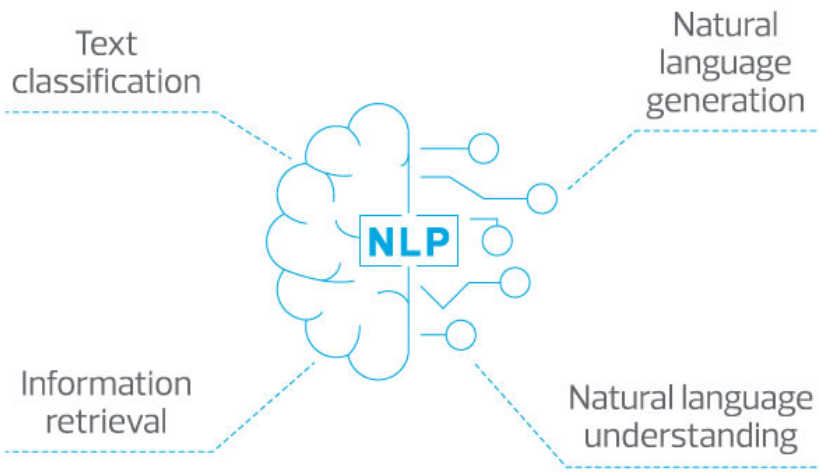
Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

Progetto di Intelligenza Artificiale



Professore:
Chiar.mo Prof.
Maurizio Gabbrielli

Presentata da:
Riccardo Preite

Anno Accademico 2020/21

Indice

Capitolo 1

Introduzione ed obiettivo

Con elaborazione del linguaggio naturale, in inglese Natural Language Processing (**NLP** [1]) identifichiamo il processo di analisi lessicale, grammaticale, sintattica e semantica, delle informazioni scritte in un lingua. La suddivisione descritta sopra, molto simile a quella effettuata sui linguaggi di programmazione, permette di discriminare le ambiguità che il linguaggio umano contiene.

- **Analisi lessicale:** Scomposizione di espressioni in token, nel nostro caso i valori delle colonne;
- **Analisi grammaticale:** Associazione di un valore ai token effettuato con il dummy_encoding;
- **Analisi sintattica:** Sempre grazie al dummy encoding è stato possibile creare una colonna per ogni token trovato dove viene riportato 1 se quella determinata riga aveva quella proprietà 0 altrimenti;
- **Analisi semantica:** L'analisi semantica assegna un significato alla struttura sintattica creata al punto prima, questo task viene effettuato tramite un algoritmo di clustering che agglomera le colonne per vicinanza dei valori.

Python è un linguaggio di programmazione che offre molteplici librerie atte al Machine Learning e alla conversione dei dataset in formati adatti agli algoritmi di ML come ad esempio vettori numpy. Inoltre pandas mette a disposizione una routine per poter trasformare valori letterali in valori binari senza perdita di significato (dummy_encoding).

1.1 Obiettivo

L'obiettivo che si vuole raggiungere in questo progetto consiste nel suddividere le attività commerciali presenti nel dataset fornito da Yelp prima per posizione (latitudine e longitudine) e poi per criteri di similitudine.

Capitolo 2

Dataset

Il dataset che viene fornito da Yelp [2] è molto vario e comprende raccolte dati sugli utenti, review e sulle attività commerciali, nel nostro caso ci siamo soffermati su quest'ultime. Il dataset viene fornito come json ma per questioni di comodità ed adattabilità è stato convertito in formato csv tramite lo script che fornisce Yelp, inizialmente si è dovuto applicare qualche modifica a questo script in quanto non era possibile eseguirlo. La comodità che fornisce il formato csv rispetto al json è che i sotto dizionari presenti nel json vengono scomposti e trasformati in colonne di un file csv. Questo fornisce un input perfetto per il dummy_encoding che per ogni valore letterale delle colonne crea una nuova colonna.
Es:

La voce Parking può assumere 3 valori:

- True
- False
- None o stringa vuota

Invece che avere la colonna Parking —- False si avranno tre colonne:

- Parking_True —- 0
- Parking_False —- 1
- Parking_None —- 0
- Parking_ —- 0

Inoltre prima di iniziare a lavorare con l'algoritmo di clustering [3] è stata fatta una suddivisione del dataset rimuovendo colonne inutili, come ad esempio il business_id o gli orari di apertura dei negozi ed inoltre sono state eliminate le colonne che riportavano meno di 50000 valori completi. Si è deciso di non rimuovere le righe che riportavano valori vuoti o None in quanto si sarebbe ridotto drasticamente il dataset.

Capitolo 3

Implementazione

La parte relativa al ML è scritta in Python3 attraverso l'utilizzo di 3 principali librerie:

- Pandas;
- Numpy;
- Scikit learn.

3.1 Pandas

Pandas viene utilizzato per leggere il csv in input, rimuovere le colonne inutili e una volta pulito il dataset applicare il dummy_encoding descritto nella sezione precedente. In seguito viene sistemato ulteriormente il dataset posizionando nelle prime due colonne la latitudine e longitudine dei punti, poi le colonne relative ad alcuni attributi dei luoghi (Es parcheggio) e infine le categorie che possono assumere le attività (Ristoranti, parrucchieri ecc).

3.2 Numpy

Una volta sistemati questi dati vengono trasformati in un numpy array pronto per essere dato in pasto a Scikit learn.

```
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
```

Figura 3.1: Routine per i diversi cluster applicati.

3.3 Scikit

Per effettuare il clustering dei punti si è deciso di utilizzare l'algoritmo K-Means fornito da Scikit-learn [4]. Inizialmente si era optato per DB-scan ma si è dovuto scartare in quanto ha una complessià maggiore e con la grandezza del dataset e la scarsità di risorse non si è rilevato adeguato. In seguito ad i vari test il numero di cluster adeguato per la suddivisione per posizione è pari a 9, tra questi troviamo un cluster che contiene un solo punto in quanto è un outlier del dataset.

Per quanto riguarda la suddivisione per criteri di similitudine la procedura seguita è stata la seguente:

- Suddivisione del dataset per i 9 cluster della posizione;
- Per ognuno dei cluster trovati dal punto precedente si effettua una clusterizzazione per le categorie. Si è impostato il numero di cluster a 15 sia perchè semprerebbe un numero adatto per suddividere le attività sia perchè dai test effettuati con un numero maggiore di cluster non si riesce ad arrivare ad una terminazione con successo ma solamente ad un out of memory;
- Per ogni cluster sulle categorie che viene trovato viene suddiviso ulteriormente in 4 cluster segnati dagli attributi che descrivono le attività. Si è scelto questo numero perchè gli attributi rimasti dopo aver rimosso le colonne oltre la soglia di valori NaN è pari a 7 inoltre k-means a volte trova 4 cluster invece di 5, come risultato da alcuni test, quindi la decisione è stat presa anche per evitare di avere ripetizione dei punti.

Quando viene finito di analizzare ogni cluster per posizione, quindi dopo aver applicato il clustering prima per categorie e poi per attributi su quell'area metropolitana questo viene salvato su un file come si può notare nella riga 113 e 99 della Figura 3.2. Per visualizzare i dati questi vengono uniti in un unico file al momento della richiesta.
Si è dovuta spezzare questa procedura per non andare out of memory.

```
    kmeans,df,label,u_labels = k(kmeans,data=np_data,cluster=cluster)
    print("finished clustering from lat&lon")
    del np_data

    print("starting sub clustering...")
    for i in u_labels:

        print("start "+str(i)+" cluster")
        if len(df[label == i]) == 1:
            print("starting categories clustering")
            df_cat,label_cat,u_labels_cat = categoriesk(kmeans_sub_1,data=df[label == i],col=start_categories_index)
            categories_json = {}

            for lab in u_labels_cat:
                print("starting attribute sub clustering")
                df_attr,label_attr,u_labels_attr = attrk(kmeans_sub_1,data=df_cat,col=start_categories_index)
                print("fixing data, removing 0 value...")
                categories_json = translateCluster(categories_json,lab,u_labels_attr,df_attr,label_attr,columns)

            saveFile(categories_json,"json/lat&lon_"+str(i)+".json")

        else:
            print("starting categories clustering")

            df_cat,label_cat,u_labels_cat = categoriesk(kmeans_sub_cate,data=df[label==i],col=start_categories_index)
            categories_json = {}

            for lab in u_labels_cat:
                print("starting attribute sub clustering lat&lon: " +str(i) + " categories: " + str(lab))
                df_attr,label_attr,u_labels_attr = attrk(kmeans_sub_attr,data=df_cat[lab==label_cat],col=start_categories_index)
                print("fixing data, removing 0 value...")
                categories_json = translateCluster(categories_json,lab,u_labels_attr,df_attr,label_attr,columns)
                print("fixed data, ended attribute sub clustering lat&lon: " +str(i) + " categories: " + str(lab))
            saveFile(categories_json,"json/lat&lon_"+str(i)+".json")

        print("end "+str(i)+" cluster")
```

Figura 3.2: Ciclo per calcolare i vari cluster.

Capitolo 4

Conlusioni

Il lavoro svolto sembra aver portato a dei buoni risultati decisamente meglio di quanto si prospettava di ottenere dai primi test fatti e dall'hardware a disposizione. Ovviamente non è possibile definire con certezza quanto il clustering per categorie/attributi sia specifico in quanto i dati che vengono mostrati sono veramente tanti e non è possibile controllarli a mano. Un lavoro futuro che si potrebbe fare riguardo a questo progetto sarebbe quello di suddividere ulteriormente le categorie per cercare di ottenere cluster più distinti ma ovviamente dovrebbe essere accompagnato da un sistema con un numero di risorse decisamente maggiore. La macchina utilizzata per questo testing contava su 16GB di RAM ed un processore AMD con velocità di clock portata a 3.7Ghz da 3.2 Ghz inoltre il sistema è supportato da un SSD per permettere uno swapping delle pagine in memoria più rapido ma nonostante questo il tempo di esecuzione varia dai 20 ai 30 min nel peggiore dei casi. Inoltre è stato riscontrato un problema spiacevole, per qualche motivo ancora sconosciuto Linux non riesce a reggere l'esecuzione del codice crashando o bloccando l'intero sistema, al contrario con Windows e Mac OS si riesce ad eseguire tranquillamente il programma ed ad utilizzare la macchina nel frattempo.

Bibliografia

[1] Natural language processing. https://it.wikipedia.org/wiki/Elaborazione_del_linguaggio_naturale.

[2] Yelp dataset. https://www.yelp.com/dataset.

[3] Robert Tryon. Cluster analysis. https://it.wikipedia.org/wiki/Clustering, 1939.

[4] David Cournapeau. Scikit-learn library. https://scikit-learn.org/stable/, 2007.