei tweet, invece, vi possono essere grandi quantità di messaggi privi di significato (pointless babbles) [4] e rumors [3]. Inoltre le caratteristiche di Twitter e la sua poplarità sono molto allettanti per

spammers e altri e altri content polluters [7] per disseminare pubblicità, virus, pornografia pishing o anche per compromettere la reputazione del sistema. La sfida più grande che bisogna affrontare nell'attività di event detection per i tweet, è quindi quella di poter separare informazioni mondane e inquinate da informazioni su eventi reali. Altre difficoltà sono causate principalmente dalla brevità dei messaggi (max 140 caratteri), dall'uso di abbreviazioni, errori di spelling e grammaticali, e l'uso improprio della struttura delle frasi e l'utlizzo di più lingue nel medesimo tweet. Per queste ragioni anche le tecniche tradizionali di natural language processing meno appropriate per i tweet. Un altro lavoro interessante che si colloca in quest'area è TwitterStand [13]: un sistema per scoprire le ultime notizie da twitter. Per poter distinguere il rumore dalle news, hanno selezionato manualmente 2000 utenti come "Seeders" ovvero utenti che pubblicano su Twitter news come stazioni televisive, giornali, bloggers etc. I tweets non appartenenti a questi seeders, invece, vengono filtrati per mezzo di un classificatore Naive Bayes. Dopo aver filtrato i tweet, viene applicato un algoritmo di clustering incrementale al fine di creare cluster tali che ognuno corrisponda ad una "news". Il modello di rappresentazione utilizzato è quello vettoriale con pesatura tf-idf e la funzione di similarità adottata è quella del coseno. Inoltre viene mantenuta una lista di cluster "attivi" per ridurre il numero di confronti da effettuare. In particolare un cluster viene definito inattivo se la media delle date di pubblicazione dei tweet, non supera i tre giorni. Una volta identificati i topic (news), il sistema cerca di localizzare ciascun cluster, ovvero cerca di assegnare una posizione geografica sia sulla base del contenuto testuale che sui geotag presenti nei tweet. Un altro sistema per scoprire news da twitter è stato proposto Phuvipadawat e Murata [11]. In questo lavoro per ridurre il rumore, i tweet vengono innanzitutto campionati utilizzando attraverso le streaming API di twitter e fornendo delle specifiche keyword da monitorare (#breakingNews, #breaking #news). I tweet raccolti vengono sucessivamente indicizzati tramite Apache Lucene[2]. I tweet simili fra loro vengono raggruppati per poter identificare news. Anche in questo caso la similarità adottata si basa sulla similarità del coseno fra le rappresentazioni tf-idf dei tweet, ma viene assegnato un *boost* per quei termini che corrispondono a nomi propri e per hashtag e username. I nomi propri sono identificati utilizzando lo Stanford Name Entity Recognizer [3] addestrato su un corpora di news tradizionali. I nuovi messaggi saranno inclusi in un cluster se sono simili al primo tweet e ai top-k termini presenti nel cluster. I cluster prodotti vengono poi ordinati sulla base dell'affidabilità (numero di followers) e popolarità

[2]https://lucene.apache.org/core/

[3]http://nlp.stanford.edu/software/CRF-NER.shtml

(numero di retweet). Gli autori hanno fortemente sottolienato l'importanza dell'identificazione dei nomi propri al fine di migliorare il calcolo della similarità fra i tweet, e di conseguenza migliorare l'accuratezza generale del sistema. Nel lavoro di [2] viene posta maggiore attenzione sull'identificazione di eventi reali da Twitter. Il metodo da loro proposto utilizza un algoritmo di clustering incrementale, che raggruppa i tweet simili fra loro e poi classifica i risultanti cluster in eventi real-world o non events. L'algoritmo di clustering utilizzato è il classico algoritmo di incrementale basato su soglia, ogni tweet è rappresentato mediante un boosted tf-idf vector, e viene utilizzata la similarità del coseno per valutare la distanza fra un tweet e il centroide di ogni cluster. Oltre ai classici step di pre-processing come tokenization, stopword removal e stemming, viene raddoppiato il peso per gli hashtag, poiché sono considerati fortemente indicativi del contenuto del messaggio. Gli autori hanno definito quattro tipologie di feature per i cluster individuati, per poter distinguere fra eventi reali ed da non-event o "twitter center topic" :

1. temporal-features: sono state definite un insieme di caratteristiche temporali per poter caratterizzare il volume dei termini più frequenti all'interno di un cluster.

2. social-features: insieme di feature che caratterizzano il grado di interazione degli utenti nei tweet del cluster come la percentuale di retweet, di replies. L'assunzione fatta è che i cluster contenenti un alta percentuale di retweet potrebbero non contenere informazioni di eventi reali.

3. topical-features: descrivono la coerenza del cluster rispetto ad un topic. L'idea sottostante è che i cluster relativi ad eventi tendono a svilupparsi attorno ad un tema comune, al contrario di non-event cluster che si sviluppano attorno a diversi termini comuni e.g: ("sleep" or "work"). Per stimare questa coerenza, gli autori calcolano la media della similarità dei tweet rispetto al centroide.

4. twitter centric-features: queste caratteristiche hanno lo scopo di identificare le attività twitter-centric. Esempi di queste feature sono la percentuale di tweet contenenti hashtag e la percentuale di tweet contenenti l'hashtag più utilizzato. Gli autori presumono che un'alta percentuale della prima stia ad indicare un topic conversazionale.

Poichè i cluster evolvono nel tempo, queste feature sono periodicamente aggiornate. Sulla base di queste feature, è stato addestrato una support vector machine (SVM) a partire da un insieme di cluster etichettati, che verrà usata

per decidere se un nuovo cluster contiene o meno informazioni relative a eventi reali. AL fine di permettere di eseguire il task di scoperta di eventi su larga scala invece, in [10] è stato ideato un algoritmo che fa ricorso al *Local Sensitive Hashing*[5] grazie a cui è possibile superare alcuni limiti degli approcci classici. L'utilizzo dell'LSH, permette di ridurre drasticamente il tempo richiesto per la scoperta del Nearest Neighbor di un punto in uno spazio vettoriale. In particolare per ogni documento proveniente dallo stream, viene calcolata la distanza rispetto al suo nearest neighbor, che verrà adoperata sia per determinare il grado di novità del documento rispetto ai precedenti (Novelty Score), sia per dall'altro è utilizzato per creare dei "thread di tweets" o in altri termini cluster, che come negli altri metodi document-pivot precedentemente descritti, si assume che corrisponderanno ad eventi. Viene definita una relazione *links* come segue: si dice che un tweet a ha un *link* verso il tweet b se la loro distanza è al di sotto di uuna soglia.

$$a \; links \; b \iff 1 - cos(a, b) < t \tag{1.2}$$

Dopodiché ciascun tweet a verrà assegnato ad un thread esistente se la sua distanza dal suo nearest neighbor è al di sotto la soglia, o sarà considerato come il primo tweet di un nuovo thread. Nel primo caso il tweet a sarà assegnato al medesimo thread cui appartiene il suo nearest neighbor. Cambiando la soglia t è possibile controllare la granularità dei thread prodotti. Se t è un valore molto alto, si avranno pochi thread ma molto grandi, mentre un valore di t molto basso genererà molti thread di piccole dimensioni. Una volta individuati i thread, vengono considerati solo quelli che crescono più velocemente. Un alto fattore di crescita da infatti indicazione del fatto che la notizia del nuovo evento si stia diffondendo, per tale ragione vengono restituiti solo i threads con il grow-rate più alto. L'aspetto più interessante del lavoro di Petrović et al. [10] è rappresentato dal fatto che il loro sistema riesce a processare un nuovo documento proveniente dallo stream, in tempo e spazio costante. Per ottenere tale risultato non solo impiegano l'lsh per ridurre il numero di confronti da effettuare, ma impongono ulteriori limiti per applicare tali tecniche allo streaming dei tweet. Infatti poiché il numero di bucket dell lsh sebbene possa essere alto è un numero finito, e in uno stream di dati, il numero di documenti che possono ricadere in un bucket potrebbe crescere a dismisura. Per tale ragione pongono un limite sul numero di documenti che può contenere un singolo bucket. Quando un bucket raggiungerà la sua massima capienza, verrà eliminato il documento più vecchio. Questa restrizione sebbene renda il numero di confronti costante, tale costante può divenire piuttosto alta e quindi viene posto un ulteriore vincolo per limitare superiormente il numero di confronti. Bisogna sottolineare però, che se da un lato questi vincoli permettono al sistema di lavorare in setting incrementale,

dall'altro andranno ad esacerbare il problema della *fragmentation*, poiché il sistema effettuando per ogni nuovo tweet un numero costante di confronti tenderà a generare un numero maggiore di sotto-eventi. La frammentazione si presenta quando tweets che discutono dello stesso evento sono assegnati a cluster diversi. Questo problema affligge la maggior parte tutti i sistemi che tentano di estrarre documenti dallo streaming fi tweet, tramite il clustering. In particolare impostando una soglia di similarità troppo bassa, si avrà una maggiore probabilità di frammentazione, impostando un valore molto basso invece, può dare origine al problema opposto il *merging* ovvero il problema duale della fragmentation, che si presenta quando tweet di che discutono di eventi diversi sono raggruppati in un unico cluster. Per alleviare il problema della frammentazione, Sankaranarayanan et al [13] suggeriscono, una volta identificati i cluster, di tentare di fondere periodicamente i cluster simili o duplicati. É doveroso sottolineare che tutti i metodi descritti precedentemente, si basano in linea di massima sulla mera similarità testuale fra i tweet, di conseguenza sono esposti non riusciranno quindi ad identificare pattern di tweet composti da parole sintatticamente diverse ma semanticamente correlate (sinonimia), o al contrario produrranno cluster di tweet composti da termini identici ma semanticamente non correlati (polisemia).

Chapter 2

Local Sensitive Hashing

Un problema fondamentale che molti task di Data Mining devono affrontare è quello di esaminare i dati per trovare item "simili".

Bibliography

[1] James Allan, editor. *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[2] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.

[3] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 675–684, New York, NY, USA, 2011. ACM.

[4] Jonathan Hurlock and Max L. Wilson. Searching twitter: Separating the tweet from the chaff. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.

[5] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[6] Jon Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 91–101, New York, NY, USA, 2002. ACM.

[7] Kyumin Lee, Brian David Eoff, and James Caverlee. Seven months with the devils: A long-term study of content polluters on twitter. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.

[8] Gang Luo, Chunqiang Tang, and Philip S. Yu. Resource-adaptive real-time new event detection. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 497–508, New York, NY, USA, 2007. ACM.

[9] R. Papka. On-line new event detection, clustering, and tracking title2:. Technical report, Amherst, MA, USA, 1999.

[10] Saša Petrović, Miles Osborne, and Victor Lavrenko. Streaming first story detection with application to twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 181–189, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[11] Swit Phuvipadawat and Tsuyoshi Murata. Breaking news detection and tracking in twitter. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '10, pages 120–123, Washington, DC, USA, 2010. IEEE Computer Society.

[12] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[13] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: News in tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 42–51, New York, NY, USA, 2009. ACM.

[14] Yiming Yang, Tom Pierce, and Jaime Carbonell. A study of retrospective and on-line event detection. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 28–36, New York, NY, USA, 1998. ACM.