2 M.Pellizzari, G. Santarelli, S. Nisato cold-start nella raccomandazione di news: nuovi articoli vengono proposti quo- tidianamente e rimangono online e rilevanti per un periodo di tempo limitato. L'utilizzo di approcci differenti, basati ad esempio sui comportamenti di utenti simili (collaborative filtering), risultano invece meno efficaci: un articolo non ri- mane rilevante per un tempo sufficiente a raccogliere abbastanza interazioni con gli user. Nell'applicare questi sistemi di raccomandazione, content-based, nell'ambito de- gli articoli online, sono stati affrontati alcuni problemi rilevanti. Innanzitutto infatti 'e stato necessario ottenere delle informazioni sugli articoli e sui compor- tamenti degli user, per le quali 'e stato indispensabile individuare una buona rappresentazione strutturata. Una prima sfida 'e stata quindi la costruzione dei profili utente e dei profili item, da utilizzare poi nelle applicazioni da confrontare. In particolare le preferenze degli utenti, necessarie per la rappresentazione tra- mite user profile, non erano disponibili in modo esplicito e diretto, ma sono state derivate dai loro comportamenti impliciti. Non solo, anche le news, oggetto della raccomandazione, sono in forma di dati non strutturati, che non seguono dunque una schema preciso. Perci'o in questo caso si 'e scelto di rappresentare gli articoli tramite un insieme di parole chiave, dette features, che vanno a costituire gli item profiles. Esistono vari approcci per la feature selection, la selezione delle parole chiave. In questo lavoro sono stati esplorati due metodi: il primo, non supervisionato, si basa sul calcolo del punteggio TF-IDF, che permette di individuare le parole pi'u significative, e quindi pi'u rilevanti, da utilizzare; il secondo 'e invece di tipo supervisionato, applicato ai risultati del primo, e si basa sulla statistica X2. Per la parte di raccomandazione vera e propria, in questo lavoro sono stati confrontati, tramite l'indice NDGC@20, due diversi metodi per effettuare la raccomandazione (testati su un campione di utenti): il Nearest Neighbor Classifier e il Bayes Classifier. Oltre alla misura di efficacia i due metodi sono stati confrontati in termini di tempo, in minuti reali, impiegato: in una vera applicazione, avere un'idea della velocit'a percepita dall'utente per la raccomandazione 'e importante. Una volta ottenuti i primi risultati 'e stata inoltre studiata una tecnica per la riduzione del tempo di esecuzione del Nearest Neighbor Classifier, basata sul clustering. 2 Base di partenza Raccomandare significa suggerire oggetti, nel nostro caso news, a determinati utenti. L'approccio che si 'e deciso di utilizzare in questo lavoro 'e di tipo content-based, quindi viene fatto un ampio utilizzo del contenuto degli articoli. L'idea di base dei metodi proposti 'e la seguente: determinare il contenuto degli articoli, che costituiscono potenziali candidati per la raccomandazione, e confrontarlo con i contenuti per i quali l'utente ha espresso una preferenza in passato. Il punto di partenza 'e costituito perci'o dalla costruzione degli item profile: l'in- sieme di attributi che caratterizzano gli articoli. La rappresentazione dell'item 4 M.Pellizzari, G. Santarelli, S. Nisato degli articoli proposti allo user in questo accesso, con una specifica del tipo di interazione che lo user ha avuto (il codice dell'articolo termina con "-1" se l'utente lo ha cliccato, "-0" se non 'e stato cliccato). Nel cercare di utilizzare questi dati per costruire un sistema di raccomandazione 'e stata fatta un'importante assunzione (necessaria per determinare le preferenze dello user): gli articoli letti prima del periodo di osservazione e quelli cliccati nelle impression a lui riferite sono graditi, mentre gli articoli ignorati sono as- sunti non graditi. Ogni record del file news.tsv fa riferimento ad un articolo, del quale vengono specificati un codice identificativo, il titolo, un abstract, la categoria, la sotto- categoria e l'URL, che porta alla pagina web dell'articolo. 'E questo URL che ci permette di ottenere il contenuto dell'articolo, usato insieme a titolo, abstract, categoria e sottocategoria per determinarne le features. 3.2 Costruzione del campione Le rilevazioni dei comportamenti degli user hanno coinvolto un periodo di osservazione di 6 settimane: per il progetto abbiamo deciso di utilizzare i dati delle prime 5 settimane (i dati riferiti ai set che MIND chiama "Training set" e "Validation set"). Per ogni user, utilizzando le impression a lui riferite, abbiamo memorizzato le news cliccate (assunte gradite) e quelle non cliccate (assunte non gradite) nel periodo di 5 settimane, in aggiunta ai dati delle news cliccate in precedenza (assunte gradite). Il dataset a disposizione conteneva un gran numero di utenti e di news. Per il lavoro effettuato abbiamo quindi deciso di considerare solo un sottogruppo degli user di numerosit'a 200, selezionati tra gli

Neighbor Classifier. Il metodo che porter‘a ad avere un NDGC@20 medio pi‘u elevato sul campione risulter‘a essere stato il pi‘u efficace. 5 Esperimenti Iniziando con la feature selection base, effettuata tenendo, per ogni articolo, le 100 features con punteggio TF-IDF pi‘u alto, i risultati sono i seguenti: Tabella 2. RISULTATI BAYES CLASSIFIER AL VARIARE DI = 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 NDCG@5 0.655 0.655 0.660 0.661 0.652 0.655 0.657 0.652 0.646 0.653 0.658 NDCG@10 0.651 0.651 0.653 0.658 0.653 0.651 0.648 0.644 0.641 0.645 0.65 NDCG@20 0.629 0.631 0.632 0.634 0.631 0.632 0.632 0.628 0.626 0.629 0.612 Tabella 3. RISULTATI NEAREST NEIGHBOR CLASSIFIER AL VARIARE DI = 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 NDCG@5 0.750 0.746 0.739 0.728 0.719 0.717 0.713 0.703 0.68 0.65 0.57 NDCG@10 0.723 0.722 0.717 0.714 0.708 0.699 0.693 0.679 0.661 0.638 0.566 NDCG@20 0.684 0.680 0.676 0.675 0.67 0.667 0.665 0.657 0.64 0.618 0.560 I risultati migliori rispetto all’NDCG@20, per i due metodi, sono stati ottenuti utilizzato, rispettivamente, pari a 0.3 e 0. Confrontando i risultati considerando, per i due metodi, quello nella migliore configurazione delle features, il Nearest Neighbor Classifier risulta essere stato il pi‘u efficace (0.684 contro 0.634). Inoltre, il N.N.C. risulta pi‘u efficace anche rispetto gli indici NDCG@5 e NDCG@10. Per quanto riguarda invece il tempo impiegato (misurato in minuti reali) il Bayes Classifier ‘e notevolmente pi‘u veloce: per una singola configurazione di features, un singolo , il B.C. ha impiegato 3.86 minuti, contro i 9.408 minuti del N.N.C.. Questi sono comunque tempi medi, una singola iterazione di alpha ‘e pi‘u lenta: per = 1 la raccomandazione ‘e molto pi‘u rapida, e questo abbassa i tempi medi. Sembra quindi esserci un trade-off tra tempo impiegato ed efficacia della raccomandazione. Ricorrendo al clustering per il Nearest Neighbor Classifier il tempo impiegato aumenta: l’overhead dovuto alla costruzione dei cluster ‘e sufficiente a negare completamente il risparmio dovuto alla riduzione del numero di misure di di- stanza da calcolare. Questo ‘e probabilmente dovuto alle dimensioni ridotte dei 12 M.Pellizzari, G. Santarelli, S. Nisato test set. Il risultato migliore ‘e per = 0: NDCG@5 = 0.635, NDCG@10 = 0.624, NDCG@20 = 0.61, ottenuto in 9.539 minuti. Anche l’efficacia ‘e quindi molto in- feriore a quella osservata senza fare clustering. Come cambiano i risultati riducendo il numero di features con il metodo basato sulla statistica X2? Dimezzando le parole chiave per ogni articolo, i risultati sono i seguenti (riportati i risultati migliori rispetto ad e all’NDCG@20 medio): Tabella 4. Risultati dopo supervised feature selection NDCG@5 NDCG@10 NDCG@20 B.C. = 0.8 0.657 0.637 0.603 N.N.C. = 0 0.72 0.700 0.673 Per quanto riguarda il confronto: il N.N.C. riporta risultati migliori per ognuno dei 3 indici medi, come nel caso precedente. Nonostante per‘o l’efficacia della raccomandazione si sia ridotta leggermente, c’‘e stato un notevole guadagno in termini di tempo: 5.171 minuti impiegati per il N.N.C. e 1.936 per il B.C. (per ogni valore di). Il tempo impiegato si ‘e quasi dimezzato, anche se in esso ‘e compreso il tempo necessario per l’ulteriore feature selection. Questo pu‘o essere spiegato con il fatto che il calcolo della distanza coseno (usata nel N.N.C) e la stima della probabilit‘a (nel B.C.) dipendano entrambi dal numero di features utilizzate. Utilizzando la supervised feature selection nel

caso di clustering l'efficacia rimane pressoch e identica (per = 0, NDCG@5 = 0.627 ,NDCG@10 = 0.62, NDCG@20 = 0.61), ma con una notevole riduzione di tempo, 5.114 minuti per ogni valore di . In conclusione il Nearest Neighbor Classifier si 'e rivelato il pi'u efficace, anche se il tempo impiegato 'e molto pi'u elevato. La riduzione di features con il meto- do del X2 permette comunque di ridurre notevolmente il tempo di esecuzione, mantenendo buoni livelli di efficacia, mentre il clustering non ha dato i risultati sperati. Dare diverso peso ai campi, tramite , non si 'e rivelato efficace: per il N.N.C. i risultati migliori si ottengono per = 0 (stesso peso per ogni campo), mentre i risultati del B.C. erano sempre tutti molti simili per ogni valore (ad eccezione di = 1). I risultati comunque devono essere letti ricordando la debolezza dell'assunzione iniziale, necessaria per determinare le preferenze degli utenti, che sicuramente ha influito negativamente sui risultati di efficacia dei metodi impiegati. Sistemi di raccomandazione content-based per articoli 13 Riferimenti bibliografici 1. Charu C. Aggarwal: Recommender Systems. Springer, NY, USA (2016) 2. Leskovec, J., Rajaraman, A., D. Ullman, J.: Mining of Massive Datasets. Palo Alto, CA (July, 2019) 3. Microsoft: Introduction to MIND and MIND-small datasets. Github. https://github.com/msnews/msnews.github.io/blob/master/assets/doc/ introduction.md.

Sistemi di raccomandazione content-based per articoli

Marco Pellizzari[1], Greta Santarelli[2], and Sara Nisato[3]

[1] Corso di laurea magistrale in Scienze Statistiche, matricola 2054751
marco.pellizzari.5@studenti.unipd.it
[2] Corso di laurea magistrale in Scienze Statistiche, matricola 2039102
greta.santarelli.1@studenti.unipd.it
[3] Corso di laurea magistrale in Scienze Statistiche, matricola 1198565
sara.nisato@studenti.unipd.it

Sommario Il tema trattato in questo elaborato sono i sistemi di raccomandazione, i quali svolgono l'importante funzione di filtrare e personalizzare in modo automatico le informazioni così da risolvere il problema legato all'immensa quantità informativa a cui l'utente è esposto quotidianamente. L'attenzione viene posta sulla raccomandazione di news, per la quale è stato scelto un approccio *content-based*. L'obiettivo del lavoro svolto è quello di confrontare la diversa efficacia nell'utilizzo di diverse tecniche nell'ambito della raccomandazione di articoli.

Keywords: Content-based recommendation· news · TF.IDF · X^2 Statistic · Nearest Neighbor Classifier · K-means clustering · Bayes Classifier · NDCG

1 Introduzione

Il problema del confronto della diversa efficacia dei sistemi di raccomandazione, trattato in questo lavoro, è significativo nell'ambito della fruizione di articoli online.
La soluzione del problema trattato permetterebbe innanzitutto una migliore esperienza per lo *user*, il fruitore degli articoli, poiché verrebbe resa più efficace la ricerca di articoli di suo interesse. Questo è un aspetto importante anche per l'offerente del servizio, che riuscirebbe a migliorare la soddisfazione dei suoi utenti.
La necessità di impiegare i sistemi di raccomandazione è legata alla grande quantità di articoli disponibili: un singolo utente deve infatti concentrarsi su un sottoinsieme di articoli di suo interesse. E da qui l'impiego di questi sistemi, che permettono di suggerire un insieme selezionato degli articoli presenti online sulla base delle scelte passate (le preferenze) dello *user*.
L'approccio scelto per questo lavoro, di tipo *content-based*, risulta particolarmente efficace nell'ambito della raccomandazione di articoli. Lavorando con il contenuto del documento è possibile infatti limitare il rilevante problema del

cold-start nella raccomandazione di news: nuovi articoli vengono proposti quotidianamente e rimangono online e rilevanti per un periodo di tempo limitato. L'utilizzo di approcci differenti, basati ad esempio sui comportamenti di utenti simili (*collaborative filtering*), risultano invece meno efficaci: un articolo non rimane rilevante per un tempo sufficiente a raccogliere abbastanza interazioni con gli *user*.
Nell'applicare questi sistemi di raccomandazione, *content-based*, nell'ambito degli articoli online, sono stati affrontati alcuni problemi rilevanti. Innanzitutto infatti è stato necessario ottenere delle informazioni sugli articoli e sui comportamenti degli *user*, per le quali è stato indispensabile individuare una buona rappresentazione strutturata. Una prima sfida è stata quindi la costruzione dei profili utente e dei profili item, da utilizzare poi nelle applicazioni da confrontare. In particolare le preferenze degli utenti, necessarie per la rappresentazione tramite *user profile*, non erano disponibili in modo esplicito e diretto, ma sono state derivate dai loro comportamenti impliciti. Non solo, anche le news, oggetto della raccomandazione, sono in forma di dati non strutturati, che non seguono dunque una schema preciso. Perciò in questo caso si è scelto di rappresentare gli articoli tramite un insieme di parole chiave, dette *features*, che vanno a costituire gli *item profiles*.
Esistono vari approcci per la *feature selection*, la selezione delle parole chiave. In questo lavoro sono stati esplorati due metodi: il primo, non supervisionato, si basa sul calcolo del punteggio TF-IDF, che permette di individuare le parole più significative, e quindi più rilevanti, da utilizzare; il secondo è invece di tipo supervisionato, applicato ai risultati del primo, e si basa sulla *statistica* X^2.
Per la parte di raccomandazione vera e propria, in questo lavoro sono stati confrontati, tramite l'indice *NDGC@20*, due diversi metodi per effettuare la raccomandazione (testati su un campione di utenti): il *Nearest Neighbor Classifier* e il *Bayes Classifier*. Oltre alla misura di efficacia i due metodi sono stati confrontati in termini di tempo, in minuti reali, impiegato: in una vera applicazione, avere un'idea della velocità *percepita* dall'utente per la raccomandazione è importante. Una volta ottenuti i primi risultati è stata inoltre studiata una tecnica per la riduzione del tempo di esecuzione del *Nearest Neighbor Classifier*, basata sul *clustering*.

2 Base di partenza

Raccomandare significa suggerire oggetti, nel nostro caso news, a determinati utenti.
L'approccio che si è deciso di utilizzare in questo lavoro è di tipo *content-based*, quindi viene fatto un ampio utilizzo del contenuto degli articoli. L'idea di base dei metodi proposti è la seguente: determinare il contenuto degli articoli, che costituiscono potenziali candidati per la raccomandazione, e confrontarlo con i contenuti per i quali l'utente ha espresso una preferenza in passato.
Il punto di partenza è costituito perciò dalla costruzione degli *item profile*: l'insieme di attributi che caratterizzano gli articoli. La rappresentazione dell'*item*

profile può essere fatta in diversi modi: in questo lavoro è stata utilizzata una rappresentazione tramite parole chiave. Ogni articolo viene quindi rappresentato tramite le parole che, tra quelle che lo costituiscono, lo rappresentano maggiormente. L'identificazione di queste parole viene fatta tramite il calcolo dello score TF-IDF, un punteggio che, calcolato per ogni parola del testo, permette di ordinarle in ordine di significatività. Per ogni documento vengono così tenute solo le parole più significative (con score più alto) che prendono il nome di *features*.
Formalmente, l'*item profile* è un vettore di dimensione d (numero di parole del vocabolario complessivo), dove alla posizione i-esima compare il punteggio TF-IDF dell'i-esima *feature* nel documento. Il valore è 0 se la parola non è tra le *features*. Quindi, dato un item c, il suo profilo sarà:

$$ItemProfile(c) = (t_1, t_2, ..., t_d) \tag{1}$$

Nel caso dello *user profile* la rappresentazione è più semplice. Ogni utente viene infatti rappresentato dall'insieme di articoli che ha gradito e non gradito in passato (quindi dal suo training set, D_L).
L'idea che sta alla base dei metodi di raccomandazione proposti in questo lavoro è la seguente: confrontare degli articoli candidati, per ogni utente, con quelli che lui ha letto in passato e, in base alle sue preferenze, cercare di ordinare gli articoli candidati secondo una qualche stima di gradimento.
I metodi proposti per fare questo sono due: il *Nearest Neighbor Classifier* e il *Bayes Classifier*.
Una volta proposto un ordinamento per gli articoli con i due metodi, con riferimento ad uno user, viene calcolato l'indice NDGC@20, che misura la capacità del metodo di ordinare, alle prime 20 posizioni, articoli rilevanti. La media dell'indice NDGC@20, calcolata sulla base degli *user*, viene utilizzata per confrontare l'efficacia dei due metodi.

3 Problema affrontato

3.1 Descrizione del dataset

Come dataset di riferimento, su cui condurre l'analisi, è stato scelto il dataset MIND, un registro dei comportamenti di oltre 1 milione di utenti, sul sito Web di Microsoft News, in un periodo di riferimento di 6 settimane.
MIND contiene circa 160k articoli di notizie in inglese e più di 15 milioni di *impression* generate dagli utenti, dove il termine *impression* fa riferimento all'accesso di un utente alla homepage del sito di news, del quale vengono registrate una serie di informazioni (utilizzate per costruire gli *user profile*).
Questi dati sono contenuti in due file: *news.tsv*, contenente le informazioni dettagliate sugli articoli, e *behaviors.tsv*, contenente le informazioni sugli accessi.
In ogni record del file *behaviors.tsv*, riferito ad un singolo accesso, vengono specificati ID dell'*impression*, ID dell'utente a cui fa riferimento, la data dell'accesso, un elenco degli articoli che l'utente ha cliccato prima dell'*impression* e una lista

degli articoli proposti allo *user* in questo accesso, con una specifica del tipo di interazione che lo *user* ha avuto (il codice dell'articolo termina con "-1" se l'utente lo ha cliccato, "-0" se non è stato cliccato).
Nel cercare di utilizzare questi dati per costruire un sistema di raccomandazione è stata fatta un'importante assunzione (necessaria per determinare le preferenze dello *user*): gli articoli letti prima del periodo di osservazione e quelli cliccati nelle *impression* a lui riferite sono graditi, mentre gli articoli ignorati sono assunti non graditi.
Ogni record del file *news.tsv* fa riferimento ad un articolo, del quale vengono specificati un codice identificativo, il titolo, un abstract, la categoria, la sottocategoria e l'URL, che porta alla pagina web dell'articolo. È questo URL che ci permette di ottenere il contenuto dell'articolo, usato insieme a titolo, abstract, categoria e sottocategoria per determinarne le *features*.

3.2 Costruzione del campione

Le rilevazioni dei comportamenti degli *user* hanno coinvolto un periodo di osservazione di 6 settimane: per il progetto abbiamo deciso di utilizzare i dati delle prime 5 settimane (i dati riferiti ai set che MIND chiama "Training set" e "Validation set"). Per ogni *user*, utilizzando le *impression* a lui riferite, abbiamo memorizzato le news cliccate (assunte gradite) e quelle non cliccate (assunte non gradite) nel periodo di 5 settimane, in aggiunta ai dati delle news cliccate in precedenza (assunte gradite).
Il dataset a disposizione conteneva un gran numero di utenti e di news. Per il lavoro effettuato abbiamo quindi deciso di considerare solo un sottogruppo degli *user* di numerosità 200, selezionati tra gli *user* con almeno 100 articoli cliccati e 100 articoli ignorati. La riduzione del numero di *user* con cui operare è una scelta di carattere pratico: ridurre il numero di utenti porta ad una riduzione del numero di articoli da considerare, e quindi una riduzione del tempo necessario per le fasi di *web scraping*, *pre-processing* e, successivamente, per la raccomandazione. Una nota importante inoltre è che, nel nostro caso, i 200 utenti sono stati scelti a partire dai dataset MIND small, relativi ad un sample di *user* e articoli messo a disposizione.
Per quanto riguarda invece la ragione dell'imposizione di condizioni sull'entrata di uno *user* nel campione, il motivo è semplice: è necessario avere sufficiente materiale per allenare lo *user model* (determinare le preferenze dello *user*) perché la raccomandazione sia efficace.
Individuati gli *user* con cui costruire il campione, abbiamo costruito training set e test set. Per ogni *user* è stato effettuato uno split 60%-40%: abbiamo selezionato il 40% delle news gradite e delle news non gradite da nascondere ed utilizzare come test set, mentre il restante 60% è stato utilizzato come training set (e che va quindi a costituire lo *user profile*). È importante però riconoscere che una divisione di questo tipo non riflette quello che può essere un caso reale: nel training set possono essere presenti articoli con cui lo *user* ha interagito in un momento successivo rispetto ad alcuni articoli nel test set. In una reale applicazione di un sistema di raccomandazione lo split degli articoli potrebbe essere

fatto utilizzando un riferimento temporale: i primi 60% degli articoli con cui lo *user* ha interagito vengono utilizzati nel training set, il restante 40% nel test set. Tuttavia, non c'è ragione di pensare che la divisione da noi effettuata porti a risultati significativamente diversi nel confronto tra i metodi di classificazione scelti rispetto a quelli osservabili in un caso reale.

3.3 Web scraping e pre-processing

Tramite l'URL dell'articolo, contenuto in *news.tsv*, è stato possibile accedere al contenuto, in formato HTML, e in un secondo momento effettuare del *preprocessing*. Per accedere al contenuto testuale dell'articolo sono state utilizzate due librerie: *Requests* e *BeautifulSoup*.
Tramite *Requests* viene effettuata la richiesta all'URL, mentre *BeautifulSoup* è stata utilizzata per fare *parsing* della pagina e convertirla in una struttura ad albero di facile accesso. La maggior parte del testo degli articoli è risultato essere contenuto tra i tag <p>, <h1>, <h2> e <h3>. Il contenuto tra i tag <p> è il testo riferito ai paragrafi, mentre il testo nei tag <h> è legato a dei sotto-titoli. Il testo tra i tag <h1> non è stato tuttavia utilizzato in quanto contiene il titolo dell'articolo, già presente nel file *news.tsv*.
Dopo aver estratto il corpo del documento viene effettuato del *pre-processing*, tramite la libreria *nltk*, con l'obiettivo di ottenere un testo privo di parole irrilevanti e di minore importanza. Innanzitutto, il testo viene convertito in *lower case* e suddiviso in *token*, dai quali viene rimossa la punteggiatura e le *stopwords*, parole come articoli e congiunzioni. Vengono inoltre eliminati i *token* ad una lettera, poco indicativi, e quelli in cui compare un apostrofo (come "'s", "'ve", "n't", risultanti dalla tokenizzazione). Successivamente viene effettuato dello *stemming*, per riportare le parole alle loro radici. Queste operazioni di *preprocessing* vengono ripetute anche per il testo del titolo e dell'abstract. Questo permette di ottenere un testo più pulito, che permette di individuare *keywords* più rilevanti al fine del confronto tra documenti. Le operazioni di *web scraping* e *pre-processing* vengono effettuate utilizzando più processi in parallelo, attraverso le funzionalità del pacchetto *multiprocessing*.

4 Metodi proposti

In seguito alle operazioni preliminari descritte in precedenza è iniziata la fase di *feature selection*. Ogni item è stato, in questa fase, rappresentato tramite un insieme di parole chiave, identificate tramite il calcolo dello score TF-IDF per ogni termine. Le parole con lo score più alto, e quindi più significative, sono state scelte per rappresentare il documento.

4.1 Feature selection

Per quanto riguardo il calcolo dello score TF-IDF ci sono degli aspetti rilevanti da tenere in considerazione legati ai metodi utilizzati per il calcolo della *term*

frequency (TF) e della *inverse document frequency* (IDF). Nel nostro caso infatti la componente TF viene calcolata come rapporto tra la frequenza della parola nel documento ed il numero totale di parole nel documento.
Inizialmente, nel calcolare la componente TF, non sono state fatte distinzioni tra i campi contenenti testo: tutte le parole che compaiono nell'articolo, a prescindere dal fatto che facciano riferimento a corpo, titolo, categoria, sottocategoria o abstract, hanno lo stesso peso. Per farlo, è sufficiente immaginare i campi testuali che descrivono l'articolo come appartenenti ad un singolo testo, e le frequenze come calcolate in riferimento ad esso. Queste frequenze sono state poi divise per il numero di termini di nel testo.
Una seconda versione considerata prevede invece che i termini abbiano un peso differente: quelli di corpo e abstract un certo peso, quelli di titolo, categoria e sottocategoria uno maggiore. Per farlo è sufficiente modificare i valori di *term frequency* come indicato nella formula che segue, utilizzando un parametro α, il cui valore varia tra 0 ed 1.

$$TF = TF(titolo, categoria, sottocategoria)\alpha + TF(1 - \alpha) \tag{2}$$

La componente TF, per i termini, è sempre la stessa: l'utilizzo di α porta ad una sua riduzione solo per quei termini che non compaiono in titolo, categoria o sottocategoria.
Per quanto riguarda il valore di α, ci sono un paio di considerazioni da fare: un valore pari a zero ci colloca nella situazione precedente, ossia tutti i termini hanno lo stesso peso. Più vicino il valore è ad 1, più lo score dei termini di abstract e corpo viene ridotto. Nel caso il valore scelto sia 1, allora i termini di abstract e corpo non vengono presi in considerazione per la determinazione delle *features* (un valore pari a zero indica un'assenza di significatività). Il valore da utilizzare con questo metodo è comunque da determinare: una possibilità presa in considerazione è quella di individuare i valori di α che massimizzino la misura di efficacia della raccomandazione per i metodi utilizzati. In particolare, si tratta di individuare i 2 valori che massimizzano i risultati per il *Nearest Neighbor Classifier* e per il *Bayes Classifier*.
Anche per il calcolo della *inverse document frequency* è necessario mettere in evidenza un aspetto importante. In una situazione reale, il calcolo della componente IDF dovrebbe essere fatto utilizzando esclusivamente i documenti del training set. Nel contesto reale infatti i documenti da raccomandare verrebbero presi in considerazione uno ad uno, e per la determinazione delle loro *features* dovrebbe essere fatto affidamento su valori preesistenti (calcolati utilizzando un training set). Nel nostro caso, però, non c'è una buona distinzione tra articoli di training e di test (da valutare per la raccomandazione): alcuni documenti compaiono in entrambi. Per semplicità, quindi, la componente IDF, per ogni termine, viene calcolata prendendo in considerazione tutti gli articoli, sia quelli dei training set che dei test set.
Il punteggio TF-IDF viene poi calcolato come prodotto tra la componente TF e IDF, data dal logaritmo del rapporto tra il numero di documenti totali (nel nostro caso quelli dei training set e dei test set) e il numero di documenti in cui

il termine in questione compare.
Una volta calcolati i punteggi dei termini per ogni articolo vengono selezionate le *features*. Per ogni articolo vengono selezionati i 100 termini con punteggio più alto (e quindi i più significativi) da utilizzare come parole chiave per rappresentare il documento.

Una volta ottenuti i risultati della raccomandazione con questo primo metodo, non supervisionato, è stato applicato un ulteriore metodo di *feature selection*, questa volta supervisionato.
In questo secondo caso abbiamo cercato di capire come sarebbero cambiati i risultati adottando una tecnica basata sulla *statistica* X^2. Dato un utente, questa statistica può essere calcolata, per ogni *feature* degli articoli di training e di test, trattando la sua co-occorrenza (nel training set) con le classi di *rating*, articolo gradito o non gradito, come una tabella di contingenza.

Tabella 1. Contingency table

	Feature compare	Feature non compare
Articoli graditi	O_1	O_2
Articoli non graditi	O_3	O_4

È quindi possibile confrontare le frequenze osservate (i valori nelle celle) con quelle attese nel caso di indipendenza tra feature e classi, calcolando la *statistica* X^2. Un punteggio alto per la statistica indica che la feature è correlata alla classe, quindi tanto più indicativa del gradimento o del non gradimento. Le *features* con valore più alto vengono tenute per rappresentare il documento.
Questo secondo metodo è stato applicato *successivamente* alla feature selection iniziale: dati gli articoli di training e di test per uno user, delle *features* utilizzate ne vengono tenute la metà. Questo è quindi un metodo di selezione in qualche modo "personalizzato" per il singolo utente.

4.2 Nearest Neighbor Classifier

Con riferimento alla notazione utilizzata: *rating 1* indica un articolo gradito dallo user, *rating 0* un articolo non gradito. Dato uno user, D_L indica il suo training set, D_U il suo test set.

Il primo metodo per effettuare la raccomandazione è il *Nearest Neighbor Classifier*. Il primo passo consiste nella definizione di una misura di similarità, da utilizzare nella classificazione. Avendo una rappresentazione vettoriale di ogni documento (ogni posizione è una parola chiave, e il valore associato è lo score), la misura di similarità scelta è la *distanza coseno*, la cui formula è:

$$Cosine(\bar{X}, \bar{Y}) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} {x_i}^2}\sqrt{\sum_{i=1}^{d} {y_i}^2}} \tag{3}$$

Nella formula, $\bar{X}$ e $\bar{Y}$ sono i vettori che rappresentano i due documenti, gli *item profile*, d è la dimensione del vocabolario complessivo, mentre x_i e y_i sono i punteggi associati alle *features* nei rispettivi articoli (0 se la *feature* non è utilizzata). Questa misura, calcolata tra due documenti, restituisce un valore tanto più alto quanto più simili sono i vettori (fino ad un massimo di 1).
Una volta scelta la misura di distanza si passa alla raccomandazione: per ogni *user*, vengono studiati i documenti appartenenti al corrispondente test set (D_U). Iniziando da uno user qualsiasi, per ognuno degli articoli nel suo test set vengono individuati, tramite la misura di distanza scelta, i k documenti più simili nel suo training set (D_L), dove k è molto minore di $|D_L|$. Il valore di k utilizzato nel nostro progetto è pari a 20. Utilizzando i *rating* associati a questi k documenti più simili (1 per "gradito", 0 per "non gradito"), viene calcolata una media pesata utilizzando i valori di similarità calcolati al passo precedente. Il risultato di questa media, compreso tra 0 e 1, funziona come una previsione del *rating* per l'articolo nel test set.
Una volta previsto il *rating* per ogni item in D_U, esso viene usato per ordinare i documenti: da quelli con *rating* previsto più alto a quelli con *rating* previsto più basso. Il problema della raccomandazione viene ricondotto quindi ad un problema di ordinamento. Se dovessimo infatti scegliere quali articoli proporre all'utente verrebbero selezionati quelli in cima alla lista, il cui *rating* previsto è più elevato.
Per valutare l'efficacia di questo ordinamento si è deciso di utilizzare l'indice NDCG@20, che misura la capacità del metodo di restituire, per l'utente in questione, articoli rilevanti alle prime posizioni della lista; dove per articoli rilevanti si intendono articoli graditi, o con *rating 1*. In particolare l'NDCG@20 prende in considerazione le prime 20 posizioni dell'ordinamento proposto, che vengono confrontate con le prime venti in un ordinamento ideale (articoli con *rating 1* in cima, con *rating 0* sotto). Per costruzione infatti conosciamo quali, tra gli articoli in D_U, sono rilevanti (*rating* 1, graditi), e quali non lo sono (*rating* 0, non graditi).
Il procedimento viene quindi ripetuto per ogni *user*, e per ognuno di essi viene misurata, tramite l'indice NDCG@20, l'efficacia dell'ordinamento. Infine verrà calcolata una media dei valori osservati per l'indice. Il risultato ottenuto è una misura di efficacia media del metodo di raccomandazione sul campione (dipende comunque dalla configurazione delle *features*), e verrà confrontato con il risultato ottenuto per il *Bayes Classifier*. Iterare il procedimento per diversi valori di α ci permette di determinare quale valore (di α) risulta in una maggiore efficacia (con riferimento alla media degli indici), e utilizzare la media corrispondente per il confronto.

Per ogni utente preso in considerazione, per ogni articolo del test set (D_U), devono essere calcolate un numero di misure di similarità pari alla numerosità del training set (D_L). Il tempo d'esecuzione quindi, per un utente, è proporzionale a $|D_U||D_L|$, e l'algoritmo deve essere ripetuto per ogni user nel campione.
Si è deciso quindi di provare ad utilizzare l'algoritmo di *clustering k-means* per

ridurre il tempo d'esecuzione. Selezionato un utente quindi, prima di passare alla raccomandazione, si utilizza l'algoritmo di *clustering* per raggruppare gli articoli in D_L in cluster omogenei. Il *clustering* viene effettuato due volte, per due gruppi di articoli separati: una per gli articoli con *rating* 1, e una per quelli con *rating* 0. In questo modo i cluster ottenuti non sarebbero misti, e il "rating" ad essi associato sarebbe 0 oppure 1, a seconda degli articoli che lo compongono. Una volta ottenuti i cluster, il *Nearest Neighbor Classifier* verrebbe applicato come prima: per ogni articolo di D_U sarebbero individuati i k' (nel nostro caso 3) cluster più simili (utilizzando la similarità coseno, calcolata con i centroidi) per calcolare, come media ponderata, il *rating* previsto.
Raggruppando quindi gli articoli nel 10% dei cluster, il tempo d'esecuzione dell'algoritmo sarebbe proporzionale a $|D_U||D_L|/10$, con l'aggiunta però del tempo necessario ad effettuare il *clustering*.
Per quanto riguarda la scelta dei centroidi iniziali, non potendo ripetere il *clustering* più volte per individuare i migliori (la nostra esigenza è quella di ridurre il tempo d'esecuzione), abbiamo adottato un metodo che non fosse casuale. Nel nostro caso solo il primo centroide viene scelto casualmente: i successivi vengono scelti in modo iterativo come gli articoli più lontani dai centroidi già scelti. Come misura di distanza è stata utilizzata, di nuovo, la similarità coseno.
Per il centroide del cluster viene utilizzata la stessa rappresentazione degli articoli: un vettore di *features* dato dalla somma degli *item profiles* (vettori di punteggi TF-IDF) degli articoli che fanno parte del cluster.

4.3 Bayes Classifier

Il secondo metodo scelto per la raccomandazione è il *Bayes Classifier*, particolarmente adatto al contesto in cui i *ratings* sono binari. Dato un utente, l'obiettivo è quello di stimare le probabilità, per ogni documento del suo *test set* D_U, che esso gli piaccia, condizionatamente al vettore binario d-dimensionale $(x_1...x_d)$, indicante la presenza o meno dei termini del vocabolario nel documento. d denota, in questo caso, il numero di *features* uniche che compaiono nei documenti in D_L e nel documento per cui sta venendo stimata la probabilità. L'i-esima posizione fa riferimento all'i-esima *feature*, e il corrispondente valore nel vettore, 0 o 1, indica la presenza (con 1) o l'assenza (con 0) di essa nel documento preso in considerazione. Non vengono quindi utilizzati, con questo secondo metodo, i punteggi TF-IDF. Essendo i *ratings* binari, il modello che stiamo utilizzando prende il nome di *Bernoulli model*.
Si indica con $\bar{X}$ il documento in D_U, il cui *rating* binario è espresso con $c(\bar{X})$.
Nel modello utilizzato la probabilità che lo *user* gradisca $\bar{X}$ ($c(\bar{X}) = 1$, *rating* 1) viene stimata come probabilità condizionata al vettore d-dimesionale associato a $\bar{X}$, assumendo che le occorrenze delle parole nel documento siano eventi, condizionatamente al *rating*, indipendenti (*Naive assumption*):

$$P(c(\bar{X}) = 1|x_1...x_d) = \frac{P(c(\bar{X}) = 1)P(x_1...x_d|c(\bar{X}) = 1)}{P(x_1...x_d)} \tag{4}$$

Per l'assunzione fatta sull'indipendenza si ha che:

$$P(x_1...x_d|c(\bar{X}) = 1) = \prod_{i=1}^{d} P(x_i|c(\bar{X}) = 1) \tag{5}$$

Per stimare la probabilità condizionata è necessario determinare il valore di ogni componente della formula. La componente $\mathrm{P}(x_1...x_d)$ (denotata con $\frac{1}{K}$) può essere determinata sfruttando una proprietà delle misure di probabilità. Infatti, sapendo che la somma delle probabilità per le due istanze del rating deve essere pari ad 1, è possibile affermare che:

$$K\big[P(c(\bar{X}) = 1)\prod_{i=1}^{d} P(x_i|c(\bar{X}) = 1) + P(c(\bar{X}) = 0)\prod_{i=1}^{d} P(x_i|c(\bar{X}) = 0)\big] = 1 \tag{6}$$

e quindi, isolando K, questo può essere riscritto in funzione di altre componenti. Per poter passare alla stima delle probabilità condizionate è quindi necessario stimare alcune quantità intermedie: $P(c(\bar{X}) = 1)$, $P(c(\bar{X}) = 0)$, le $P(x_i|c(\bar{X}) = 1)$ e le $P(x_i|c(\bar{X}) = 0)$.
$P(c(\bar{X}) = 1)$ può essere stimata come la frazione di documenti graditi (con un *rating* pari ad 1) nel training set dello *user*, riducendo l'*overfitting* utilizzando una tecnica di *lisciamento laplaciano* aggiungendo al numeratore e al denominatore del rapporto un parametro $\alpha > 0$ con un valore basso. Quindi otteniamo:

$$P(c(\bar{X}) = 1) = \frac{|D_L^+| + \alpha}{|D_L| + 2\alpha} \tag{7}$$

La stessa formula, stavolta con riferimento agli articoli non graditi, può essere utilizzata per stimare la $P(c(\bar{X}) = 0)$.
Le probabilità condizionate $P(x_i|c(\bar{X}) = 1)$, riferite alla probabilità che le *features* compaiano o meno nel documento, vengono invece stimate come la frazione, nei documenti graditi del training set dello *user*, nei quali la i-esima *feature* assume valore x_i. Nella formula seguente $q^+(x_i)$ rappresenta il numero di istanze, tra i documenti con *rating* 1 nel training set, in cui la i-esima *feature* assume valore x_i, mentre $|D_L^+|$ è il numero di documenti con *rating* 1. β è invece il parametro utilizzato per il *lisciamento laplaciano*.

$$P(x_i|c(\bar{X}) = 1) = \frac{q^+(x_i) + \beta}{|D_L^+| + 2\beta} \tag{8}$$

Allo stesso modo può essere stimata $P(x_i|c(\bar{X}) = 0)$, facendo riferimento stavolta alle istanze tra i documenti con *rating* 0.
Una volta stimate le probabilità condizionate che il *rating* sia 1 per ogni documento del test set, è possibile ordinarli in base ad essa. L'efficacia dell'ordinamento viene poi studiata tramite il calcolo dell'indice *NDGC@20*. Iterando il procedimento per ogni *user* è possibile calcolare una media delle misure di efficacia. Ripetendo il procedimento per diversi valori di α, il parametro che modifica

il punteggio TF-IDF, è possibile individuare, usando il valore di efficacia media, quale valore di α porta ad una configurazione di features più favorevole (per il *Bayes Classifier*). Il valore medio migliore viene poi confrontato con il risultato migliore per il *Nearest Neighbor Classifier*. Il metodo che porterà ad avere un *NDGC@20* medio più elevato sul campione risulterà essere stato il più efficace.

5 Esperimenti

Iniziando con la *feature selection* base, effettuata tenendo, per ogni articolo, le 100 *features* con punteggio TF-IDF più alto, i risultati sono i seguenti:

Tabella 2. RISULTATI BAYES CLASSIFIER AL VARIARE DI α

$\alpha =$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
NDCG@5	0.655	0.655	0.660	0.661	0.652	0.655	0.657	0.652	0.646	0.653	0.658
NDCG@10	0.651	0.651	0.653	0.658	0.653	0.651	0.648	0.644	0.641	0.645	0.65
NDCG@20	0.629	0.631	0.632	0.634	0.631	0.632	0.632	0.628	0.626	0.629	0.612

Tabella 3. RISULTATI NEAREST NEIGHBOR CLASSIFIER AL VARIARE DI α

$\alpha =$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
NDCG@5	0.750	0.746	0.739	0.728	0.719	0.717	0.713	0.703	0.68	0.65	0.57
NDCG@10	0.723	0.722	0.717	0.714	0.708	0.699	0.693	0.679	0.661	0.638	0.566
NDCG@20	0.684	0.680	0.676	0.675	0.67	0.667	0.665	0.657	0.64	0.618	0.560

I risultati migliori rispetto all'NDCG@20, per i due metodi, sono stati ottenuti utilizzato, rispettivamente, α pari a 0.3 e 0.
Confrontando i risultati considerando, per i due metodi, quello nella migliore configurazione delle *features*, il *Nearest Neighbor Classifier* risulta essere stato il più efficace (0.684 contro 0.634). Inoltre, il N.N.C. risulta più efficace anche rispetto gli indici NDCG@5 e NDCG@10.
Per quanto riguarda invece il tempo impiegato (misurato in minuti reali) il *Bayes Classifier* è notevolmente più veloce: per una singola configurazione di *features*, un singolo α, il B.C. ha impiegato 3.86 minuti, contro i 9.408 minuti del N.N.C.. Questi sono comunque tempi medi, una singola iterazione di alpha è più lenta: per $\alpha = 1$ la raccomandazione è molto più rapida, e questo abbassa i tempi medi. Sembra quindi esserci un trade-off tra tempo impiegato ed efficacia della raccomandazione.
Ricorrendo al *clustering* per il *Nearest Neighbor Classifier* il tempo impiegato aumenta: l'overhead dovuto alla costruzione dei cluster è sufficiente a negare completamente il risparmio dovuto alla riduzione del numero di misure di distanza da calcolare. Questo è probabilmente dovuto alle dimensioni ridotte dei

test set. Il risultato migliore è per $\alpha = 0$: NDCG@5 = 0.635, NDCG@10 = 0.624, NDCG@20 = 0.61, ottenuto in 9.539 minuti. Anche l'efficacia è quindi molto inferiore a quella osservata senza fare *clustering*.

Come cambiano i risultati riducendo il numero di *features* con il metodo basato sulla *statistica* X^2? Dimezzando le parole chiave per ogni articolo, i risultati sono i seguenti (riportati i risultati migliori rispetto ad α e all'NDCG@20 medio):

Tabella 4. Risultati dopo *supervised feature selection*

	α	NDCG@5	NDCG@10	NDCG@20
B.C.	$\alpha = 0.8$	0.657	0.637	0.603
N.N.C.	$\alpha = 0$	0.72	0.700	0.673

Per quanto riguarda il confronto: il N.N.C. riporta risultati migliori per ognuno dei 3 indici medi, come nel caso precedente. Nonostante però l'efficacia della raccomandazione si sia ridotta leggermente, c'è stato un notevole guadagno in termini di tempo: 5.171 minuti impiegati per il N.N.C. e 1.936 per il B.C. (per ogni valore di α). Il tempo impiegato si è quasi dimezzato, anche se in esso è compreso il tempo necessario per l'ulteriore *feature selection*. Questo può essere spiegato con il fatto che il calcolo della distanza coseno (usata nel N.N.C) e la stima della probabilità (nel B.C.) dipendano entrambi dal numero di *features* utilizzate.
Utilizzando la *supervised feature selection* nel caso di *clustering* l'efficacia rimane pressoché identica (per $\alpha = 0$, NDCG@5 = 0.627 ,NDCG@10 = 0.62, NDCG@20 = 0.61), ma con una notevole riduzione di tempo, 5.114 minuti per ogni valore di α.

In conclusione il *Nearest Neighbor Classifier* si è rivelato il più efficace, anche se il tempo impiegato è molto più elevato. La riduzione di *features* con il metodo del X^2 permette comunque di ridurre notevolmente il tempo di esecuzione, mantenendo buoni livelli di efficacia, mentre il *clustering* non ha dato i risultati sperati.
Dare diverso peso ai campi, tramite α, non si è rivelato efficace: per il N.N.C. i risultati migliori si ottengono per $\alpha = 0$ (stesso peso per ogni campo), mentre i risultati del B.C. erano sempre tutti molti simili per ogni valore (ad eccezione di $\alpha = 1$).
I risultati comunque devono essere letti ricordando la debolezza dell'assunzione iniziale, necessaria per determinare le preferenze degli utenti, che sicuramente ha influito negativamente sui risultati di efficacia dei metodi impiegati.

Riferimenti bibliografici

1. Charu C. Aggarwal: Recommender Systems. Springer, NY, USA (2016)
2. Leskovec, J., Rajaraman, A., D. Ullman, J.: Mining of Massive Datasets. Palo Alto, CA (July, 2019)
3. Microsoft: Introduction to MIND and MIND-small datasets. Github. https://github.com/msnews/msnews.github.io/blob/master/assets/doc/introduction.md.