TLN:

text categorization con risorse semantiche: il metodo di Rocchio

Daniele Radicioni

terminologia

- *documento* è un'unità di testo indicizzata nel sistema e disponibile per la ricerca.
- *collezione* è un insieme di documenti, utilizzati per soddisfare le richieste dell'utente.
- *termine* è l'elemento (dal singolo elemento lessicale a intere *phrases*) che occorre nella collezione.
- *interrogazione* è la richiesta dell'utente, espressa come insieme di termini.

vector space model

- il valore di ogni feature è detto *term weight*, ed è una funzione della *frequenza* del termine *nel documento*.
  - e.g., in una ricetta di cucina sul *pollo fritto* trovata sul Web, i termini *pollo, fritto, olio, pepe* occorrono rispettivamente *8,2,7,4* volte.
  - usando questa frequenza come *peso del termine*, il vettore per questo documento (documento *j*) sarà espresso come

$$\overrightarrow{d}_j = (8, 2, 7, 4)$$

il vettore del documento

- in generale, il vettore di un documento d_j è rappresentato come

$$\overrightarrow{d}_j = (w_{1,j}, w_{2,j}, w_{3,j}, \ldots, w_{n,j})$$

- dove l'elemento $w_{i,j}$ individua l'elemento *i* fra gli *N* complessivamente presenti nella *collezione*, e *j* è il *j-esimo* documento della collezione.

il vettore della query

- analogamente, anche la query è rappresentata come un vettore; la query *pollo fritto* potrebbe essere

$$\overrightarrow{q} = (1, 1, 0, 0)$$ ← (pollo, fritto, olio, pepe)

- e in generale,

$$\overrightarrow{q} = (w_{1,q}, w_{2,q}, w_{3,q}, \ldots, w_{n,q})$$

un'altra ricetta

- consideriamo un'altra ricetta per il *pollo cotto*, contenuta nel documento d_k, che potrebbe essere rappresentata dal vettore

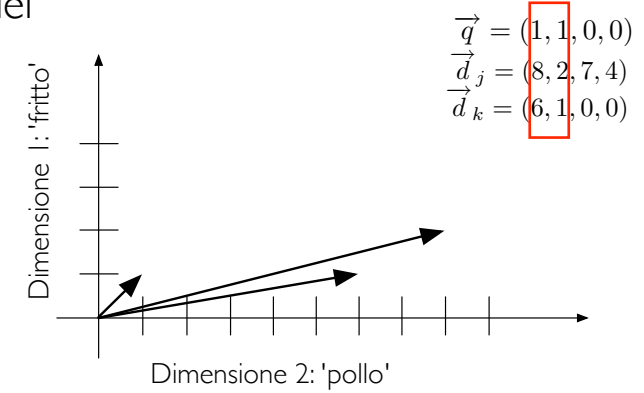
$$\overrightarrow{d}_k = (6, 1, 0, 0)$$

- vorremmo che la query *pollo fritto* restituisse il documento d_j, (la ricetta del pollo fritto), piuttosto che il documento d_k (pollo cotto)

FV Model

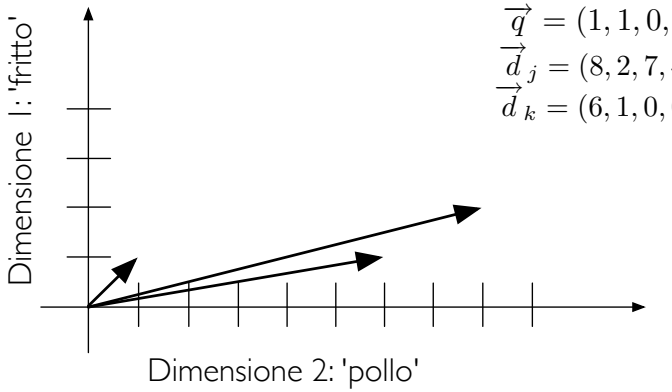
- è utile immaginare le features usate per rappresentare i documenti e le interrogazioni in questo modello come dimensioni in uno spazio multidimensionale, in cui i pesi delle features servono a individuare i documenti.
  - una *query utente* tradotta in un vettore denota un punto in quello spazio.
- i documenti che sono *vicini* alla query sono in qualche modo più *rilevanti* (pertinenti?) ai fini della query, di quanto siano i documenti lontani.

FV Model



rappresentazione delle prime due dimensioni del vector model.

similarity



$$\vec{q} = (1, 1, 0, 0)$$
$$\vec{d}_j = (8, 2, 7, 4)$$
$$\vec{d}_k = (6, 1, 0, 0)$$

- se per misurare la somiglianza fra i vettori consideriamo gli *angoli* fra i vettori, la query è più simile al documento *j*.

metriche di similarità

- invece dell'angolo si usa il *coseno* dell'angolo fra i vettori, calcolato come:

$$\text{sim}(\vec{q}, \vec{d}_j) = \frac{\sum_{i=1}^{N} w_{i,q} \times w_{i,j}}{\sqrt{\sum_{i=1}^{N} w_{i,q}^2} \times \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

- il coseno fra due documenti identici sarà 1, mentre il coseno fra due documenti *ortogonali* (privi di termini in comune) sarà 0.

normalizzazione del dot product

- il coseno può essere pensato come *prodotto scalare*

$$\text{sim}_{dot-product}(\vec{v}, \vec{w}) = \vec{v} \cdot \vec{w} = \sum_{i=1}^{N} v_i \times w_i$$

- normalizzato per la lunghezza dei vettori, definita come

$$|\vec{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

- quindi

$$\text{sim}_{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \times \sqrt{\sum_{i=1}^{N} w_i^2}}$$

pesatura dei termini

- finora abbiamo assunto che i *pesi* dei termini fossero determinati semplicemente con il conto della frequenza dei vari termini nei documenti.
- in realtà due fattori sono critici nell'attribuzione del giusto 'peso' ai termini:
  1. la *frequenza*;
  2. la frequenza in un documento ma la non-frequenza a livello dell'intera collezione

IDF: inverse document frequency

- l'idea è che i termini che sono presenti in pochi documenti sono utili per distinguere quei documenti dal resto della collezione (i termini che ricorrono in tutti i documenti non sono utili: e.g., *stop words*).
- pertanto l'IDF è definito per mezzo del rapporto N/n_i, dove N è il numero totale di documenti nella collezione, e n_i è il numero di documenti in cui il termine i occorre.

inverse document frequency

- quanti meno sono i documenti in cui un termine occorre, tanto più alto è il peso. il peso minimo è 1, corrispondente a termini che occorrono in tutti i documenti.
- dato l'elevato numero di documenti in varie collezioni, il rapporto è 'schiacciato' con una funzione logaritmica.

$$\mathrm{idf}_i = \log\left(\frac{N}{n_i}\right)$$

inverse document frequency

- combinando *tf* e *idf* otteniamo il seguente schema:

$$\mathrm{w}_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_{i,j}$$

- utilizzando lo schema *tf-idf* il peso del termine *i* nel vettore del documento *j* è calcolato come il prodotto della sua frequenza totale in *j* per il logaritmo della sua *idf* nella collezione

$$\mathrm{sim}(\vec{q}, \vec{d}) = \frac{\sum_{w \in q,d} \mathrm{tf}_{w,q}\mathrm{tf}_{w,d}(\mathrm{idf}_w)^2}{\sqrt{\sum_{q_i \in q}(\mathrm{tf}_{q_i,q}\mathrm{idf}_{q_i})^2} \times \sqrt{\sum_{d_i \in d}(\mathrm{tf}_{d_i,d}\mathrm{idf}_{d_i})^2}}$$

The Rocchio Method

profilo di una classe di documenti

- Some linear classifiers consist of an *explicit profile* (or *prototypical* document) of the category.
- Learning a linear classifier is often preceded by local *term space reduction*; in this case, a profile of c_i is a weighted list of the terms whose presence or absence is most useful for discriminating c_i.

il centroide

- Let *D* be the set of documents and *T* be the dictionary i.e. the set of all different terms occurring in *D*.

$$T = \{t_1, \ldots, t_m\}$$

- then the absolute frequency of term $t \in T$ in document $d \in D$ is given by *tf(d, t)*.
- We denote the term vectors

$$\vec{t_d} = (\mathrm{tf}(d, t_1), \ldots, \mathrm{tf}(d, t_m))$$

il centroide

- il *centroide* di un insieme di *X* vettori di termini è definito come

$$\vec{t_X} := \frac{1}{|X|} \sum_{\vec{t_d} \in X} \vec{t_d}$$

metodo di Rocchio: intuizione

- A classifier built by means of the Rocchio method rewards
  - the closeness of a test document to the centroid of the positive training examples, and
  - its distance from the centroid of the negative training examples.

metodo di Rocchio

$$\overrightarrow{c_i} = \langle \mathrm{f}_{1i}, \ldots, \mathrm{f}_{|\mathcal{T}|i} \rangle$$

- for each class c_i (with *i* varying over classes) we compute the weight of the *k*-th feature *f* as

$$\mathrm{f}_{ki} = \beta \cdot \sum_{d_j \in POS_i} \frac{w_{kj}}{|POS_i|} - \gamma \cdot \sum_{d_j \in NEG_i} \frac{w_{kj}}{|NEG_i|}$$

$$POS_i = \{d_j \in Tr | \Phi(d_j, c_i) = T\}$$

$$NEG_i = \{d_j \in Tr | \Phi(d_j, c_i) = F\}$$

- β and γ are control parameters that allow setting the relative importance of positive and negative examples
  - if β is set to *1* and γ to *0* the profile of *c* is the centroid of its positive training examples.

raffinamento: i near positives

$$\mathrm{f}_{ki} = \beta \cdot \sum_{d_j \in POS_i} \frac{w_{kj}}{|POS_i|} - \gamma \cdot \sum_{d_j \in NPOS_i} \frac{w_{kj}}{|NPOS_i|}$$

- Open issue: whether the set NEG_i should be considered in its entirety, or whether a well- chosen sample of it, such as the set $NPOS_i$ of near-positives (defined as "*the most positive among the negative training examples*").
- Near-positives are the most difficult documents to tell apart from the positives.

consegna

1. sono dati due set di documenti (italiano vs. inglese; 20 docs x 10 classi, vs. 20 docs x 20 classi) con caratteristiche diverse. sceglierne uno.
2. scrivere un programma che utilizzi una rappresentazione basata sul *feature vector model*;
3. il programma deve costruire i profili (Rocchio) relativi alle classi cui appartengono i documenti, e
4. deve classificare i nuovi documenti (utilizzando come metrica di distanza la *cosine similarity* o una delle metriche derivate) in una delle classi date.

- per ciascuna classe prendiamo il 90% dei documenti per costruire i profili, e il 10% per testare
- opzionale: partizionare in 10 split il dataset, ripetendo la procedura 10 volte ('training' su 90% e test su 10%), calcolando la media dell'accuratezza così ottenuta.

consegna - scelte progettuali

- a proposito del punto 1:
  - quali feature scegliere? sperimentare con i lemmi dei termini in input e con i relativi BabelNet IDs (nel secondo caso è necessario predisporre una qualche forma di WSD).
  - implementando entrambe le alternative, quali differenze nei risultati ci possiamo attendere?
- nell'implementazione del metodo di Rocchio partire assegnando a β e γ i valori di *16* e *4*, rispettivamente.
- algoritmo per calcolare i documenti *NPOS*?