Sistemi di raccomandazione content-based per articoli

Autori (gruppo 12): Marco Pellizzari 2054751, Greta Santarelli 2039102 , Sara Nisato 1198565.

Introduzione

Il programma utilizzato per confrontare l’efficacia dei metodi di raccomandazione *Nearest Neighbor Classifier* e *Bayes Classifier.*

Il progetto si sviluppa in due file, Sampler.py e Recommender\_System.py. Partendo dai dati del dataset MIND Small, i programmi permettono di estrarre un campione di user (e quindi di articoli) da utilizzare per effettuare il confronto. In particolare, Sampler.py è responsabile della parte “offline” della raccomandazione, quindi ottenimento del campione di user, ottenimento del contenuto degli articoli, del loro pre-processing e del calcolo degli score tf-idf per ogni termine. Il secondo file invece confronta i diversi metodi di raccomandazione sul campione prodotto. Sampler.py si trova nella directory “SAMPLER”, Recommender\_System.py in “RECOMMENDER\_SYSTEM”. Il link per ottenere il campione utilizzato nel progetto si trova a fine file.

Utilizzo

**Sampler.py**

Per utilizzare Sampler.py sono necessarie innanzitutto le librerie *random, time, nltk, pickle, requests, BeautifulSoup, argparse, multiprocessing e math.*

Per eseguire il programma è necessario scaricare alcuni file. Dall’Url in seguito è possibile scaricare il dataset MIND Small, dei file contenenti un campione di tutti i dati raccolti in un periodo di 5 settimane, divisi in training set e validation set.

DOWNLOAD -> <https://msnews.github.io/>

Questi sono i file che sono stati utilizzati nel progetto. MIND Small si divide in due set, il training set e il validation set, i cui dati sono suddivisi a loro volta in 2 file, uno relativo agli user behaviors e uno agli articoli. Questi 4 file vengono utilizzati per costruire il campione. La divisione in “Training set” e “Validation set” che viene proposta ***non*** è comunque quella che viene utilizzata. I 4 file sono necessari per avere tutti i dati dal periodo di 5 settimane.

*Eseguire il programma*

python Sampler.py --UTrain behaviors\_train.tsv --NTrain news\_train.tsv --UVal behaviors\_val.tsv --NVal news\_val.tsv

Argomenti:

* --UTrain: il file behaviors.tsv del training set MIND Small;
* --NTrain: il file news.tsv del training set MIND Small;
* --UVal: il file behaviors.tsv del validation set MIND Small:
* --NVal: il file news.tsv del validation set MIND Small.

È meglio rinominare i file in modo da distinguere quelli del training set e quelli del validation set, e collocarli nella stessa directory di Sampler.py.

Seguendo il link a fine file è possibile scaricare i file già rinominati, ready-to-use con l’istruzione riportata sopra.

*Parametri*

L’esecuzione del programma dipende da due parametri, specificati nelle prime righe del file Sampler.py: “nUsers” (di default 200), che determina la numerosità del campione, e “nNews” (di default 100), che determina la soglia di news che lo User deve aver cliccato e ignorato per entrare nel campione. È sufficiente modificare i due parametri per modificarne la relativa funzione.

*Output*

Il programma crea due dizionari, *Users\_sample.pkl* e *News.pkl* (il campione), che vengono salvati e utilizzati dal secondo file. Questi file vanno quindi spostati nella stessa directory in cui è salvato Recommender\_System.py.

**Recommender\_System.py**

Per utilizzare Recommender\_System.py sono necessarie le librerie *time*, *nltk*, *math*, *pickle*, *random*, *multiprocessing*, *numpy* e *tabulate*.

Per eseguire il programma è necessario collocare i file News.pkl e Users\_sample.pkl nella stessa directory del file.

*Eseguire il programma*

python Recommender\_System.py

*Parametri*

L’esecuzione del programma dipende da diversi parametri, specificati nelle prime righe del file:

* nFeat (di default 100), è il numero di features massime da utilizzare per la rappresentazione di ogni articolo (metodo tf-idf).
* retain (di default 0.5) indica la quantità di features da tenere quando viene fatta la riduzione con il metodo x2 (es 0.5 -> 50% delle feature per ogni articolo, da 100 a 50).
* top\_k (di default 20), necessario per la Nearest Neighbor Classification. È il numero di “vicini” da utilizzare per predire il rating per gli articoli del test set.
* clusterFactor (di default 0.1), indica quanti cluster realizzare nel caso si voglia introdurre il clustering nel Nearest Neighbor Classifier (0.1 -> 100 articoli vengono aggregati in 10 cluster).
* nIter (di default 4), il numero di iterazioni massime che vengono concesse all’algoritmo k-means per realizzare i cluster.
* top\_k\_cluster (di default 3), ha la stessa funzione di top\_k, nel caso in cui gli articoli del training set vengano raggruppati in cluster.

*Output*

La funzione di Recommender\_System.py è quella di confrontare, sul campione, l’efficacia della raccomandazione tramite Bayes Classifier e Nearest Neighbor Classifier. Questo viene fatto al variare dei valori di *alpha*, il parametro utilizzato per pesare diversamente i termini che compaiono in titolo, categoria e sottocategoria (nella funzione “main” è possibile modificare la lista associata alla variabile “alpha\_values” e specificare i valori per i quali vogliamo fare le raccomandazioni).

Durante l’esecuzione del programma viene chiesto all’utilizzatore quali metodi utilizzare: utilizzare sia il Bayes Classifier che il Nearest Neighbor Classifier restituirà delle statistiche riguardanti la raccomandazione per entrambi i metodi. Possono anche essere fatti 1 alla volta.

Per quanto riguarda l’output, questo è mostrato sotto forma di tabelle, come mostrato in seguito:

[RISULTATI BAYES CLASSIFIER (time per rep = “Tempo in minuti per ogni alpha”]

| Alpha | NDCG@5 | NDCG@10 | NDCG@20 |
| --- | --- | --- | --- |
| 0 | 0.656352 | 0.642129 | 0.625729 |
| 0.1 | … | … | … |
| 0.2 | … | … | … |
| … | … | … | … |

In modo analogo il risultato per il Nearest Neighbor Classifier. Prima delle tabelle viene inoltre mostrato un recap dei parametri indicati nella sezione “parametri” e delle modalità utilizzate (se è stato fatto clustering o meno per il n.n.c., o se è stata effettuata la riduzione delle features con il metodo del X2). Ogni riga della tabella mostra i risultati della raccomandazione per un singolo valore di *alpha.* Nelle colonne sono riportati i risultati degli indici di efficacia NDCG (risultati medi sul campione).

Dizionari “NewsArchive” e “Users”

Sampler.py genera due dizionari, Users\_sample.pkl e News.pkl, che vengono utilizzati da Recommender\_System.py. In particolare, questi file vengono caricati in “NewsArchive” e “Users”, due dizionari. Users serve a memorizzare i comportamenti degli user, NewsArchive le informazioni sugli articoli.

Le chiavi del dizionario “Users” sono gli ID degli user del campione a cui sono, a loro volta, associati dei dizionari. Coppie UserID – Dizionario. Ognuno di questi dizionari ha a sua volta 3 chiavi:

* “Liked”, a cui è associato un insieme di ID di articoli che lo user ha cliccato (ottenuti dai file behaviors.tsv).
* “Disliked”, a cui è associato un insieme di ID di articoli che lo user non ha cliccato (ottenuti dai file behaviors.tsv).
* “Test”, a cui è associato l’insieme di ID di articoli che vengono utilizzati come test set per lo user. Gli elementi di “Test” compaiono anche in “Liked” e “Disliked”: questo serve per mantenere il riferimento al “rating” che lo user ha dato all’articolo (sapere se lo ha letto o no per la fase di valutazione di efficacia). Quindi “Test” è un sottoinsieme di “Liked” | “Disliked”, e “Liked” | “Disliked” – “Test” corrisponde al training set dello user.

Quindi:

Users[UserID] -> accesso al dizionario associato ad uno User.

Users[UserID][“Liked”] -> accesso all’insieme di ID di articoli su cui ha cliccato.

-> {“N55528”, “N19639”, “N2073”, … }

Analogamente per Users[UserID][“Disliked”] e Users[UserID][“Test”].

Le chiavi del dizionario “NewsArchive” sono gli ID degli articoli, a cui sono associati dei dizionari. Coppie ID - dizionario. Questo dizionario contiene 6 chiavi: “Title” “Category” “Subcategory”, “Url”, ”TF\_IDF” e “Features”. Le informazioni riguardanti Abstract e Body dell’articolo non vengono salvate da sampler.py, visto che non sono necessarie nella raccomandazione.

Quindi:

NewsArchive[ArticleID] -> accesso al dizionario che contiene tutte le informazioni.

NewsArchive[ArticleID][“Title”] -> Stringa di testo, contenente il titolo pre-processato.

Analogo per “Category”, “Subcategory” e “Url”, che però non sono pre-processate.

NewsArchive[ArticleID][“TF\_IDF”] -> Dizionario termine-punteggio tf-idf.

NewsArchive[ArticleID][“Features”] -> Insieme di termini che costituiscono feature.

Effettuare gli esperimenti

Per effettuare gli esperimenti descritti nella relazione potrebbe essere necessario fare delle modifiche al codice di Recommender\_System.py. Il campione utilizzato (i file Users\_sample.pkl e News.pkl) sono presenti nella directory “CAMPIONE”, partendo dal link riportato a fine file.

*Confrontare i risultati per diversi pesi dei campi categoria, sottocategoria e titolo*

Come descritto nella relazione, i risultati dell’applicazione dei metodi vengono confrontati per i diversi pesi dati ai campi. Secondo la formula, il peso viene gestito tramite il parametro *alpha*.

Per farlo è sufficiente specificare, nella lista associata ad “alpha\_values” in main(), i valori di alpha da utilizzare. Può essere specificato anche un singolo valore.

*Fare clustering per il Nearest Neighbor Classifier*

È sufficiente rispondere “Y” quando viene chiesto di farlo.

*Utilizzare il metodo X2*

È sufficiente rispondere “Y” quando viene chiesto di farlo (una volta per il Nearest Neighbor Classifier, una per il Bayes). È possibile utilizzare il metodo anche per uno solo dei due.

Nel caso si voglia modificare il numero di features da tenere, si modifichi il parametro “retain” a inizio file.

Modificare il valore dei parametri ad inizio file permette di cambiare quei valori “arbitrari” che sono stati utilizzati nel progetto (es. il numero di vicini da tenere nel n.n.c.).

Link per dataset e campione: <https://drive.google.com/drive/folders/1JPQn4A63FlWmLWmOH9eLTvU9e4gZ8Yxq?usp=sharing>