UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea magistrale in
Informatica

IL TITOLO DELLA TESI

Relatore: Relatore 1
Correlatore: Correlatore 1

Tesi di Laurea di:
Lorenzo D'Alessandro
Matr. Nr. 939416

ANNO ACCADEMICO 2020-2021

Dedica

Ringraziamenti

Questa sezione, facoltativa, contiene i ringraziamenti.

Indice

Capitolo 1

Dataset

In questo capitolo sono descritti i dataset context-aware usati per valutare il modello moveCARS, e confrontarlo con altre soluzioni stato dell'arte. I risultati sono riportati nel Capitolo 2. Uno dei problemi nella valutazione dei CARS~~,~~ è la scarsità di dataset pubblici che contengono informazioni di contesto ad alta dimensionalità. Molti dataset pubblici infatti, hanno informazioni di contesto limitate unicamente al timestamp dei rating o alla posizione dell'utente, come ad esempio Yelp[1], Nowplaying-RS[2], Travel TripAdvisor[3]. Esistono invece dataset privati come CARS [24] e Hearo [29], che contengono i dati raccolti da esperimenti sul campo, in cui gli utenti interagivano con il proprio telefono con un RS che ~~che~~ raccomandava punti di interesse nelle vicinanze. I rating raccolti sono associati a numerose feature di contesto estratte da sensori ~~mobile~~ come accelerometro, microfono, giroscopio, etc. È difficile trovare dataset pubblici simili a Hearo e CARS, che caratterizzano il contesto fisico e sociale dell'utente con una grande quantità di feature. Dato che il modello moveCARS si inserisce in un ambiente mobile e pervasivo, ho selezionato due dataset context-aware in cui i feedback impliciti corrispondono alle applicazioni in esecuzione sui dispositivi Android degli utenti: (i) My Digital Footprint[4], (ii) Frappe[5].

1.1 Frappe

Frappe [41] è un dataset di feedback impliciti pubblicamente disponibile collezionato da un sistema di raccomandazione context-aware di applicazioni Android.

[1] `https://www.yelp.com/dataset`

[2] `https://zenodo.org/record/3247476#.YK9FxqgzY2x`

[3] `https://github.com/irecsys/CARSKit/blob/master/context-aware_data_sets/Travel_TripAdvisor_v1.zip`

[4] `https://github.com/contextkit/MyDigitalFootprint`

[5] `https://www.baltrunas.info/context-aware`

L'applicazione che monitora l'utilizzo degli smartphone, è stata installata da 957 utenti che hanno utilizzato un totale di 4082 applicazioni. Le informazioni raccolte descrivono la frequenza di utilizzo di un'applicazione da parte di un utente per un periodo di 2 mesi. Il numero totale di feedback presenti è 96203. Il task per un recommender system su questo dataset è prevedere se un'applicazione Android è rilevante per un utente in un determinato contesto.

1.1.1 Feature di contesto

Le feature di contesto descrivono la situazione dell'utente nel momento in cui ha utilizzato un'applicazione Android. Frappe può essere considerato un dataset con un contesto a bassa dimensionalità, e non contiene informazioni raccolte dai sensori dei dispositivi Android. Di seguito sono descritte tutte le feature di contesto presenti nel dataset.

Daytime è il momento della giornata in cui un'applicazione è stata utilizzata. La giornata è divisa in sette momenti diversi: mattina, mezzogiorno, pomeriggio, sera, tramonto, alba, notte.

Weekday è il giorno della settimana in cui un'applicazione è stata utilizzata. I possibili valori sono ovviamente i sette giorni della settimana.

Isweekend indica se un'applicazione è stata utilizzata nel fine settimana oppure in un giorno lavorativo. Può assumere due valori diversi: weekend e workday.

Homework indica se l'utente si trova al lavoro o a casa. Può assumere tre diversi valori: work, home, unknown.

Weather descrive la situazione meteo nel momento in cui un'applicazione è stata utilizzata. Può assumere nove valori differenti: soleggiato, nuvoloso, nebbioso, temporalesco, piovoso, nevoso, piovigginoso, nevischio, sconosciuto.

Country indica la nazione in cui si trovava l'utente nel momento in cui ha utilizzato un'applicazione. Ci sono 80 stati diversi, ma il 55% dei feedback sono stati generati da USA, Spagna e Regno Unito.

City è un valore numerico che rappresenta la città in cui si trovava l'utente nel momento in cui ha utilizzato un'applicazione. Ci sono 233 città diverse, ma per il 40% dei feedback la città è sconosciuta.

Delle feature di contesto appena descritte sono eliminate `homework, country`, e `city` perché poco utili a definire il contesto dell'utente. In particolare `city` è stata eliminata perché contiene troppi valori nulli, ed in corrispondenza di una città sconosciuta spesso anche il valore della feature `country` è sconosciuto. Anche `homework` è stato eliminato perché il numero di feedback che hanno la feature `homework` con valore sconosciuto è pari al 78%. `Country` non viene considerata perché la maggior parte dei feedback utente sono associati a poche nazioni, e ci sono un gran numero di nazioni a cui sono associati un numero non sufficiente di feedback. Le feature rimaste compongono il contesto: `daytime, weekday, isweekend, weather`. L'unica di queste feature che può assumere valore sconosciuto è `weather`. Le righe del dataset in cui `weather` è sconosciuto sono eliminate. Il risultato è un dataset con 78335 righe, 857 utenti e 3180 oggetti.

Per quanto riguarda l'encoding, essendo tutte variabili categoriche sono codificate con one-hot encoding. Il vettore del contesto risultato contiene 24 feature: 7 per `daytime`, 7 per `weekday`, 8 per `weather` e 2 per `isweekend`.

1.1.2 Feedback

Qualsiasi dataset per sistemi di raccomandazione ha tre feature fondamentali: user, item, rating. User e item sono valori numerici che identificano univocamente gli utenti e gli oggetti. Il rating in Frappe è il numero di volte in cui un oggetto (un'applicazione) è stato utilizzato da un utente in un determinato contesto. Ad esempio, se una riga del dataset è composta da `(user:1, item:20, rating:50, daytime:morning, weekday:monday)`, significa che l'utente 1, ha utilizzato l'applicazione 20, il lunedì mattina, 50 volte. Il numero di volte è ottenuto sommando tutti gli utilizzi durante il periodo di raccolta dei dati. Il valore minimo dei rating è 1, il valore massimo 21262, e la media 88.26. Questi rating vanno convertiti in feedback impliciti con valore 0 o 1 per essere compatibili con l'input di moveCARS, come spiegato nella **??**. Considerando il valore medio dei rating, e cercando di avere un dataset bilanciato, ho deciso di convertire tutti i rating con valore maggiore di 4 in feedback con valore 1, mentre i rating con valore 4 o minore in feedback negativi. Il risultato è un dataset con il 62% di feedback positivi (48604 ~~data sample~~).

1.1.3 Feature degli oggetti

Le feature degli oggetti sono caratteristiche che descrivono le applicazioni usate dagli utenti di Frappe. Il dataset contiene tre feature rilevanti per il task di classificazione: la categoria dell'applicazione, la lingua, e il costo.

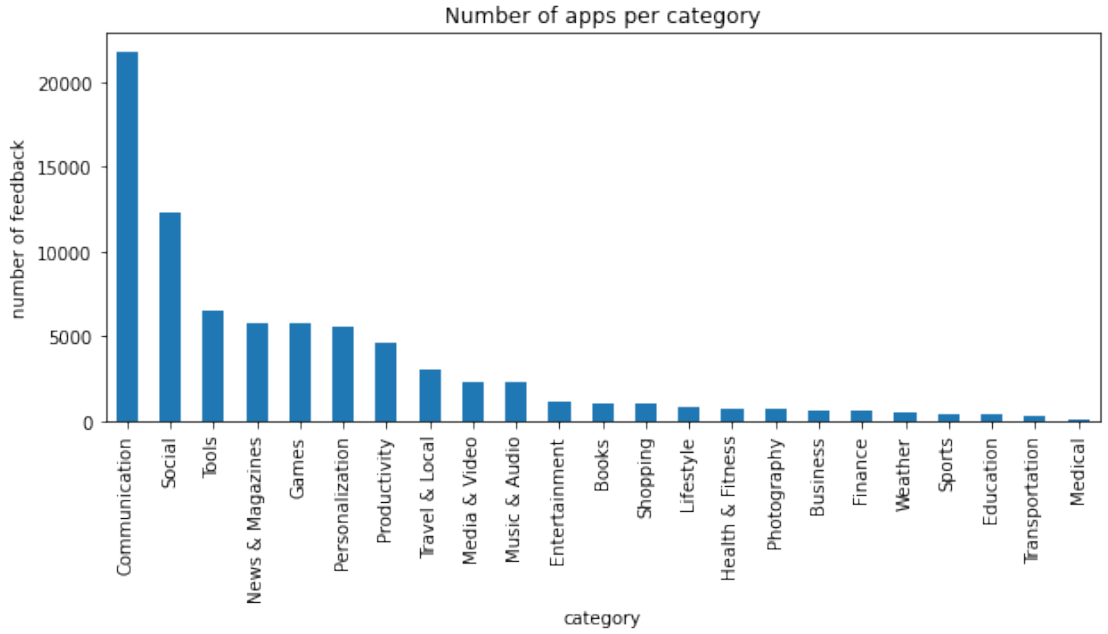

Figura 1: Numero di ~~data sample~~ per ogni categoria di applicazioni nel dataset Frappe

Category è la categoria delle applicazioni ottenuta dal Google Play Store. In Frappe le applicazioni sono divise in 23 categorie che descrivono la loro funzionalità principale (es. videogiochi, notizie, social). Come si può vedere in Figura 1, il numero di ~~data sample~~ per ogni categoria è molto sbilanciato: la categoria Communication che comprende applicazioni di messaggistica come WhatsApp e Telegram ha più di 20k ~~data sample~~. La seconda categoria più popolare è Social che comprende applicazioni come Facebook, Instagram e Twitter.

Language indica la lingua dell'applicazione ottenuta dal Google Play Store. Dato che il 96% dei campioni hanno applicazioni in lingua inglese, ho assegnato a tutte le altre lingue il valore other.

Cost indica se l'applicazione è gratuita o a pagamento.

Come per le feature di contesto, anche le feature degli oggetti sono variabili categoriche. Vengono codificate con one-hot encoding; il risultato è un vettore di 27 feature: 23 per `category`, 2 per `language`, e 2 per `cost`.

1.1.4 Feature degli utenti

Il dataset Frappe non contiene nessuna informazione associata agli utenti oltre all'ID dell'utente. Per questo motivo, ho generato le feature degli utenti a partire dalle feature degli oggetti e di contesto.

User category indica la categoria di applicazioni più utilizzata dall'utente. Come era prevedibile dalla distribuzione delle categorie mostrata in Figura 1, le categorie preferite dagli utenti sono Communication e Social.

User weekday indica il giorno della settimana in cui l'utente ha generato il maggior numero di feedback. I giorni più popolari sono venerdì e sabato.

User daytime indica il momento della giornata in cui l'utente ha generato il maggior numero di feedback. I momenti della giornata più popolari sono la sera e il pomeriggio.

User weather indica la condizione meteo in cui l'utente ha generato il maggior numero di feedback. Le condizioni meteo più popolari sono nuvoloso e soleggiato.

User weekend indica se l'utente ha generato più feedback in settimana o nel weekend. La maggior parte degli utenti (85%), ha generato più feedback in settimana.

User paid apps indica se l'utente ha mai utilizzato un'applicazione a pagamento. Il 59% degli utenti ha utilizzato almeno una volta un'applicazione a pagamento.

Come le feature di contesto e degli oggetti, anche le feature degli utenti sono tutte variabili categoriche. Vengono codificate con one-hot encoding in un vettore di 47 feature: 22 per `user category`, 7 per `user weekday`, 7 per `user daytime`, 7 per `user weather`, 2 per `user weekend`, 2 per `user paid apps`.

Ricapitolando, il dataset Frappe processato contiene 78335 data sample, 857 utenti e 3180 oggetti. Oltre alle colonne user, item e feedback, il dataset contiene 47 feature degli utenti, 22 feature degli oggetti, e 24 feature di contesto.

1.2 My Digital Footprint

My Digital Footprint (MDF) [42] è un nuovo dataset composto da dati di sensori di smartphone, informazioni di prossimità fisica, e interazioni sugli online social

network. Il dataset include due mesi di misurazioni e informazioni collezionate dai dispositivi personali di 31 volontari, nel loro ambiente naturale, senza limitare il loro comportamento usuale. I dati raccolti costituiscono un insieme completo di informazioni per descrivere il contesto utente in ambiente mobile.

Il dataset è organizzato in cartelle, una per ogni utente, che contengono diversi file csv, ognuno contenente misurazioni e informazioni di tipo diverso. Ogni campione in qualsiasi file csv contiene il timestamp in cui è stato acquisito. Alcuni sensori sono stati campionati molto frequentemente, mentre informazioni come il meteo sono state raccolte ogni ora. Di seguito è riportata la struttura delle cartelle:

```
MDF
├── user_0
│   ├── activities.csv
│   ├── audio.csv
│   ├── running_apps.csv
│   ├── wifi_scans.csv
│   └── ...
├── user_1
│   ├── activities.csv
│   ├── audio.csv
│   ├── running_apps.csv
│   ├── wifi_scans.csv
│   └── ...
└── ...
```

L'obiettivo è costruire un unico file csv, in cui ogni riga ha una struttura del tipo `(user_ID, item_ID, context)`. Siccome il task di un recommender system su questo dataset è prevedere se un'applicazione Android è rilevante per un utente in un determinato contesto, il punto di partenza per costruire il dataset sono le applicazioni in esecuzione nel file `running_apps.csv`. Qui sotto è riportato il codice Python per generare il dataset:

```python
data_path = 'Datasets/MDF/'
df = pd.DataFrame()
# foreach user folder
for user in range(31):
    user_dir = data_path + 'user_' + str(user)
    # read running_apps.csv and use it as a starting point
    df1 = pd.read_csv(user_dir + '/running_apps.csv', header=0)
    df1['time'] = pd.to_datetime(df1['time'], unit='ms')
    df1.sort_values('time', inplace=True)
    df1.reset_index(drop=True, inplace=True)
    df1.insert(1,'user',user) # insert user ID column

    rows = []
```

```
14      # foreach row in running apps dataframe find the closest row
        in all other csv file using timestamp
15      for dt in df1['time']:
16          row = []
17          # foreach csv file in user folder
18          for filename, columns in file_dict.items():
19              file_path = user_dir + '/' + filename
20              # single row with all the context features
21              row = row + get_closest_row(file_path, columns, dt).
        tolist()
22          rows.append(row)
23
24      df2 = pd.DataFrame(rows, columns=np.concatenate(list(file_dict
        .values())))
25      df3 = pd.concat([df1, df2], axis=1) # concat by column
26      df = pd.concat([df, df3], axis=0) # concat by row
27
28  df.reset_index(drop=True, inplace=True)
```

Per ogni cartella utente (riga 4) si legge il file `running_apps.csv` (riga 7). Per ogni elemento nel file con timestamp t (riga 15), e per ogni altro file csv nella cartella dell'utente corrente, si seleziona la riga con timestamp più vicino a t (riga 21). Il risultato è una tupla `(user_ID, item_ID, context)` (riga 25) che viene concatenata al dataset finale (riga 26).

1.2.1 Negative sampling

In MDF sono presenti solo i log indicanti che un'applicazione era in esecuzione sul dispositivo dell'utente, ad un certo timestamp t, in una situazione contestuale c. Per eseguire il training di una rete neurale sono però necessari degli esempi negativi, i quali indicano che un'applicazione non era in uso da parte di un utente al tempo t', in una specifica situazione contestuale c'. Ad ogni campione è associata una ~~label,~~ che riassume il contesto dell'utente ad alto livello con i seguenti valori: `home, school, workplace, external school` (quando gli autori del dataset incontravano i volontari), `free time`, e `holiday`. Questa etichetta non è usata come feature di contesto, ma per fare negative sampling del dataset. L'algoritmo 1 mostra il procedimento: per ogni ~~sample~~ d nel dataset D con struttura $(user, item, feedback, context, label)$, vengono identificate le label in cui $d.item$ non è mai stato utilizzato (riga 2). Per ogni label n viene scelto in modo casuale un sample $\in D$ con label $= n$ (riga 4). Di questo sample viene mantenuto solo il contesto $context_{neg}$ scartando user, item e feedback. Il sample negativo d_{neg} è ottenuto concatenando $d.user$ e $d.item$, con 0 (il feedback negativo) e $context_{neg}$ (riga 6). d_{neg} è aggiunto al dataset D_{neg} che contiene solo esempi negativi (riga

7). In ultimo il dataset D_{neg} è unito al dataset D, ed è eliminata la colonna corrispondente alle label (righe 10 e 11). Il risultato è una dataset con 31 utenti, 338 oggetti, e 73176 feedback, di cui il 66% con valore 1.

Algoritmo 1 Negative sampling di MDF

Input: D - dataset
Output: D_{neg} - dataset D with negative samples

1: **for all** $d \in D$ **do**
2: $labels_{neg} \leftarrow$ labels where $d.item$ was never used
3: **for all** $n \in labels_{neg}$ **do**
4: $context_{neg} \leftarrow$ context of a random data sample $\in D$ with $label = n$
5: $feedback \leftarrow 0$
6: $d_{neg} \leftarrow$ concatenate $d.user$, $d.item$, $feedback$ and $context_{neg}$
7: $D_{neg} \leftarrow D_{neg} \cup d_{neg}$
8: **end for**
9: **end for**
10: $D_{neg} \leftarrow D \cup D_{neg}$
11: $D_{neg} \leftarrow$ drop all labels from samples $\in D_{neg}$
12: Return D_{neg}

1.2.2 Feature di contesto

Il dataset MDF contiene numerosi dati estratti dai sensori e dal sistema operativo dei dispositivi personali degli utenti. Di seguito sono elencate solo le feature selezionate che compongono il contesto fisico e sociale dell'utente.

Attività utente L'attività utente, riconosciuta da Android Activity Recognition system[6], include sia movimenti a piedi che su mezzi di trasporto. Le attività possibili sono `in vehicle, on bicycle, on foot, running, still, tilting, walking, unknown`. Ogni feature rappresenta la probabilità da 0 a 100 che l'utente stia facendo quell'attività specifica.

Modalità audio `Ringer mode` indica se la modalità audio del telefono è impostata su silenzioso, vibrazione o suono.

[6] `https://developers.google.com/location-context/activity-recognition`

Volume `Alarm volume, music volume, notification volume` e `ring volume`, sono quattro feature con valore tra 0 e 1 che indicano il livello audio della sveglia, della musica, delle notifiche e della suoneria.

Musica `music active` è un valore booleano che indica se il dispositivo sta riproducendo della musica, `speaker on` specifica se è riprodotta dall'altoparlante del telefono; `headset connected` indica se sono collegate delle cuffie.

Batteria Alla batteria sono associate due feature: `level` indica la carica della batteria (molto bassa, bassa, media, alta, carica), `charging` è un valore booleano che indica se la batteria si sta ricaricando oppure no.

Schermo Associate al display ci sono due feature: `state` indica se il display è spento, acceso, o se si sta per spegnere. `Rotation` indica se l'utente sta usando il telefono in verticale o in orizzontale.

Meteo Il meteo è descritto da sei variabili diverse: temperatura, umidità, pressione atmosferica, velocità del vento, nuvole, e se ha piovuto nelle ultime tre ore.

Wifi La feature `connected` indica se il dispositivo dell'utente è connesso oppure no ad una rete Wi-Fi.

Data e ora Dai timestamp dei feedback in formato `YYYY-MM-DD HH:MM:SS`, sono estratte nuove feature: `daytime` (mattina, pomeriggio, sera, e notte) `weekday` e `isweekend`. Oltre a queste, con la libreria Python Holidays[7], è calcolato se è un giorno di vacanza o no, in base al calendario delle festività italiano.

Feature sociali Le feature sociali sono ottenute dall'ego network descritta nella ??. Ci sono quattro feature `social_c1, social_c2, social_c3, social_c4`, che corrispondono alle quattro cerchie sociali della ego network. I valori tra 0 e 1 di queste feature indicano la percentuale di alter in ogni cerchia sociale in prossimità dell'utente, nel momento in cui ha utilizzato un'applicazione. La feature `layer` indica il posizionamento nella ego network dell'utente da cui è stata ricevuta una raccomandazione. Questo presuppone un dataset diverso per ogni utente perché l'ego network è personale (es. l'utente u_1 potrebbe essere uno sconosciuto per u_2, ma un amico per u_3, e quindi u_1 è posizionato su un layer della ego network di u_2 diverso dal layer della ego network di u_3).

[7] `https://pypi.org/project/holidays/`

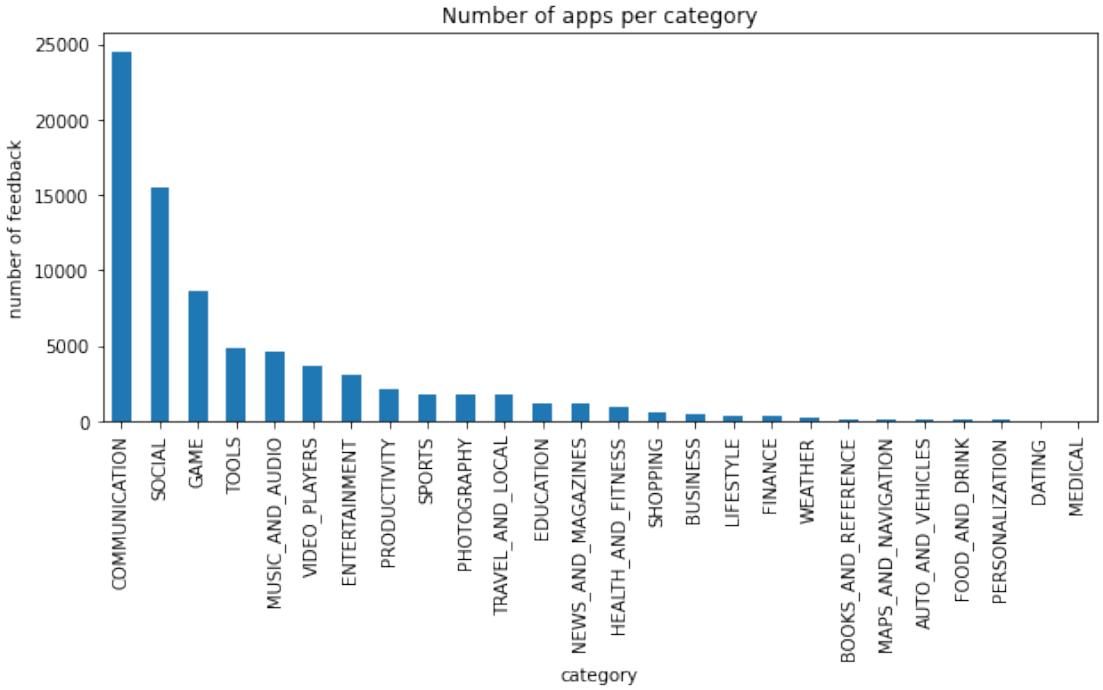

Figura 2: Numero di ~~data sample~~ per ogni categoria di applicazioni nel dataset My Digital Footprint

Le feature categoriche sono codificate con one-hot encoding, mentre le feature numeriche sono normalizzate. Il risultato è un vettore che contiene 63 feature che descrivono il contesto fisico dell'utente, e 8 feature che descrivono il contesto sociale dell'utente.

1.2.3 Feature degli oggetti

Category è la categoria delle applicazioni ottenuta dal Google Play Store. In MDF le applicazioni sono divise in 26 categorie che descrivono la loro funzionalità principale (es. videogiochi, notizie, social). Come si può vedere dalla Figura 2, anche in MDF il numero di data sample per ogni categoria è molto sbilanciato: la categoria Communication che comprende applicazioni di messaggistica come Whatsapp e Telegram ha più di 20k data sample. La seconda categoria più popolare è Social che comprende applicazioni come Facebook, Instagram e Twitter. La categoria è codificata con one-hot encoding.

1.2.4 Feature degli utenti

Come per Frappe, il dataset MDF non contiene nessuna informazione sugli utenti. Per questo motivo ho generato quattro feature utente dalle feature di contesto e degli oggetti.

User category indica la categoria di applicazioni più utilizzata dall'utente. Anche in questo caso le categorie più popolari sono Communication e Social.

User weekday indica il giorno della settimana in cui l'utente ha generato il maggior numero di feedback. I giorni più popolari sono giovedì e venerdì.

User daytime indica il momento della giornata in cui l'utente ha generato il maggior numero di feedback. Il momento della giornata più popolare è la mattina.

User weekend indica se l'utente ha generato più feedback in settimana o nel weekend. La maggior parte degli utenti (97%), ha generato più feedback in settimana.

Come le feature degli oggetti, anche le feature degli utenti sono tutte variabili categoriche. Vengono codificate con one-hot encoding in un vettore di 27 feature: 14 per `user category`, 7 per `user weekday`, 4 per `user daytime`, 2 per `user weekend`.

Ricapitolando, il dataset MDF processato contiene 73176 data sample, 31 utenti e 338 oggetti. Oltre alle colonne user, item e feedback, il dataset contiene 27 feature degli utenti, 26 feature degli oggetti, e 71 feature di contesto.

Bibliografia

[1] Prem Melville and Vikas Sindhwani. *Recommender Systems*, pages 829–838. Springer US, Boston, MA, 2010.

[2] Mattia G. Campana and Franca Delmastro. Recommender systems for online and mobile social networks: A survey. *Online Social Networks and Media*, 3-4:75–97, 2017.

[3] Arthur Mello. How do netflix and amazon know what i want? `https://towardsdatascience.com/how-do-netflix-and-amazon-know-what-i-want-852c480b67ac`, 2020.

[4] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, USA, 1st edition, 2010.

[5] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[6] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[7] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system. *ACM Computing Surveys*, 52(1):1–38, Feb 2019.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[9] Google Developers. Matrix factorization. `https://developers.google.com/machine-learning/recommendation/collaborative/matrix`, 2020.

[10] Simon Funk. Netflix update: Try this at home. `https://sifter.org/~simon/journal/20061211.html`, 2006.

[11] Yancheng Jia, Changhua Zhang, Qinghua Lu, and Peng Wang. Users' brands preference based on svd++ in recommender systems. In *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*, pages 1175–1178, 2014.

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008.

[13] Christopher C. Johnson. Logistic matrix factorization for implicit feedback data. 2014.

[14] Victor Köhler. Als implicit collaborative filtering. `https://medium.com/radon-dev/als-implicit-collaborative-filtering-5ed653ba39fe`, 2017.

[15] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[16] F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.

[17] Google Developers. Collaborative filtering advantages and disadvantages. `https://developers.google.com/machine-learning/recommendation/collaborative/summary`, 2020.

[18] Jesùs Bobadilla, Fernando Ortega, Antonio Hernando, and Jesùs Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238, 2012.

[19] Charu C. Aggarwal. *Recommender Systems - The Textbook*. Springer, 2016.

[20] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105. 01 2011.

[21] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[22] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, page 301–304, New York, NY, USA, 2011. Association for Computing Machinery.

[23] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, page 79–86, New York, NY, USA, 2010. Association for Computing Machinery.

[24] Moshe Unger, Alexander Tuzhilin, and Amit Livne. Context-aware recommendations based on deep learning frameworks. *ACM Trans. Manage. Inf. Syst.*, 11(2), May 2020.

[25] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. Contextual and sequential user embeddings for large-scale music recommendation. In *Fourteenth ACM Conference on Recommender Systems*, RecSys '20, page 53–62, New York, NY, USA, 2020. Association for Computing Machinery.

[26] Jiwon Hong, Won-Seok Hwang, Jin-Hyung Kim, and Sang-Wook Kim. Context-aware music recommendation in mobile smart devices. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, page 1463–1468, New York, NY, USA, 2014. Association for Computing Machinery.

[27] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. Lcars: A location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 221–229, New York, NY, USA, 2013. Association for Computing Machinery.

[28] Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, Oct. 2011.

[29] Moshe Unger, Ariel Bar, Bracha Shapira, and Lior Rokach. Towards latent context-aware recommendation systems. *Knowledge-Based Systems*, 104, 04 2016.

[30] Moshe Unger, Bracha Shapira, Lior Rokach, and Amit Livne. Inferring contextual preferences using deep encoder-decoder learners. *New Review of Hypermedia and Multimedia*, 24(3):262–290, 2018.

[31] Moshe Unger and Alexander Tuzhilin. Hierarchical latent context representation for context-aware recommendations. *IEEE Transactions on Knowledge and Data Engineering*, PP, 09 2020.

[32] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[33] Ian Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *Knowledge Discovery in Databases: PKDD 2005*, pages 59–70, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[34] Aurélien Géron. Hands-on machine learning with scikit-learn and tensorflow: Concepts. *Tools, and Techniques to build intelligent systems*, 2017.

[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[36] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010.

[37] Jason Brownlee. How to choose an activation function for deep learning. `https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/`, 2021.

[38] Daniel Godoy. Understanding binary cross-entropy / log loss: a visual explanation. `https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a`, 2018.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[40] Unsupervised modelling of the user's social context and visited locations at the edge of the internet.

[41] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. 05 2015.

[42] Mattia G. Campana and Franca Delmastro. Mydigitalfootprint: an extensive context dataset for pervasive computing applications at the edge. 2021.