Università degli studi di Milano-Bicocca

Big data in Business, economics and society

Progetto Finale

FEDERLEGNO

DOCUMENTO TECNICO

Autori:
Beatrice Fumagalli - 784549 - b.fumagalli9@campus.unimib.it
Nicholas Missineo – 791050 – n.missineo@campus.unimib.it
Francesco Simoncelli - 834313 – f.simoncelli@campus.unimib.it
Beatrice Somaschini – 789554 - b.somaschini1@campus.unimib.it

Federlegno_code_python_select.ipynb

In questa sezione viene documentato il metodo utilizzato per il pre-processing del Dataset iniziale 'dataset.csv' al fine di selezionare il campione di aziende sul quale effettuare le successive analisi.

Prerequisiti

Per eseguire il codice all'interno del file "Federlegno_code_python_select.ipynb" è necessario aver installato: Anaconda 5.2 (Python 3.6 version)

Preparazione all'ambiente

All'interno della working directory di Anaconda creare un cartella ed estrarre al suo interno il contenuto dell'archivio " Federlegno_project "

Spiegazione

Il codice inizia con il caricamento delle librerie necessarie allo sviluppo della parte di pre-processing del dataset.
Il *dataset* iniziale 'dataset.csv' viene caricato attraverso la funzione ***read_csv*** della liberia ***Pandas***[1], precedentemente importata richiamandola **pd** e viene convertito in un *DataFrame* chiamato **data**.
Il *pre-processing* del *Dataset* iniziale prevede pochi passaggi, in quanto già in qualità.
Si inizia con la rinomina delle colonne del *Dataset* attraverso il comando ***rename***, per facilitarne il richiamo nei passaggi successivi.
Si indica con '>' il passaggio dal vecchio al nuovo nome dell'attributo. Vengono rinominati i seguenti attributi:

- Indirizzo sede legale Regione > regione
- SISTEMA PREVALENTE > sistema
- Ragione sociale > azienda
- Provincia > provincia
- ultimo Bilancio Ricav delle vendite > fatturato
- Partita IVA > p.iva

[1] https://pandas.pydata.org/

Successivamente le aziende vengono divise in base al loro fatturato. Viene appositamente creata la funzione '**assegna_dimensione**' che riceve in *input* un ***value.*** In base al *range* di valori entro il quale si trova il *value* in ingresso la funzione ritorna una stringa a cui è stata assegnata una dimensione corrispettiva: 'micro', 'piccola', 'media' o 'grande'. I *range* sono stati decisi in base ai fondamenti di economia aziendale ricercati[2]. La funzione viene poi applicata al *Dataset* attraverso il comando ***apply***(). In questo caso la funzione prende in *input* il valore della colonna "fatturato" di ogni azienda e per ciascun valore, in base al *range* entro cui si trova, inserisce in una nuova colonna chiamata 'dimensione' la dimensione corrispettiva.

Il *DataFrame* è ora pronto per selezionare il campione di aziende sul quale effettuare le successive analisi. Per una questione di tempistiche e volume di dati, ma per permettere un confronto sufficientemente informazionale è stato concordato di concentrarsi sulle aziende "brianzole", ossia della provincia di Monza e Brianza di dimensioni medie e grandi dei tre sistemi in cui erano disponibili, ossia arredamento, ufficio e illuminazione.
Pertanto è stata creata una serie di *DataFrame* con il campionamento graduale degli attributi. Il campionamento è stato graduale per stabilire man mano se campionare ulteriormente ed i criteri.

- az_lomb: contiene tutte le aziende con sede nella regione Lombardia
- az_lomb_mb: contiene tutte le aziende con sede in provincia di Monza e Brianza
- az_lomb_mb_medie_grandi: contiene tutte le aziende con sede in provincia MB e di dimensioni medie o grandi.

La tabella **'az_lomb_mb_medie_grandi'** rappresenta il campione delle 26 aziende su cui verranno svolte le successive fasi per l'analisi.
Si procede con lo *Scraping* delle pagine *Homepage* e "chi siamo" del sito *web* di ciascuna delle aziende appena selezionate attraverso il codice R 'Federlegno_code_scraping' inserito nell'archivio.

Federlegno_code_scraping.R

In questa sezione viene documentato il metodo utilizzato per lo *scraping* attraverso cui è possibile ricavare i testi desiderati per la successiva fase di analisi.

[2] http://www.economiaziendale.net/lezioni/azienda/aziende_piccole_medie_grandi.htm

Prerequisiti

Per eseguire il codice all'interno del file " Federlegno_code_scraping.R" è necessario aver installato: R 3.5.1 (potrebbe funzionare anche con versioni precedenti)

Preparazione all'ambiente

Settare la working directory in modo da poter salvare in modo agevole i file csv che si otterranno come output.

Spiegazione codice

Per prima cosa vengono importate le librerie necessarie per il funzionamento del codice: **rvest** e **tidyverse**.

La fase di *scraping*, sebbene aiutata dal codice presentato, non è del tutto automatizzata, infatti come primo passaggio è necessario settare l'*url* della pagina che si desidera che il codice vada ad esplorare.

Per ognuna delle aziende considerate e campionate, di cui si può ricavare la ragione sociale all'interno del file precedentemente creato: *"az_lomb_mb_medie_grandi",* ricercare il sito *web*, una volta individuato copiarne l'*url* dell'*homepage* e incollarlo nel codice nell'oggetto chiamato ***base_url***.

Questo viene quindi processato dalla funzione ***read_html***: la funzione legge la pagina *web* e la salva in un oggetto chiamato ***webpage***.

Una volta creato questo oggetto, si procede con l'estrazione vera e propria del testo, ciò avviene attraverso l'impiego di due funzioni: la prima è ***html_nodes***, che richiede come argomenti l'oggetto *webpage* e l'estensione di testo che si intende estrarre. È possibile dedurre l'estensione attraverso l'ispezione della pagina *web*, per aiutarsi in questa operazione è stata scaricata l'estensione di *Chrome*: ***SelectorGadget***, questo strumento consente di far apparire l'estensione della porzione di testo selezionata con il mouse (ad esempio: .p, .h1, ecc…) e delle porzioni simili. Alternativamente si può procedere con l'ispezione della pagina andando a cliccare il tasto destro del mouse sulla pagina *web* e andando a selezionare con il cursore la porzione di testo a cui si è interessati.

L'estensione del testo va inserita come testo nella funzione ***html_nodes***, quindi va inserita tra virgolette, ad esempio ".h1". L'*output* di questa funzione viene salvato in un oggetto, a cui diamo ad esempio il nome di *'home1'*.

La seconda funzione impiegata è ***html_text***, che trasforma le estensioni precedentemente salvate in *home1* in testo. Gli argomenti di questa funzione sono: l'oggetto home1 e il parametro trim=T, quest'ultimo consente di tagliare ad eventuali spazi ed a capo.

Tutta la funzione ***html_text*** va a sua volta passata all'interno della funzione ***as.character***, che consente ad R di interpretare l'*output* della funzione ***html_text*** come lista di caratteri. L'*output* di questa funzione viene salvato in un oggetto che per comodità viene nominato **h1**.

A volte il testo selezionato potrebbe richiedere un'ulteriore fase di pulizia da alcuni caratteri che non vengono eliminati dai passaggi precedenti, ad esempio "\n" , per fare questo, all'oggetto appena creato: h1, viene passata la seguente sequenza di funzioni: **unlist(strsplit(gsub("\n"," ",h1),split=" "))**. La funzione più interna: ***gsub***, si occupa di sostituire al carattere indesiderato uno spazio, la funzione successiva ***strsplit***, serve a dividere la componente di testo interna ad ogni spazio creandone una lista di caratteri, questa viene poi riportata al formato originale attraverso la funzione più esterna ***unlist***.

Il procedimento viene ripetuto per tutte le estensioni individuate nella *homepage*.
Successivamente si procede con il medesimo codice presentato ma andando a sostituire l'*url* della *homepage* con quello della pagina "Chi Siamo" dell'azienda presa in analisi. Si segnala che questa pagina non sempre assume questo nome, va quindi ricercata tra le possibili pagine: ad esempio "About us", "The company", etc.
Una volta copiato l'*url* della pagina descrittiva dell'azienda, lo si incolla nell'oggetto **base_url** e si segue con il procedimento sopra descritto.

Una volta salvati in vettori diversi (ad esempio quello nominato in precedenza: h1) tutte le porzioni di testo desiderate di entrambe le pagine le si salva in un unico vettore, questo si esegue semplicemente con il comando **c()**, e inserendo la lista di tutti gli oggetti; ad esempio: c(h1, h2, h3). Il vettore di *output* viene salvato in un oggetto chiamato con nome testo.

Il vettore testo viene quindi esportato in formato .csv attraverso la funzione ***write.csv.*** Il file esportato viene salvato nella directory inizialmente settata.

Ora il testo della singola azienda selezionata compare su un file .csv, interamente sulla prima colonna ma distribuito su più righe. Per rendere leggibile attraverso il codice successivo che si occuperà dell'operazione di analisi, si procede manualmente a trasferire tutto il testo in un programma di testo (attraverso un'operazione di copy-paste) per poi andare a copiare e incollare nuovamente il testo in questione in un'unica cella del file *excel* '*az_lomb_mb_medie_grandi.csv',* rispettivamente nella riga relativa all'azienda selezionata. La colonna contenente il testo viene rinominata 'homepage + chi_siamo'.

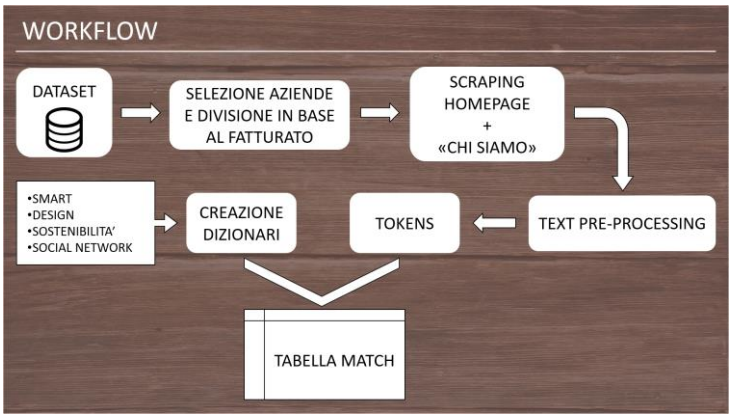
Il procedimento mostrato viene ripetuto per tutte e 26 le aziende in questione. Da evidenziare che se il sito dell'azienda non include una versione in inglese o addirittura l'azienda selezionata non presenta un sito web, allora nella colonna 'homepage + chi_siamo' si va ad inserire **NaN**.
Il file finale viene salvato in formato .csv e rinominato come: **scrape_fin.csv.**

Nota aggiuntiva

Si vuole sottolineare che il procedimento utilizzato è molto macchinoso e poco automatizzato in diversi passaggi, specialmente in fase di salvataggio del testo. Questo non è stato un problema per il lavoro svolto dato il numero esiguo di aziende analizzate. Se si dovesse disporre di un campione più consistente da analizzare e di tempo necessario, è vivamente consigliato di cercare un'automatizzazione del processo molto più adeguata al compito da svolgere.

Federlegno_code_python_text_mining.ipynb

In questa sezione viene documentato il metodo utilizzato per la fase di *Text Mining* del testo ottenuto attraverso il processo di *Scraping* della fase precedentemente descritta.



Spiegazione codice

Il codice inizia con il caricamento delle librerie necessarie allo sviluppo della parte di *text pre-processing*.

Il *dataset* 'scrape_fin.csv' viene caricato attraverso la funzione ***read_csv*** della liberia ***Pandas*** , precedentemente importata richiamandola ***pd*** e viene convertito in un *DataFrame* chiamato ***homepage***.

La fase di *Text pre-processing* inizia con la rimozione delle aziende per le quali non è stato possibile effettuare lo *scraping* e pertanto presentano valore '*NaN*' in corrispondenza dell'attributo "homepage + chi siamo", che si riferisce al testo precedentemente estratto. Ciò avviene attraverso il comando ***df.dropna()*** in cui viene specificato l'attributo in cui verificare la presenza di valori *NaN* ed eventualmente *droppare* il *record*. Solamente due aziende presentano valore *NaN* e pertanto si passa da 26 a 24 aziende. A queste può essere eseguita la fase di pulizia del testo. Viene creata appositamente una funzione chiamata ***'text_preproc'*** al fine di eseguire tutti i passaggi di *Text Normalization* scelti. Quest'ultima include:

- conversione di tutte le lettere in minuscolo
- rimozione della punteggiatura
- rimozione di eventuali spazi bianchi
- rimozione di numeri
- rimozione delle stop words in quanto prive di significato semantico
- rimozione dei tags
- Stemming
- Tokenization

In particolare si è deciso di effettuare il processo di Stemming, ossia la riduzione della forma flessa di una parola alla sua forma radice. Questo perché la medesima parola potrebbe essere presente all'interno del testo in svariate forme, basti pensare a *automation, automatic, automate.* Ai fini dell'analisi è importante che tutte le parole, indipendentemente dalla loro forma, siano trovate e contate. Infine il metodo ***'preprocess_string'*** applica la lista di filtri sopra descritti al testo in *input* e ritorna dei *tokens* puliti. La funzione viene poi applicata all'attributo 'homepage + chi siamo'

attraverso il comando ***apply***() ed i *tokens* puliti vengono messi in una nuova colonna chiamata "*text_clean*".
La colonna con il testo originale non viene *droppata* ma viene semplicemente rinominata in "*homepage*" per facilitarne il richiamo successivo.
Il testo è ora pronto per la fase di *Topic Modeling*. Quest'ultima inizia con la creazione di quattro differenti dizionari:

- ***Smart_keys:*** contenente le parole che si riferiscono al *topic* **Innovazione**
- ***Design_keys:*** contenente le parole che si riferiscono al *topic* **Design**
- ***Env_keys:*** contenente le parole che si riferiscono al *topic* **Sostenibilità**
- ***Social_net_keys:*** contente le parole che si riferiscono al *topic* **Social Network**

Le parole all'interno di ciascun dizionario possono essere aggiunte, rimosse o modificate. A ciascuna parola presente in ciascun dizionario è stato effettuato il processo di *Stemming* al fine di garantire un corretto *match* con i *tokens* puliti precedentemente ottenuti. Per il medesimo motivo i dizionari sono stati creati in lingua inglese.

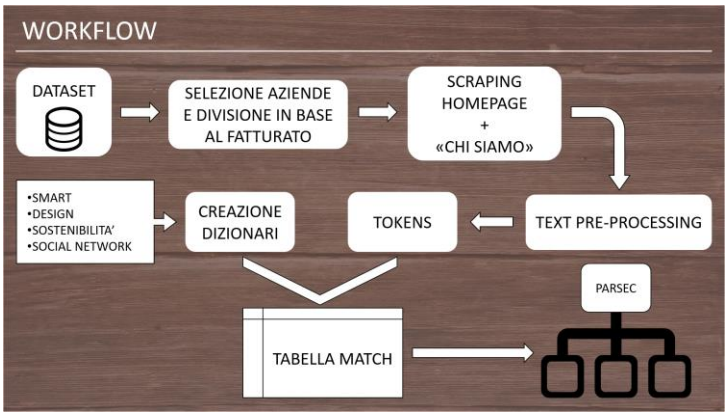
Vengono ora create due funzioni per ciascun *topic*, per un totale di 8 funzioni. La prima di ciascuna sezione ha il compito di controllare se nel testo pulito della colonna '*homepage'* per ciascuna azienda sia presente almeno una *keyword* del *topic* considerato. In tal caso viene assegnato valore *True* in una nuova colonna per ciascun *topic* ('smart', 'design', 'env', 'soc_net'), altrimenti *False*.
La seconda funzione invece riporta in una nuova colonna per ciascun *topic* ('smart_words', 'design_words', 'env_words', 'soc_net_words') tutte le parole del *topic* che sono state riscontrate nel testo pulito per ciascuna azienda. Questo consente il passaggio successivo, ossia il ***Count*** delle parole di ciascun *topic* trovate per ciascuna azienda. Il *Count* avviene attraverso un semplice *check* con ciclo *for* ed una successiva **sum()**, al fine di ottenere il numero di occorrenze totali per ciascun *topic* in ciascuna azienda; successivamente la *sum* viene inserita in una nuova colonna, una per ogni *topic*.

Selezionando solamente gli attributi di interesse si ottiene la tabella finale 'tab_finale_parsec', pronta per effettuare il *ranking* con *Parsec* in R. Questa tabella è formata dai seguenti attributi:

- **Azienda:** nome dell'azienda
- **Sistema:** arredamento, illuminazione o ufficio
- **Dimensione:** media o grande
- **Smart_count:** numero di occorrenze delle parole del dizionario del *topic Smart* all'interno del testo
- **Design_count:** numero di occorrenze delle parole del dizionario del *topic Design* all'interno del testo
- **Sust_count:** numero di occorrenze delle parole del dizionario del *topic Sustainability* all'interno del testo
- **Soc_net_count:** numero di occorrenze delle parole del dizionario del *topic Social Network* all'interno del testo

Federlegno_code_ranking.R

In questa sezione viene documentato il metodo utilizzato per la fase di *Ranking* delle aziende basato sulla sensibilità di quest'ultime rispetto ai temi trattati da ciascun *topic*, sensibilità misurata in base alla frequenza delle occorrenze.



Spiegazione codice

Il codice inizia con l'importazione del dataset creato nella precedente sezione: 'tab_finale_parsec.csv'.

L'importazione viene fatta attraverso la funzione ***read.csv*** specificando come argomenti head=T e sep=";" e salvandolo con il nome **dati**.

Per l'esecuzione della funzione di ranking è necessaria l'istallazione e il richiamo della libreria ***parsec***.

Per l'utilizzo della funzione è necessario salvare i dati in una matrice o *DataFrame* numerico; viene quindi selezionata la porzione di *DataFrame* dati contente le variabili con i conteggi. La selezione avviene indicando l'indice delle colonne. Viene salvato un nuovo *DataFrame* con il nome di **df** e a questo viene imputata la ragione sociale (univoca) delle aziende con nomi di riga.

Successivamente dal *dataset* df vengono estratti i profili univoci attraverso la funzione ***pop2prof***,salvandoli in un ottetto con nome **pp**. Si sottolinea che è stato scelto in questo caso di selezionare unicamente i profili effettivamente esistenti nel *dataset*, un'altra versione che va invece a settare tutti i possibili profili è resa disponibile da un'altra funzione del pacchetto.

Si precisa che con profilo si intende la combinazione dei valori assunti dalle variabili per ciascuna riga.

Per riuscire a risalire alla ragione sociale dell'azienda caratterizzata da un certo profilo (si noti che un profilo può appartenere a più aziende) si crea un vettore vuoto nominandolo **id**. Successivamente viene creato un ciclo *for* che per ogni riga del *dataset* df collassa in una stringa i valori delle variabili e aggiunge la stringa al vettore id. A questo punto il vettore delle frequenze dei profili, accessibile mediante il comando ***pp$freq***, viene trasformato in un oggetto *DataFrame* e se ne richiamano i nomi di riga (ovvero i profili stessi) attraverso la funzione ***rownames***, questi vengono salvati in un vettore chiamato **n**. A questo punto viene ricavato il vettore ***index*** che, attraverso la funzione ***match***, va a trovare l'indice della posizione di ogni elemento del vettore id all'interno del vettore n.

Se si desidera avere una visualizzazione dei profili attraverso il **Diagramma di Hasse** è possibile semplicemente applicare la funzione ***plot*** all'oggetto pp trasformato attraverso la funzione

getzeta(). La visualizzazione risulta chiara se i profili sono pochi, se si sta lavorando con tanti profili probabilmente si otterrà un diagramma poco comprensibile.

A questo punto viene creata la matrice di **mutal ranking probability**. Questo avviene attraverso la funzione **MRP** del pacchetto ***parsec***. La funzione richiede come argomento un oggetto di classe ***incidence***, che si ottiene applicando nuovamente la funzione ***getzeta*** ai profili salvati nell'oggetto pp. Il secondo argomento richiesto è il metodo di calcolo: avendo pochi profili da analizzare si è optato per il metodo ***exact***; se si avesse un cospicuo numero di profili da analizzare è consigliabile settare il parametro con metodo di tipo ***mcmc*** o ***approx***.
L'output della funzione è stato salvato in una matrice chiamata ***mrp***.

Il vettore di *ranking* dei profili è stato ottenuto estraendo la prima colonna della matrice v, ottenuta dalla decomposizione spettrale della matrice di mutual ranking probability. Ciò è stato possibile applicando la funzione ***svd()*** alla matrice **mrp** e richiamandone unicamente la matrice v attraverso il comando **$v**, da questa è stata selezionata unicamente la prima colonna. Il risultato è salato in un vettore chiamato ***rank***. La lunghezza di questo vettore coincide con il numero di profili.

Al *dataset* originale viene aggiunta la variabile di *ranking* andando opportunamente ad abbinare il valore di *ranking* caratteristico per ciascun profilo al profilo presentato da ciascuna azienda. Questo viene fatto attraverso il vettore precedentemente creato: ***index***.
Infine il *dataset* viene ordinato in base al *ranking* decrescente con la funzione ***order***, settando come argomento *decreasing*=T.

I dati vengono quindi esportati attraverso la funzione ***write.csv()*** con il nome di '***dati_mg_ranked.csv'.***

Appendice analisi

In appendice viene inoltre riportato il codice utilizzato per l'analisi del dataset.
Per l'analisi grafica le librerie necessarie sono ***ggplot2*** e ***RColorBrewer***. Con la funzione ***ggplot*** vengono creati due *boxplot* che mettono in relazione il *ranking* prima con il sistema di produzione e poi con la dimensione aziendale.
Poi sempre con ***ggplot*** viene creato un *Bar Chart* con le aziende e il livello di *ranking* raggiunto da ciascuna.

Infine viene riportato il codice per eseguire il *test* di ***Kruskal-Wallis***: questo viene fatto con la funzione ***kruskal.test()*** il cui argomento deve essere una lista contenente i dati numerici divisi per i gruppi.