Sistema di supporto alle decisioni economiche per la corsa alle elezioni comunali

PROPOSTA DI PROGETTO

LM Informatica
CV Management
Università di Bologna

Dario Floris 953729

Stefano Balla 953765

Proposta

Nella primavera del 2020 in Italia erano state programmate le elezioni amministrative, ma dato l'avvento della pandemia causata dal COVID-19 sono state rimandate all'autunno seguente.

Oltre al rinvio delle date, il virus ha stravolto il tessuto socioeconomico di molte realtà tra le quali certamente anche le amministrazioni comunali, le quali si sono dovute trovare ad affrontare un'emergenza sanitaria in termini organizzativi, medici ed economici allo scadere del bilancio di fine mandato. Questo ha comportato un depauperamento delle casse comunali e necessariamente una rivisitazione dei programmi elettorali dei candidati per le prossime elezioni, i quali dovranno confrontarsi ancora più di prima con la gestione economica; sia per i fondi rimasti nelle casse comunali che per quelli che potrebbero entrare come finanziamenti europei, statali o simili.

Al di fuori dello scenario introdotto dal virus, si aggiunge il fatto che negli ultimi anni i metodi da sempre utilizzati dai partiti e dalle istituzioni per confrontarsi con i cittadini e coglierne le esigenze sono diventati sempre di più obsoleti, ne è prova il basso livello di fiducia dei cittadini nei partiti [1]. Questo non significa che i cittadini siano meno interessati alle vicende politiche, amministrative in questo contesto, anzi è sufficiente ricordarsi della grande partecipazione e al dialogo che i vari sindaci hanno costruito sui loro social network durante la recente emergenza e ancora i dati istat ci dicono che una gran parte della popolazione parla e si informa di politica quotidianamente, per questo usa internet fra cui sempre di più i social network [2]. Quindi potrebbe davvero essere necessario uno strumento che aiuti chi concorre per cariche amministrative, in maniera più efficace rispetto ai metodi classici, a comprendere le esigenze dei cittadini di un certo comune.

In risposta a queste riflessioni, la seguente proposta di progetto vuol essere un sistema di supporto alle decisioni sul tema della destinazione dei fondi a varie classi di problemi sottoposti dai cittadini in maniera indiretta.

Il sistema deve raccogliere dei giudizi o delle lamentele sottoposte dai residenti di un comune direttamente dai commenti o dai post che questi lasciano sui social Network, poi deve classificare questi feedback in classi di problemi, usando un sistema di priorità all'interno di ogni classe che favorisca la precedenza ai problemi più frequenti. Infine, utilizzando i dati forniti dal comune nei quali vengono indicate le voci dei fondi e dei preventivi nei quali sono indicati i costi per diverse opere, il sistema deve fornire una configurazione dello scheduling di risorse economiche ai vari temi massimizzando la risoluzione di problemi con priorità alta, così da soddisfare per più classi di problemi le lamentele/proposte più sentite dai cittadini.

La proposta trova quindi un target d'utenza tra liste civiche, partiti o simili che: in fase di costruzione del programma elettorale, possono usufruire di questo supporto per assecondare le richieste dei cittadini e guadagnare consenso, sfruttando le risorse economiche rimanenti dichiarate dall'amministrazione uscente.

Design

Il sistema proposto dovrebbe essere costituito da 4 componenti principali, delle quali si fornisce la spiegazione in seguito e di altre necessarie per il corretto funzionamento le quali però non verranno elencate, in quanto il sistema non è stato completamente implementato per mancanza di risorse.

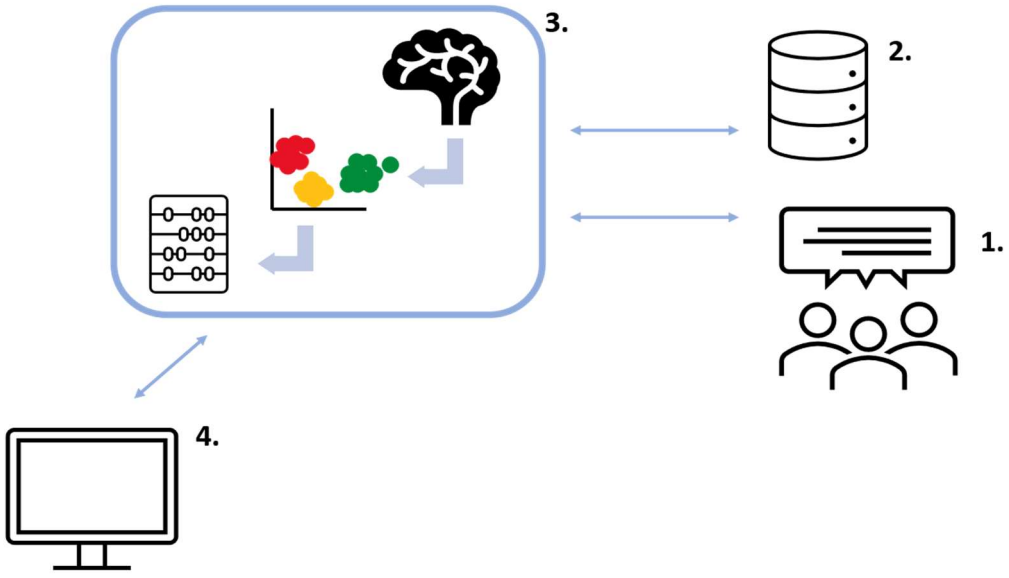
Una fase necessaria per il problema è la raccolta dei dati intesi come commenti e post da Facebook, il social network per il quale viene proposto questo progetto.

I dati che si stanno cercando in questa fase sono composti da proposte o lamentele che i residenti di un particolare Comune esprimono su Facebook sotto forma di testo.

Queste informazioni devono essere recuperate sulla base dell'appartenenza alla città, a partire da profili e pagine tipicamente legate al comune, e.g la pagina del sindaco, servizi di cronaca locale etc.

Per la raccolta di dati abbiamo pensato all'utilizzo di un *webcrawler* integrato alle *graphAPI* di Facebook di cui si discuterà ampiamente nel seguito.

Questa raccolta di informazioni sarebbe poi la base del database del sistema alla quale andrebbero aggiunte le informazioni di bilancio del comune e preventivi necessari per il progetto. (vedi 1. e 2. in Figura)



La seconda fase è quella di riconoscimento e divisione del tipo di lamentela o proposta; i dati precedentemente raccolti devono ora essere compresi come problemi per poi essere clusterizzati in classi di problemi simili. Queste classi di problemi dovrebbero contenere quindi delle liste di problemi o proposte, simili per tema.

A questo punto è necessaria un'integrazione dei preventivi con tutti i problemi trovati; la descrizione di questo passo viene escluso perché fortemente legato all'implementazione. Si noti però che l'integrazione o meglio, l'associazione, dei problemi a dei preventivi è un passo importante per l'operazione successiva.

Infatti, dopo la divisione in cluster di problemi simili e l'integrazione di un modello di costi per i relativi problemi si può immaginare un cluster A come una lista di elementi chiave-valore con problema-costo e.g A= {rifacimento strade - 10, buca marciapiede via Rossi- 5, ..}.

Con questo tipo di rappresentazione e con un'informazione di quante risorse sono disponibili, in termini economici, viene poi proposto un problema di soddisfacimento vincoli con ottimizzazione che assegni ad ogni classe di problema le risorse necessarie per ridurre il numero totale di problemi preferendo l'assegnamento di risorse a problemi con priorità elevata.

Infine, la quarta componente dovrebbe consistere in un'interfaccia dalla quale l'utente possa recuperare la soluzione del problema di vincoli.

Raccolta dati

Per la realizzazione di un sistema che rappresenti un elemento di supporto, ai processi decisionali di carattere socioeconomico dei Comuni, è necessario identificare le fonti dalle quali attingere i dati e successivamente, definire dei meccanismi che permettano la loro raccolta.

Attualmente i SNS (Social Network Service) come Facebook, Twitter, Instagram etc. si sono dimostrati essere degli elementi centrali e fondamentali nello studio di fenomeni di carattere sociale.

Abbiamo dunque riconosciuto i SNS come fonte principale da cui attingere i dati.

Una volta identificate le fonti, è necessario concentrarsi sulla definizione delle meccaniche da utilizzare durante la raccolta di informazioni.

La ricerca bibliografica effettuata ha portato ad identificare i Web Crawler e le Graph API di Facebook come i mezzi più adatti a tale scopo.

Qui esponiamo una possibile architettura per un Web Crawler di Facebook e un software che utilizzi le API messe a disposizione dallo stesso.

Considerazioni preliminari

Realizzare un Crawler che esplori il grafo di Facebook può risultare un compito arduo e difficile, soprattutto se non vengono fatte delle importanti osservazioni in merito al tipo di operazioni che si vogliono effettuare.

Secondo recenti studi [3], il grafo di Facebook sarebbe costituito da 44 Terabytes di dati.

Per lo scopo del nostro progetto la quantità di dati da scaricare risulta infinitesimale rispetto ad essa, ma la dimensione del grafo con cui si lavora non è da trascurare perchè varia a seconda della località geografica d'interesse e per città molto grandi potrebbero risultati grafi di dimensioni notevoli.

È dunque importante essere in grado di fare una stima della quantità di dati, e in caso non risulti possibile effettuare una raccolta dati completa per una certa località, bisogna assicurarsi di reperire e generare dei "sample" di dati rappresentativi.

Metodologia della ricerca

Per rendere possibili le operazioni di crawling prevediamo l'utilizzo dell'algoritmo di ricerca BFS (*Breadth-First-Search*) in combinazione con i criteri specifici della ricerca. L'algoritmo BFS è facile da implementare e si è dimostrato ottimale per l'esplorazione dei nodi di un grafo di SNS. L'algoritmo è stato utilizzato in vari studi relativi ai SNSs quali ad esempio i seguenti [4], [5].

Come abbiamo esposto in precedenza, non siamo in grado di determinare con precisione la grandezza della porzione del grafo che andremo ad esaminare, anche inserendo precisi criteri di ricerca (località dell'utente, età ecc) che permettono di ridurne la grandezza.

Web Crawler

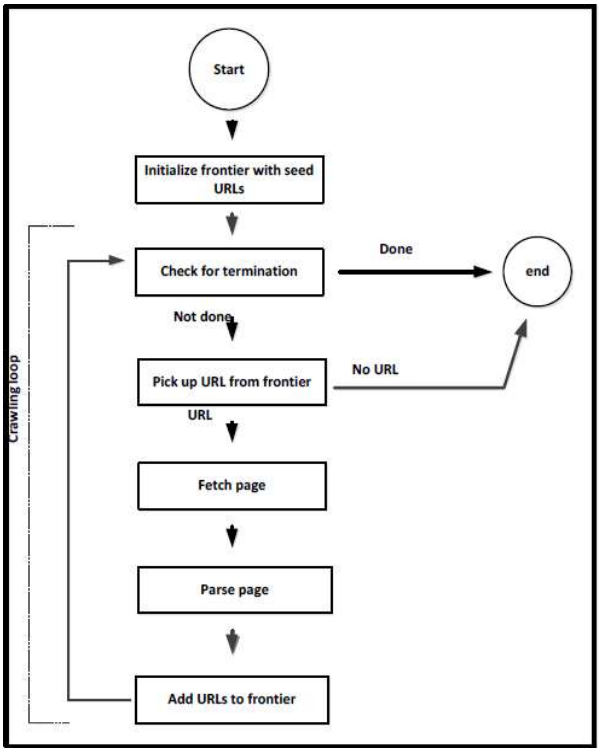
Un Web Crawler è un software che naviga il World Wide Web in maniera automatica e con una metodologia. Questi software sono utilizzati per indicizzare le pagine Web, permettere la manutenzione dei siti e consentire l'estrapolazione e il raccoglimento dei dati.

Il crawler necessita di uno o più URL da cui iniziare ad operare.

Nella forma generale, il crawler si compone di due elementi:

- **Frontiera:** lista di URL o ID utente. La lista viene aggiornata durante l'esecuzione del crawler da nodi del grafo che non sono stati ancora visitati.
- **Loop crawler:** qui è dove si svolge la maggior parte delle operazioni di crawling. Attinge dalla frontiera gli URL o gli ID utente rispettando le politiche FIFO che prevediamo di implementare. Per ogni nodo visitato vengono esportate tutte le informazioni ritenute utili. Se vengono identificati nuovi URL e ID utente, questi vengono aggiunti alla frontiera.

L'architettura prevista per il Crawler è esposta qui di seguito [6].



Abbiamo ipotizzato di implementare un Parser che permetta di individuare e raccogliere le informazioni utili al caso di studio, presenti nelle pagine Web che vengono visitate durante l'esecuzione del software.

Facebook Crawling

La realizzazione del Crawler di Facebook prevede l'utilizzo di librerie alternative che sfruttano le Facebook Graph API. Esistono diverse librerie open source messe a disposizione degli sviluppatori per realizzare applicazioni che interrogano i Server database di Facebook per svolgere diverse operazioni. Le librerie che in letteratura e in altri studi [6] sono state utilizzate nell'ambito della raccolta dei dati sono le RestFB, per comunicare con i server DataBase di Facebook, e l'interfaccia FacebookClient per comunicare e sfruttare le Facebook Graph API. Quest'ultima libreria permette l'estrapolazione e la raccolta dei dati.

Il Crawling di Facebook è svolto rispettando le TOS della piattaforma, dunque è necessario che il software sia in grado di autenticarsi per stabilire la connessione col server di Facebook. A seguito dell'autenticazione, l'agente potrà iniziare la ricerca e la raccolta dei dati: partendo da un seed visiterà i profili degli utenti identificati per ID, questi ultimi estratti dalla lista amici rispettando i criteri di ricerca e, per ogni utente raccoglierà i feed (post e commenti) che sono stati generati.

Prevediamo di iniziare il processo di crawling dagli URL che riportano alle pagine dei gruppi del comune, alle pagine dei profili utente delle persone che fanno parte dell'amministrazione comunale o profili utente che rappresentano il comune.

Da queste pagine vengono individuati e raccolti i post e i commenti e viene aggiornata la frontiera con gli URL e gli ID della lista dei membri del gruppo o dalla lista amici.

Data Cleaning e pre-processamento

Una volta ottenuti i dati è prevista l'implementazione di una serie di operazioni necessarie per trasformare i dati e renderli utili per l'analisi. Nello specifico ipotizziamo di effettuare la tokenizzazione del testo attraverso la trasformazione di tutte le parole in lower-case, lo stemming e la rimozione dei numeri.

I token vengono così inseriti in una matrice sparsa usando la frequenza inversa dei termini.

Questo metodo consente di catturare e considerare i vocaboli più importanti che, in altri termini, rappresentano le parole chiave e, di conseguenza, l'argomento principale esposto dagli utenti. [7], [8], [9].

Elaborazione dati

Per il sistema troviamo utile la realizzazione e l'impiego di due sottosistemi. Tali sottosistemi possono essere considerati complementari perché entrambi contribuiscono a identificare le problematiche espresse dagli utenti per cui si cercherà di suggerire una soluzione nella fase finale. Il primo sottosistema è rappresentato da un modello di *Topic Detection;* il secondo consiste nella realizzazione e utilizzo di un modello per la *Sentiment Analysis*. Le motivazioni che ci hanno spinto verso questa decisione sono date dalla necessità di rendere possibile l'identificazione dei problemi (realizzabile attraverso l'analisi dei post e dei commenti degli utenti), e la clusterizzazione di questi.

Dunque, l'idea di base per il nostro sistema è la seguente: attraverso la sentiment analysis si potranno classificare le opinioni degli utenti; mentre attraverso la topic detection sarà possibile identificare l'argomento a cui le opinioni fanno riferimento.
Infine sarà possibile eseguire la clusterizzazione dei topic, realizzando degli agglomerati sulla base della correlazione semantica dei topic stessi.

Ci occuperemo prima di trattare la realizzazione del modello di *Sentiment Analysis* e successivamente tratteremo la definizione di un possibile modello per la *Topic Detection.*

Sentimenti Analysis:

Classificazione basata sul lessico

Un requisito necessario per svolgere la classificazione basata sul lessico è l'utilizzo di un dataset con lessico polarizzato. Per migliorare i risultati della classificazione è buona pratica fondere i dataset di lessici differenti: il primo relativo al campo specifico dello studio e il secondo di carattere generale.

Un altro requisito per il classificatore è l'impiego di una *sentiment scoring function.* Esistono diverse funzioni e schemi per realizzare modelli per il punteggio delle opinioni espresse dagli utenti [10].

Classificazione basata sull'apprendimento automatico

Per rendere possibile la classificazione basata sull'apprendimento automatico risulta necessario utilizzare, oltre al *corpus* (insieme dei documenti da classificare), un dataset per la fase di training della macchina. Il classificatore sarà così modellato sulla base del dataset garantendo la possibilità di identificare le opinioni negative e positive riguardo uno specifico argomento.

Intendiamo implementare il classificatore Naive Bayes che, come riportato in [11],[12] risulta essere adeguato alla classificazione del testo. Prevediamo comunque che sarà necessario svolgere delle modifiche al classificatore di base per ottenere risultati soddisfacenti e adeguati al caso di studio.

Topic Detection e clusterizzazione

I risultati ottenuti dal modello di *Sentiment Analysis* verranno utilizzati per l'identificazione vera e propria delle problematiche esposte dagli utenti. Più precisamente, i risultati che prevediamo di ottenere dalla Sentiment Analysis permettono di identificare le opinioni negative, neutre e positive degli utenti.

Il modello di Topic Detection si baserà su tali risultati per identificare con precisione l'argomento per cui è stato scritto un insieme di opinioni negative, positive o neutre. Il sottosistema di Topic Detection previsto richiede l'impiego di tecniche per la clusterizzazione dei termini presenti nei documenti. La clusterizzazione di tali termini si baserà sulla similarità che esiste fra di essi.

Il primo elemento fondamentale previsto per identificare i topic è l'impiego del modello LDA (*Latent Dirichlet Allocation*) che permette la realizzazione delle distribuzioni dei termini.

La complessità delle distribuzioni è a noi attualmente sconosciuta, ma prevediamo di rielaborare il modello con l'implementazione di tecniche di sampling specifiche (Markov Chain Monte Carlo, Gibbs Sampling) poiché i risultati ottenuti da [13] sono molto incoraggianti per l'analisi Topic Detection applicata ai servizi di Microblog quali Facebook e Twitter.

Arrivati a questo punto è possibile svolgere la clusterizzazione dei termini, prevedendo di implementare l'algoritmo *Single-Pass Clustering,* i cui risultati della sua applicazione risultano essere soddisfacenti in diversi ambiti [14], [15].

L'algoritmo sfrutta una funzione di similarità per determinare la distanza fra i termini. Maggiore è la distanza calcolata fra due termini, minore sarà la loro similarità.

In base all'algoritmo siamo così in grado di realizzare il raggruppamento delle parole chiave simili tra loro all'interno di categorie, che rappresenteranno i topic principali esposti dagli utenti.

Risoluzione problema

I passi svolti finora hanno portato alla classificazione di problemi, pubblicati dai cittadini, in classi di problemi, per la quale la lista civica si vuole impegnare nel raccogliere consensi sulla base di ciò che i cittadini pensano vada fatto.

Formulazione

Il problema che emerge è un problema di soddisfacimento vincoli, infatti sappiamo che ci sono N problemi, divisi in M cluster. Ognuno degli N problemi ha un costo di risoluzione associato C, che potrebbe essere ricavato da dei dati quali ad esempio preventivi già presenti nei db comunali, o fatture di spese per opere simili negli anni passati etc. e quindi momentaneamente da intendere come dato di input.

Oltre ad un costo, ognuno degli N problemi ha una priorità P che nel nostro caso è intesa come la frequenza di volte che un problema viene proposto dai cittadini, è evidente che se per esempio molti cittadini diversi sottolineano il bisogno di rifare "via Rossi" e un solo cittadino esprima il desiderio di rifare "via Verdi", il rifacimento di "via Rossi" debba avere una priorità più alta. Quindi calcolata la frequenza di ogni problema è possibile normalizzarla in una scala predefinita, ad esempio una scala da 1 a 4 che indicherà la priorità.

L'obiettivo finale è quello di trovare il massimo numero di problemi risolvibili con il budget a disposizione massimizzando la risoluzione di problemi con priorità alta; infine si vogliono trovare le quantità di risorse che devono essere destinate ad ogni classe di problema e quali problemi è possibile risolvere con un budget B fissato a priori.

La soluzione di questo problema di vincoli consentirebbe di risolvere i problemi più sentiti dai cittadini con il budget a disposizione e quindi possibilmente di guadagnare consenso.

Implementazione

L'implementazione di questo problema è stata realizzata in Minizinc 2.4.3 e viene proposta nella figura seguente:

```
1  %cluster di problemi numerati consecutivamente
2 set of int : ITEMA;
3 set of int: ITEMB;
4 set of int: ITEMC;
5
6 % budget riorse a diposizione
7 int: budget;
8
9 % priorità di ogni problema
10 array[ITEMA union ITEMB union ITEMC] of 1..4: prior;
11 % costo di ogni problema
12 array[ITEMA union ITEMB union ITEMC] of int: costi;
13
14 %insieme dei problemi risolvibili con quel dato budget/priorità
15 var set of ITEMA union ITEMB union ITEMC: problrisolti;
16
17 %costo problemi minore del budget
18 constraint sum (i in problrisolti) (costi[i]) <= budget;
19
20 % massimizzare la risoluzione di problemi con priorità alta
21 solve maximize sum (i in problrisolti) (prior[i]) ;
22
23 % quantità di fondi da destinare ad ogni cluster
24 var int: destinazionefondiA= sum(i in (problrisolti intersect ITEMA))(costi[i]);
25 var int: destinazionefondiB= sum(i in (problrisolti intersect ITEMB))(costi[i]);
26 var int: destinazionefondiC= sum(i in (problrisolti intersect ITEMC))(costi[i]);
27
28 output ["problrisolti = \(problrisolti)\n"++
29 "destinazionefondiA=\(destinazionefondiA)\n"++
30 "destinazionefondiB=\(destinazionefondiB)\n"++
31 "destinazionefondiC=\(destinazionefondiC)"];
32
```

Fino a riga 4 vengono definiti 3 cluster di problemi, ovviamente si possono aumentare il numero di cluster in base alla natura del problema; in questo caso abbiamo quindi M=3.

Alle righe 10 e 11 vengono definiti due array che indicano la priorità P e il costo C di ogni problema. Questi array sono da riempire, nel nostro caso manualmente, ma sarebbe possibile aggiungere un'estensione che si occupi dell'inizializzazione di questi parametri.

Alla riga 15 invece viene definita la prima variabile decisionale, ovvero ciò che vogliamo trovare: ovvero un sottoinsieme dell'unione di tutti i problemi che soddisfi i nostri vincoli.

La somma dei costi di questo sottoinsieme deve essere minore del budget B, riga 18; inoltre vogliamo che vengano scelti i problemi con priorità P più alta, riga 21.

Nelle righe 24-26 vengono definite le variabili decisionali relative a come dividere le risorse, ovvero il budget, per ogni classe di problema.

Risultati

Il risultato di questo problema fornisce l'insieme di numeri interi, che devono rappresentare univocamente un problema, che massimizzano la priorità tenendo conto dei costi e del budget a disposizione. Inoltre, restituisce la suddivisione delle risorse per ogni classe di problema.

Si noti che è necessario una funzione che associ ad ogni problema un numero intero come identificatore univoco e che se il cluster A contiene elementi identificati con valori da 1 a n, il cluster B deve contenere elementi identificati con valori da n+1 a l e C con valori da l+1 a z, con n,l,z numeri interi.

Se si lancia il programma con i seguenti valori, ci aspettiamo che venga trovato un sottoinsieme che contenga sicuramente il problema 13 in quanto ha un costo basso e una priorità alta, probabilmente conterrà anche il problema 10 in quanto nonostante abbia un peso alto ha priorità massima.

```
ITEMA= {1,2,3};
ITEMB= {4,5,6,7};
ITEMC= {8,9,10,11,12,13};
budget=20;
costi= [1,2,11,2,4,3,5,7,1,8,4,2,1];
prior= [1,1,2,3,4,3,2,1,2,4,1,1,4];
```

Il risultato ottenuto è il seguente:

Output

```
----------
problrisolti = {1,2,3,4,6,9}
destinazionefondiA=14
destinazionefondiB=5
destinazionefondiC=1
----------
problrisolti = {1,2,3,4,6,13}
destinazionefondiA=14
destinazionefondiB=5
destinazionefondiC=1
```

```
----------
problrisolti = {1,2,4,5,6,7,9,12}
destinazionefondiA=3
destinazionefondiB=14
destinazionefondiC=3
----------
problrisolti = {1,2,4,5,6,7,9,13}
destinazionefondiA=3
destinazionefondiB=14
destinazionefondiC=2
----------
problrisolti = {1,4,5,6,9,10,13}
destinazionefondiA=1
destinazionefondiB=9
destinazionefondiC=10
----------
==========
Finished in 420msec
```

Bisogna tener conto solo delle 4 righe che precedono "Finished in 420 msec", fra questo la voce *problrisolti* rappresenta l'elenco dei problemi che conviene risolvere con quel budget, le seguenti righe indicano quanti fondi andrebbero assegnati ad ogni classe per poter risolvere i problemi appena elencati.

Il risultato rispetta, le nostre ipotesi infatti il risolutore prova più soluzioni fino ad arrivare a quella migliore che utilizza tutto il budget a disposizione, e in effetti la somma dei fondi destinati ai 3 cluster è pari a 20 e massimizza il punteggio-priorità.

Conclusioni

In conclusione, è necessario dare qualche precisazione sul metodo utilizzato per la realizzazione di questa proposta: potrebbe essere necessario apportare delle modifiche nel caso si procedesse alla fase di implementazione. Infatti, quello qui descritto è l'idea di un processo con relative tecniche e tecnologie che si potrebbero applicare per ottenere i risultati desiderati espressi all'inizio di questo report. Tuttavia, attraverso la ricerca bibliografica è stato possibile ricreare questo sistema ideale sfruttando delle operazioni che fossero scientificamente valide e non prive di fondamento.

Inoltre, per quanto riguarda il codice prodotto si precisa che quella fornita è un'implementazione del problema ideale che quindi è semplificato rispetto alla formulazione del problema reale; nonostante questa differenza abbiamo ritenuto interessante la realizzazione di questo modello funzionante come base di partenza per aggiornamenti successivi.

Infine, ribadiamo come un sistema simile potrebbe semplificare l'avvicinamento dei cittadini alle istituzioni o alle forze politiche in un momento in cui l'emergenza COVID-19 ha indirizzato quest'ultime a confrontarsi con il tema di assegnamento di risorse.

Il sistema ideale proposto si presta a innumerevoli estensioni, una di quelle pensate è la possibilità di utilizzare questo sistema con opportune modifiche per la gestione delle risorse dell'amministrazione in carica (e non di chi corre per le elezioni) il che richiederebbe l'inserimento di parecchia conoscenza della materia amministrativa.

Bibliografia

1. Istat 2019, pagina 89 “Politica e istituzioni”, https://www.istat.it/it/files//2019/12/6.pdf
2. Istat 2019, “La partecipazione politica in Italia” https://www.istat.it/it/archivio/244843
3. M. Gjoka, M. Kurant, C. T. Butts and A. Markopoulou, "Walking in Facebook: A Case Study of Unbiased Sampling of OSNs," *2010 Proceedings IEEE INFOCOM*, San Diego, CA, 2010, pp. 1-9, doi: 10.1109/INFCOM.2010.5462078.
4. Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement* (*IMC '07*). Association for Computing Machinery, New York, NY, USA, 29–42. DOI:https://doi.org/10.1145/1298306.1298311
5. Duen Horng Chau, Shashank Pandit, Samuel Wang, and Christos Faloutsos. 2007. Parallel crawling for online social networks. In *Proceedings of the 16th international conference on World Wide Web* (*WWW '07*). Association for Computing Machinery, New York, NY, USA, 1283–1284. DOI:https://doi.org/10.1145/1242572.1242809
6. Mfenyana, Sinesihle. (2013). Development of a Facebook Crawler for Opinion Trend Monitoring and Analysis Purposes: Case Study of Government Service Delivery in Dwesa. International Journal of Computer Applications. Volume 79. 32-39.
7. H. Isah, P. Trundle and D. Neagu, "Social media analysis for product safety using text mining and sentiment analysis," 2014 14th UK Workshop on Computational Intelligence (UKCI), Bradford, 2014, pp. 1-7, doi: 10.1109/UKCI.2014.6930158.
8. Li, X.F., Li, D., 2013. Sentiment Orientation Classification of Webpage Online Commentary Based on Intuitionistic Fuzzy Reasoning. Applied Mechanics and Materials 347–350, 2369–2374. https://doi.org/10.4028/www.scientific.net/amm.347-350.2369
9. J. Akaichi, Z. Dhouioui and M. J. López-Huertas Pérez, "Text mining facebook status updates for sentiment classification," 2013 17th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, 2013, pp. 640-645, doi: 10.1109/ICSTCC.2013.6689032.
10. https://sites.google.com/site/miningtwitter/mining-viz
11. Hanhoon Kang, Seong Joon Yoo, Dongil Han,Senti-lexicon and improved Naïve Bayes algorithms for sentiment analysis of restaurant reviews, Expert Systems with Applications,Volume 39, Issue 5,2012,Pages 6000-6010,ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2011.11.107.(http://www.sciencedirect.com/science/article/pii/S0957417411016538)
12. C. Troussas, M. Virvou, K. J. Espinosa, K. Llaguno and J. Caro, "Sentiment analysis of Facebook statuses using Naive Bayes classifier for language learning," IISA 2013, Piraeus, 2013, pp. 1-6, doi: 10.1109/IISA.2013.6623713.
13. Huang B., Yang Y., Mahmood A., Wang H. (2012) Microblog Topic Detection Based on LDA Model and Single-Pass Clustering. In: Yao J. et al. (eds) Rough Sets and Current Trends in Computing. RSCTC 2012. Lecture Notes in Computer Science, vol 7413. Springer, Berlin, Heidelberg.
14. Z. Indra, N. Zamin and J. Jaafar, "A clustering technique using single pass clustering algorithm for search engine," 2014 4th World Congress on Information and Communication Technologies (WICT 2014), Bandar Hilir, 2014, pp. 182-187, doi: 10.1109/WICT.2014.7077325.
15. Y. Xiaolin, Z. Xiao, K. Nan and Z. Fengchao, "An improved Single-Pass clustering algorithm internet-oriented network topic detection," *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, Beijing, 2013, pp. 560-564, doi: 10.1109/ICICIP.2013.6568138.