

NLP Clustering su Yelp

Progetto di Intelligenza Artificiale - A.A. 2020/21
Riccardo Preite

SOMMARIO

Introduzione

Comprensione del problema

Obiettivo

Dataset

Tecnologie

Implementazione

Difficoltà

Risultati

Conclusioni



Introduzione



Con **NLP** si identifica il processo di elaborazione del linguaggio naturale in modo da poter rimuovere l'ambiguità che troviamo nei linguaggi umani attraverso 4 analisi:

1. Analisi lessicale: valori delle colonne -> token
2. Analisi grammaticale: token->valori codificati
3. Analisi sintattica: valori codificati -> nuove colonne
4. Analisi semantica: nuove colonne -> clustering

Python offre molteplici librerie adatte ad organizzare ed elaborare vari tipi di dati, tra cui pandas e numpy.



Comprendere il problema

Trovare un modo per mappare le descrizioni nel linguaggio naturale in informazioni comprensibili dalla macchina.

- One-hot Encoding;
- Clustering.



Obiettivo del progetto

Suddividere le attività commerciali raccolte da Yelp per:

- Posizione;
- Categorie;
- Attributi.

Estrapolando così informazioni importanti dalle descrizioni delle attività commerciali.



Dataset

01

Più di 160000 diverse attività commerciali.

02

Più di 1000 colonne che descrivono queste attività.

03

Veramente tante (forse anche troppe!) colonne con molti valori NaN.

Tecnologie

Python3 mette a disposizione:

- Pandas: Per leggere i dati dal csv;
- Numpy: Per trasformare i DataFrame in vettori;
- Scikit: Per manipolare vettori Numpy sui quali fare i cluster.



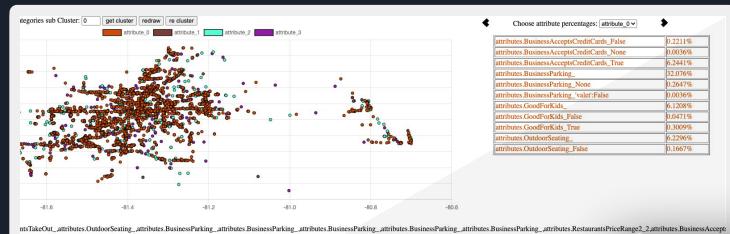
Implementazione

Il software è compatibile con:

Windows;

Linux;

Mac OS.



Snapshot del codice

```
def k(model,data,cluster):
    #Predict only on first two columns, lat&lon
    label = model.fit_predict(data[0:,:2])
    u_labels = np.unique(label)

    return model,data,label,u_labels

def categoriesk(model,data,start_categories_col):
    #Predict only on last columns, categories
    label = model.fit_predict(data[0:,categories_col:])
    u_labels = np.unique(label)

    return data,label,u_labels

def attrk(model,data,start_categories_col):
    #Predict only on middle columns, attributes
    label = model.fit_predict(data[0:,2:start_categories_col])
    u_labels = np.unique(label)

    return data,label,u_labels
```

```
80     kmeans,df,label,u_labels = k(kmeans,data=np_data,cluster=cluster)
81     print("finished clustering from lat&lon")
82     del np_data
83
84     print("starting sub clustering...")
85     for i in u_labels:
86
87         print("start "+str(i)+" cluster")
88         if len(df[label == i]) == 1:
89             print("starting categories clustering")
90             df_cat,label_cat,u_labels_cat = categoriesk(kmeans_sub_1,data=df[label == i],col=start_categories_index)
91             categories_json = {}
92
93             for lab in u_labels_cat:
94                 print("starting attribute sub clustering")
95                 df_attr,label_attr,u_labels_attr = attrk(kmeans_sub_1,data=df_cat,col=start_categories_index)
96                 print("fixing data, removing 0 value...")
97                 categories_json = translateCluster(categories_json,lab,u_labels_attr,df_attr,label_attr,columns)
98
99                 saveFile(categories_json,"json/lat&lon_"+str(i)+".json")
100
101             else:
102                 print("starting categories clustering")
103
104             df_cat,label_cat,u_labels_cat = categoriesk(kmeans_sub_cate,data=df[label==i],col=start_categories_index)
105             categories_json = {}
106
107             for lab in u_labels_cat:
108                 print("starting attribute sub clustering lat&lon: "+str(i) + " categories: " + str(lab))
109                 df_attr,label_attr,u_labels_attr = attrk(kmeans_sub_attr,data=df_cat[lab==label_cat],col=start_categories_index)
110                 print("fixing data, removing 0 value...")
111                 categories_json = translateCluster(categories_json,lab,u_labels_attr,df_attr,label_attr,columns)
112                 print("fixed data, ended attribute sub clustering lat&lon: "+str(i) + " categories: " + str(lab))
113
114                 saveFile(categories_json,"json/lat&lon_"+str(i)+".json")
115
116             print("end "+str(i)+" cluster")
```

Difficoltà e limiti

L'hardware a disposizione è molto limitato:

- 16GB RAM 3000Mhz;
- 3.7Ghz di clock.

Linux, al contrario di Windows e Mac OS, non riesce ad eseguire con la stessa fluidità.



A causa dei molteplici valori che si hanno non è possibile avere una visione completa di tutte le caratteristiche che le attività hanno.



Risultati

- 9 cluster per le aree metropolitane
- 15 cluster per le categorie;
- 4 categorie per gli attributi.



Precisione cluster
posizione



Precisione cluster
categorie



Precisione cluster
attributi



Conclusioni e sviluppi futuri

Difficoltà:

- Scarso hardware;
- Molteplicità dei valori e poca chiarezza nel visualizzare l'insieme;
- Grandezza spropositata dei possibili cluster;
- Impossibile definire il numero dei cluster a priori;

Sviluppi futuri:

- Migliorare hardware;
- Testare diversi numeri di cluster;



Grazie per
l'attenzione!

