Il modello vettoriale

Argomenti della lezione

- Limiti del modello booleano.
- Il modello vettoriale.
- Combinazione lineare con pesi.
- TF/IDF.
- Calcolo efficiente del ranking.

Limiti del modello booleano

- Nel modello booleano, un documento soddisfa le condizioni oppure no.
- Questo modello può essere ragionevole solo per degli utenti esperti che conoscano perfettamente la collezione di documenti e le proprie necessità.
- Una query può ritornare migliaia di risultati, ma la maggior parte degli utenti non vogliono scorrere migliaia di voci.
- Inoltre una eventuale riformulazione della query provoca il ricalcolo dell'intero risultato, con evidenti problemi prestazionali.

Il modello vettoriale

- Idea: invece di cercare di predire se un documento è rilevante o no, ordiniamo i documenti secondo il loro grado di *similarità* rispetto alla query.
- Dobbiamo quindi assegnare uno score (ad esempio compreso tra 0 e 1) ad ogni documento della collezione rispetto alla query formulata.
- Successivamente ritorneremo un elenco di documenti ordinato in base alla probabilità che siano di interesse per l'utente.

Combinazione lineare delle zone

- Consideriamo una possibile ricerca per zone: *"sorting in Title AND smith in Bibliography AND recursive in Body"*.
- Assegniamo ad ogni sottoquery (ad es. *"sorting in Title"*) un peso e calcoliamo lo score di un documento sommando i risultati delle sottoquery.

  Score = 0.6 * <*sorting in Title*> + 0.3 * <*smith in Bibliography*> +
  0.1 * <*recursive in Body*>
- Ogni sottoquery può assumere il valore 0 o 1 e la somma dei pesi è 1, quindi lo score complessivo sarà compreso tra 0 e 1.
- Verranno visualizzati in risposta alla query i *k* documenti aventi lo score più alto.

Come scegliere i pesi

- In teoria, il sistema potrebbe mettere a disposizione un'interfaccia utente che permetta all'utente stesso di scrivere la propria espressione.
- Di nuovo, ciò è ragionevole solo se si ha a che fare con utenti esperti.
- In generale, il query parser leggerà una query booleana sottomessa dall'utente e calcolerà i pesi da applicare, sulla base di regole generali e/o di uno studio sul comportamento precedente dell'utente.

Scoring basato sulla densità

- Sia nel caso di query su zone che di query semplici dobbiamo tenere in conto il problema della densità.
- Supponiamo di voler rispondere alla query *"bill OR rights"*. I documenti che contengono entrambe le parole verranno valutati come quelli che ne contengono solo una.
- Analogamente, un documento che contiene 100 volte la parola cercata verrà valutato come un'altro che la contiene solo 5 volte.
- Informalmente, se un documento parla di un certo argomento *più* di un altro documento, probabilmente risponde meglio alle necessità dell'utente e quindi dovrebbe avere uno score più alto.

Rappresentazione di documenti tramite vettori

- Come già visto, l'appartenenza di una parola ad un certo documento può essere valutata rappresentando il documento come un vettore $\{0,1\}^M$, dove M è la dimensione del dizionario.

Opere Termini	Antonio e Cleopatra	Giulio Cesare	La Tempesta	Amleto	Otello	Macbeth
Antonio	1	1	0	0	0	1
Bruto	1	1	0	1	0	0
Cesare	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0

Misura della sovrapposizione

- Allo stesso modo, anche una query può essere rappresentata tramite un vettore $\{0,1\}^M$.
- Lo score di un documento può essere quindi misurato come la *sovrapposizione* tra i due vettori.
- Ad esempio, relativamente alla query *"idi di marzo"*:
  - il documento *"Giulio Cesare"* avrà uno score pari a 3 (poichè contiene tutte e tre le parole contenute nella query);
  - Qualche documento avrà uno score pari a 2 (poichè contiene le parole *"di"* e *"marzo"*);
  - Tutti gli altri documenti avranno uno score pari a 1 (poichè contengono la parola *"di"*).

Problemi della misura della sovrapposizione

- La misura della sovrapposizione non può essere considerata soddisfacente, poiché non considera:
  - Quante volte un documento contiene un certo termine;
  - Quanti documenti contengono un certo termine (ad es. *"di"* è molto più comune di *"idi"*);
  - La lunghezza dei documenti e delle query (quindi gli score non sono normalizzati).

Documenti come vettori di interi

- Il primo problema può essere risolto rappresentando i documenti come vettori di interi, invece che di booleani:
  - $T[i,j] = w$ se il documento d_j contiene il termine t_i w volte.

Opere Termini	Antonio e Cleopatra	Giulio Cesare	La Tempesta	Amleto	Otello	Macbeth
Antonio	157	73	0	0	0	0
Bruto	4	157	0	1	0	0
Cesare	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0

Frequenza dei termini

- Non abbiamo ancora risolto il problema dei termini più comuni: se calcoliamo lo score sommando il numero di occorrenze di ogni parola all'interno di un documento, probabilmente lo score più alto sarà quello del documento che contiene più volte il termine *"di"*.
- Inoltre i documenti più lunghi sono favoriti, perché hanno maggiori possibilità di contenere i termini della query (e di contenerli più volte).
- La situazione migliora utilizzando come misura, invece del conteggio dei termini, la frequenza dei termini
  - $tf_{t,d}$ (term frequency) = numero di occurrenze di *t* in *d* diviso per il numero totale di parole in *d*.

Calcolo del matching

- Rappresentiamo i documenti della collezione e la query tramite dei vettori $\mathbb{N}^M$.
- Esprimiamo il matching tra un documento e la query come il prodotto scalare dei relativi vettori:
  - $q \bullet d = \Sigma_i \, tf_{i,q} \, x \, tf_{i,d}$
- Al posto di *tf* è spesso utilizzata un'altra misura, chiamata *weighted term frequency* (*wf*), che tiene conto del fatto che, sebbene l'importanza di un documento cresca col numero di occorrenze di un certo termine, tale crescita non è (probabilmente) direttamente proporzionale:
  - $wf_{t,d} = tf_{t,d} > 0 \; ? \; 1 + \log tf_{t,d} : 0$
- Ma come risolvere il problema dei termini comuni?

Document Frequency e misura *tf / idf*

- Per Document Frequency si intende il numero di documenti che contengono un certo termine.
- L'inverso del Document Frequency (Inverse Document Frequency, *idf*), cioè la *rarità* di un termine all'interno della collezione, è una buona misura della significatività di un termine.
- Solitamente viene usata la seguente formula:

$$idf_i = 1 / \log (N / df_i)$$

  dove:
  - N = numero totale di documenti della collezione;
  - df_i = numero di documenti che contengono il termine i.
- Ad ogni termine della query viene assegnato un peso in base ad una misura combinata di *tf* e *idf* (*tf / idf*):

$$w_{i,d} = tf_{i,d} \times \log (N / df_i)$$

Collezione come spazio vettoriale

- Ogni documento può essere quindi visto come un vettore di valori *tf / idf* (o *wf / idf*).
- Di conseguenza, l'intera collezione può essere vista come uno spazio vettoriale, i cui assi sono i termini, contenente i documenti.
- Le query possono essere viste come dei brevi documenti, e quindi anch'esse sono dei vettori appartenenti a questo spazio.
- Abbiamo quindi bisogno di una nozione di *prossimità* tra vettori, sulla quale basarci per assegnare uno score ad ogni documento.
- Tale nozione di prossimità ci permetterebbe anche, dato un documento, di trovare altri documenti "simili": documenti che sono "vicini" nello spazio vettoriale parlano "più o meno" della stessa cosa.

Proprietà auspicabili della nozione di prossimità

- Se d_1 è vicino a d_2, allora d_2 è vicino a d_1 (commutatività).
- Se d_1 è vicino a d_2 e d_2 è vicino a d_3, allora d_1 è vicino a d_3 (transitività).
- Nessun documento è più vicino a d_1 di d_1 stesso.

Similarità cosenica

- Esprimiamo la distanza tra i vettori d_1 e d_2 tramite il coseno dell'angolo tra loro.
- Per tenere conto della lunghezza dei documenti ed evitare che i documenti più lunghi ottengano un peso maggiore, normalizziamo il risultato dividendo per la lunghezza dei vettori.

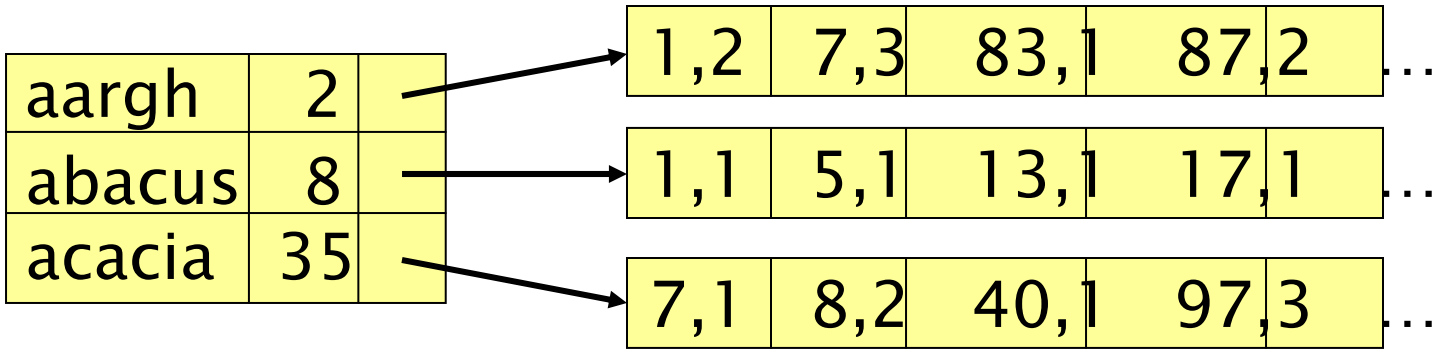
$$sim(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\left|\vec{d}_j\right|\left|\vec{d}_k\right|} = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,k}^2}}$$

Calcolo efficiente del ranking

- Riepilogando, per rispondere ad una query utilizzando il modello vettoriale dobbiamo trovare i *k* documenti appartenenti alla collezione che sono più "vicini" alla query, cioè dobbiamo trovare i *k* valori più alti del coseno tra i vettori della query e del documento.
- Vogliamo rendere tale operazione efficiente, cioè vogliamo:
  - Calcolare efficientemente un singolo coseno;
  - Scegliere efficientemente i *k* valori più alti del coseno: riusciamo a farlo senza calcolare tutti gli *N* valori?

Calcolo efficiente di un singolo coseno

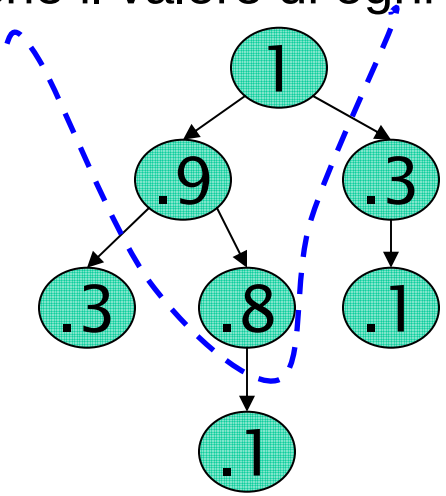
- Per ogni termine, memorizziamo nel dizionario il Document Frequency.
- Per ogni termine in ogni documento, memorizziamo nei posting il Term Frequency (in realtà di solito si memorizza la semplice frequenza).



- Ciò comporta uno spreco di spazio; si possono utilizzare le tecniche di compressione già viste.

Scelta efficiente dei *k* valori di coseno più alti

- Il metodo più semplice è ordinare i valori e successivamente prendere i primi *k*.
- In realtà non è necessario ordinare completamente gli *N* risultati; possiamo scegliere i *k* valori più alti costruendo un albero binario avente la proprietà che il valore di ogni nodo è maggiore dei valori dei nodi figli.



- Costo: 2*N* (costruzione dell'albero) + 2*k*log*N* (scelta dei migliori *k*); per *N* = 1M e *k* = 100: ~ 10% del costo dell'ordinamento.

Diminuzione del numero di candidati (1)

- Il collo di bottiglia del processo è però il calcolo del coseno per gli *N* documenti.
- Possiamo accelerare questa fase evitando di calcolare il coseno per quei documenti che sicuramente avranno un valore uguale a zero, cioè quei documenti che non contengono alcuna delle parole richieste nella query.
- Dobbiamo cioè calcolare il valore del coseno solo per i documenti appartenenti all'unione dei posting dei termini contenuti nella query.
- Possiamo ulteriormente migliorare le prestazioni riducendo il calcolo del coseno solo ai documenti contenenti almeno una parola "rara" (cioè con un alto valore di *idf*).

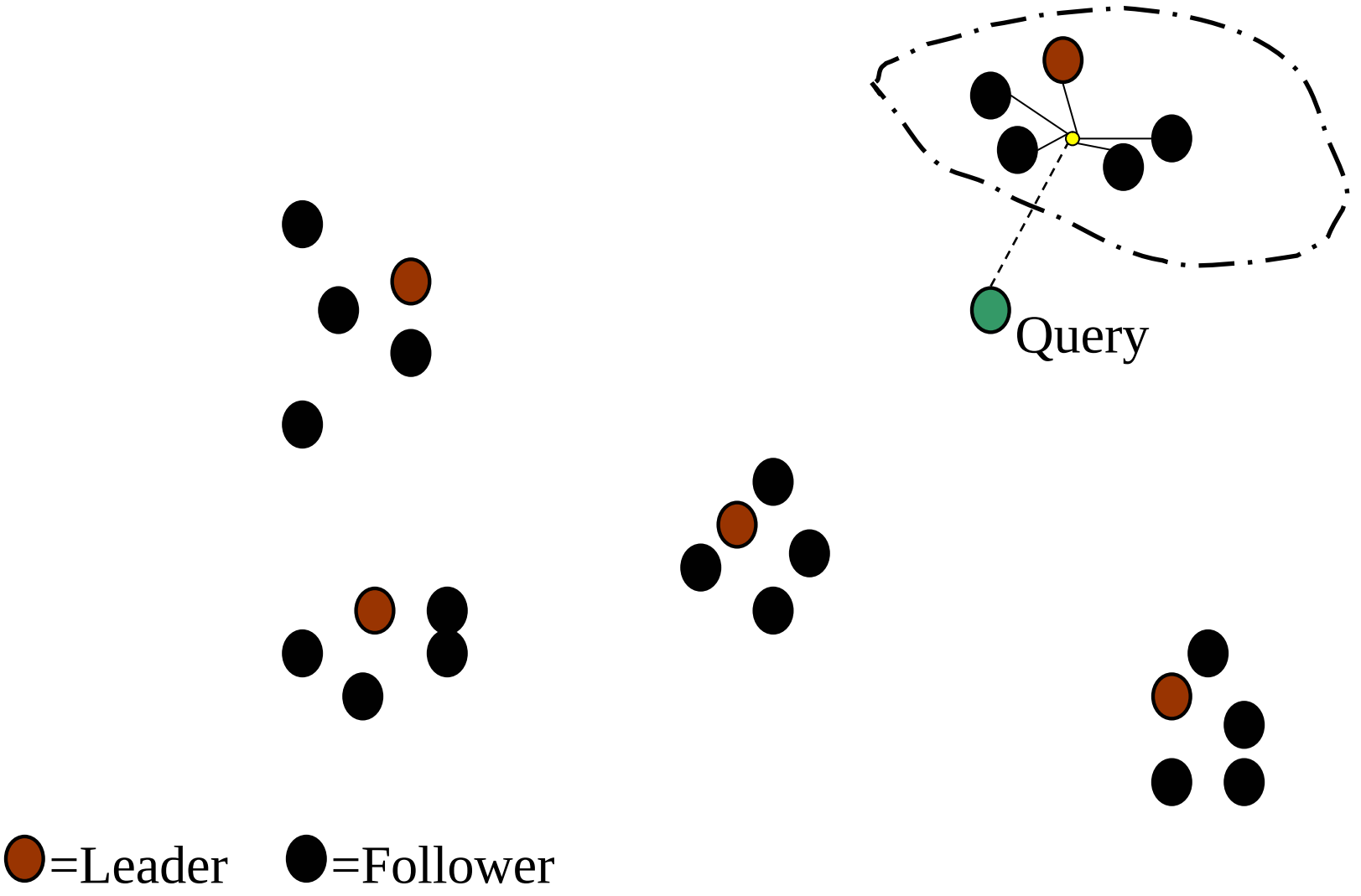
Diminuzione del numero di candidati (2)

- Un'altra tecnica si basa sul calcolo a priori degli *m* documenti più simili per ognuno degli *M* termini indicizzati, scegliendo $m > k$.
- In pratica si "simulano" *M* query composte da un solo termine, ottenendo così una "lista dei preferiti" per ogni termine.
- Quando si deve eseguire una query di lunghezza *t*, si procede come segue:
  - Calcoliamo l'insieme *S* dato dall'unione delle *t* "liste dei preferiti", $|S| < mt$;
  - Calcoliamo il coseno solo per i documenti contenuti in *S* e scegliamo i migliori *k*.

Un'altra tecnica: cluster pruning

- Fase di pre-processing:
  - Scelta casuale di $\sqrt{N}$ documenti, chiamati *leader*;
  - Per ognuno degli altri documenti, calcoliamo il leader più simile;
  - I documenti associati ad un leader vengono chiamati *follower*; mediamente avremo $\sqrt{N}$ follower per ogni leader.
- Esecuzione di una query:
  - Calcolo del leader più simile;
  - Scelta dei k documenti da ritornare tra i follower del leader scelto.
- Possibili varianti:
  - Associare ogni follower a più di un leader;
  - All'esecuzione della query scegliere più di un leader e cercare i documenti da ritornare tra i follower di tutti i leader scelti.

Cluster pruning: visualizzazione



Riepilogo

- Limiti del modello booleano.
- Il modello vettoriale.
- Combinazione lineare con pesi.
- TF/IDF.
- Calcolo efficiente del ranking.

Risorse per questa lezione

- MIR ch. 2.5.3