Progetto di Ingegneria del software 2021

Paolo Ciancarini

11 ottobre 2021

1 Introduzione

Vogliamo costruire uno strumento di raccolta e analisi dei tweet di Twitter, specie quelli con foto e geolocalizzabili. I tweet possono riferirsi ad una persona, ad un luogo, ad un evento. In alcuni casi i tweet riguardano situazioni di emergenza. In altri possono essere usati per diffondere informazioni o per farsi pubblicità.

Le persone che vivono un'emergenza e la comunicano attraverso un social media quale Twitter possono essere considerate dei sensori umani sul territorio e i messaggi che si scambiano generano, se aggregati in tempo reale, una serie di informazioni utili proprio per gestire l'emergenza stessa.

In una situazione di protezione civile, come subito dopo un terremoto distruttivo, i tweet costituiscono una delle poche fonti immediate ed economiche di informazione dal campo, utilizzabile per ricostruire una situazione operativa[1].

Ci sono tante altre situazioni interessanti e non emergenziali, ad esempio: la condivisione di un evento come un concerto o una partita di calcio, con le foto. Oppure i visitatori di un museo che in uno stesso periodo di tempo pubblicano tweet sul museo stesso. In questo caso i tweet possono essere "ricomposti" per offrire un punto di vista speciale sull'evento o sul luogo visitato.

Un'altra situazione tipo è quella di un gruppo di turisti che si sposta in una città, visitandola e condividendo commenti e foto via tweet: si può cercare di ricostruire i movimenti delle persone del gruppo a partire dai tweet che producono, piazzando inoltre le foto su una mappa.

2 Descrizione del problema

L'obiettivo del progetto è quello di creare un'applicazione in grado di raccogliere i tweet e organizzarli. Lo scopo è rilevare eventi locali o basati su particolari parole chiave esaminando i tweet stessi. La raccolta può essere storica (es. ultima settimana) o in stream in tempo reale.

L'applicazione dovrà permettere di visualizzare, consultare i tweet con certi hashtag e - sotto certe condizioni- attivare una procedura specifiche. Ad esempio, il sistema dovrà essere in grado di:

- raccogliere i tweet (scoprirete che certe raccolte di tweet sono a pagamento: ovviamente vogliamo usare solo tweet gratis, quindi imparate a raccoglierli voi)
- classificare i tweet geolocalizzati
- classificare i tweet pubblicati in una data area geografica
- classificare i tweet contenenti un certo punto di interesse, ad esempio #SanSiro
- classificare i tweet con una certa parola chiave, ad esempio #viaggiodiclasse5B
- classificare i tweet geolocalizzati di una persona specifica, per seguirne gli spostamenti
- analizzare il sentimento (*sentiment analysis*) di una serie di tweet, ad esempio un gruppo di persone che visita un museo gradisce la visita?

[1] blog.twitter.com/en_sea/topics/insights/2018/5-Tips-for-using-Twitter-during-emergencies-and-natural-disaster.html

I tweet così raccolti e classificati potranno essere aggregati in vari modi, principalmente in modalità grafica.

In particolare si richiede di aggregare i tweet in una *dashboard* (visualizzatore) interattiva, possibilmente mostrando viste coordinate che presentino diversi dettagli dei dati (es. le posizioni su una mappa, una *term cloud* (nuvola di parole dei termini usati nei tweet), un diagramma a barre con il numero di tweet nell'unità di tempo, ecc.), un grafico a torta con sentiment positivi e negativi. È un requisito fondamentale mostrare la posizione dei dati raccolti su una mappa (es. su Google Maps).

Suggerimento: iscrivetevi a Twitter (se non lo siete già) e trasmettete vostri tweet personali usando l'hashtag `#IngSw2021`. Pubblicate anche tweet su luoghi o eventi o persone a vostra scelta. Attivate sulla app Twitter del vostro telefono la geolocalizzazione dei tweet (questa è una funzione volontaria di Twitter[2]).

IMPORTANTE: ogni team può aggiungere specifiche funzioni di propria scelta al prodotto, previa approvazione del PO. La proposta di funzionalità è a cura del PO Operativo, che contatta il docente. Siate creativi, la cosa verrà apprezata adeguatamente in sede di esame.

Esempio: una proposta riguarda la possibilità di fare una mappatura dei bagni pubblici di una città, per esempio Bologna. Quando usate un bagno pubblico - in un bar, un ristorante, all'università,ecc.-, dopo mandate un tweet usando l'hashtag #WC-OK e un commento geolocalizzato. Si propone di costruire una mappa aggiornata dei bagni pubblici "di qualità" di Bologna.

3 Un processo agile "estremo"

I gruppi di progetto dovrebbero avere cinque membri. Tollerati gruppi con un membro in più o in meno solo se residuali.

Uno dei membri deve assumere il ruolo di Product Owner Operativo (in raccordo coi PO che sono i docenti); un altro membro sarà Scrum Master. Gli altri membri sono developer (sviluppatori).

Il PO Operativo è il responsabile del prodotto entro il team, e partecipa allo sviluppo. Lo Scrum Master è il coordinatore di processo entro il team (e quindi dei documenti di processo, in particolare del rapporto finale, che comunque verrà firmato da tutti i membri del team). Anche lo Scrum Master può partecipare allo sviluppo.

Il processo si svolge su almeno tre sprint di tre settimane ciascuno, a partire dall'11 ottobre. Ci sono inoltre una fase preliminare (*sprint zero*) e una fase conclusiva (*release sprint*), che conclude lo sviluppo.

Fase preliminare a) Team forming tramite Trello b) Addestramento a Scrum del gruppo mediante uso del gioco Scrumble (vedere link alla fine di questo documento), con raccolta degli indicatori di valutazione del team da parte dello Scrum Master. Una partita a Scrumble si può giocare sia in presenza sia con Teams e richiede circa un paio di ore.

Fase di esecuzione Il team esegue il progetto con almeno tre sprint della durata di tre settimane. Al termine di ciascuno sprint il team dovrà preparare una demo dello stato di avanzamento del prodotto (*Sprint review*), una riflessione sul processo (*retrospettiva*) usando le carte Essence, e una sessione di *backlog grooming* cioè una revisione del backlog dopo una riunione coi PO docenti.

IMPORTANTE: all'inizio di sprint 2 e di sprint 3 il PO (Docente) fornirà nuove user story, che dovranno essere aggiunte alle altre, eventualmente modificandole.

Fase conclusiva Preparazione del rapporto finale a cura del team, inclusa autovalutazione (verranno date indicazioni sul rapporto finale e sull'autovalutazione entro la fine del corso). Consegna del software su gitlab@CAS e del rapporto finale entro 25 gennaio. Si richiede la preparazione di un breve video con audio (max 3 min.) di presentazione del prodotto sviluppato, che mostri anche un'analisi "interessante" eseguita con il prodotto stesso.

Voti Voto unico al team basato su demo e qualità del rapporto finale.

Allo scopo di mostrare la flessibilità dell'approccio agile, ciascun team decide e inizia lo sviluppo con un insieme di user story che potrà essere integrato in corso d'opera con altre, richieste dal docente.

[2] https://help.twitter.com/en/safety-and-security/tweet-location-settings

Esempi (lista non esaustiva): aggiungere un elemento di filtro/selezione dei tweet, come la possibilità di selezionare una finestra temporale, in modo che l'utente del sistema possa selezionare i tweet solo di un determinato periodo; aggiungere la possibilità di cliccare su una delle parole della tag cloud in modo da filtrare i dati sulla base di tale parola chiave e di vedere quindi la mappa aggiornata di conseguenza; aggiungere altre sintesi grafiche di data analytics, quale un istogramma che mostri la frequenza con cui è comparsa una parola chiave nella finestra temporale prescelta.

Tecnologie: per lo sviluppo potete usare Java, Javascript, Python o altre tecnologie, a vostra scelta. Si richiede che il deployment sia comunque basato su docker. Per quanto possibile TUTTE le tecnologie utilizzate dovrebbero essere open source.

3.1 I servizi di CAS

L'uso dell'ambiente di sviluppo CAS [1] è obbligatorio; il nostro scopo è di fare una esperienza di sviluppo "estremo", cioè totalmente basato su tecnologie open source; inoltre vogliamo poter raccogliere dati di processo grazie alle funzionalità presenti nell'ambiente CAS [2]. L'ambiente CAS contiene i seguenti elementi, tutti open source:

1. Taiga (simile a Trello) per il project management e per raccogliere alcuni documenti nel suo wiki
2. Mattermost (simile a Slack) per le comunicazioni tra gli sviluppatori
3. SonarQube per l'analisi statica del codice
4. gitlab per il controllo delle versioni e della configurazione
5. jenkins per il testing e la *Continuous Integration*
6. logger server, per raccogliere dati di sviluppo direttamente dai client IDE arricchiti con plugin (disponibili plugin per Eclipse, Atom, IntelliJ).

L'ambiente CAS è disponibile in più versioni:

- server AMINSEP: all'indirizzo `https://aminsep.disi.unibo.it` si accede Taiga. Gli altri servizi si accedono aggiungendo il nome del servizio: ad esempio gitlab si accede da `https://aminsep.disi.unibo.it/gitlab`
- CAS si può istallare su un server privato del team, ma alla fine del progetto occorre duplicare il contenuto del gitlab privato su gitlab@CAS
- è accettabile deployare CAS su cloud, ma alla fine del progetto occorre duplicare il contenuto del gitlab privato su gitlab@CAS

Le carte Essence [3] per le retrospettive si trovano al link di cui sotto.
Altre informazioni verrano date a lezione.

4 Link

Twitter https://developer.twitter.com/en/docs

Tweet-Tracker https://github.com/Zacomo/Tweet-Tracker

Scrumble http://scrumble.pyxis-tech.com

Essence Introduzione http://semat.org/essence-user-guide

Carte Essence https://practicelibrary.ivarjacobson.com/start

Riferimenti bibliografici

[1] P. Ciancarini, M. Missiroli, F. Poggi, and D. Russo. An open source environment for an agile development model. In *IFIP International Conference on Open Source Systems*, volume 582 of *Advances in Information and Communication Technology*, pages 148–162. Springer, 2020.

[2] P. Ciancarini, M. Missiroli, and S. Zani. Empirical evaluation of agile teamwork. In *International Conference on the Quality of Information and Communications Technology*, pages 141–155. Springer, 2021.

[3] I. Jacobson, H. Lawson, P. Ng, P. McMahon, and M. Goedicke. *The Essentials of Modern Software Engineering*. ACM Books. Morgan & Claypool Publishers, 2019.