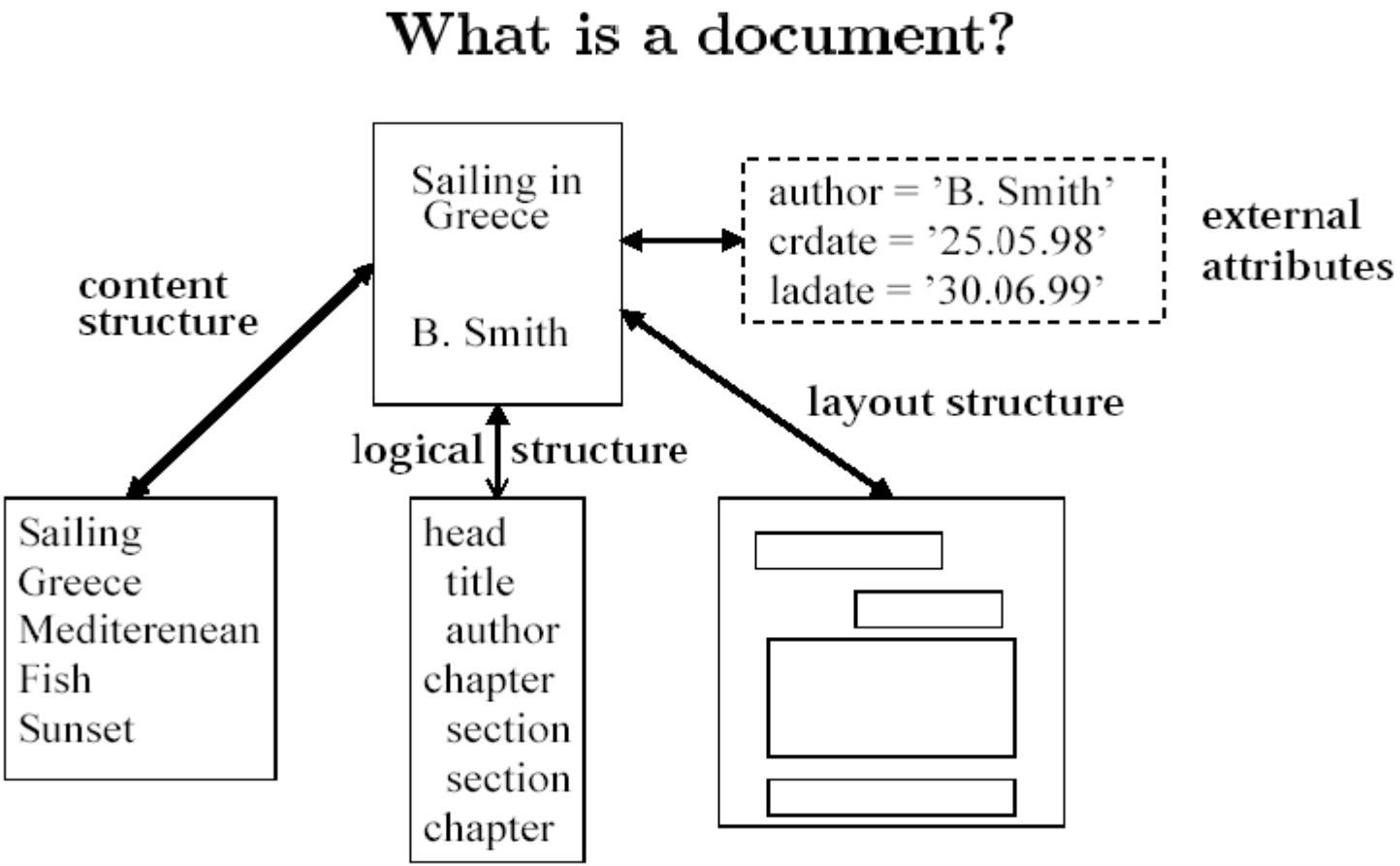
Modelli di Information Retrieval

Roberto Basili

Web Mining & Retrieval

a.a. 2009-2010

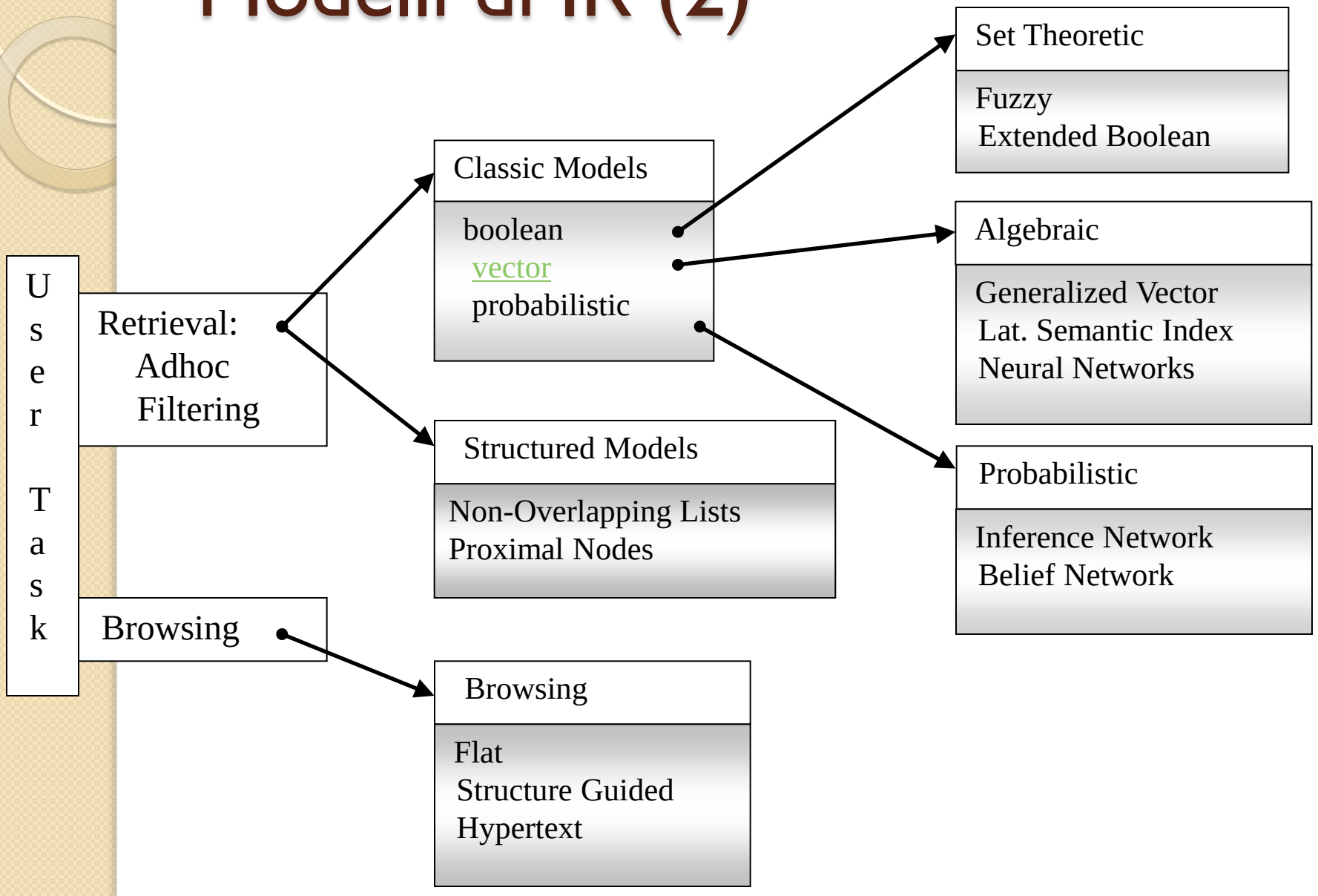
Documenti, Informazioni e Struttura



Modelli di *Retrieval*

- Un modello di IR specifica (almeno) :
  - Rappresentazione del documento
  - Rappresentazione della Query
  - La funzione di Retrieval
- Determina una specifica nozione di *relevance*.
- La nozione di *relevance* puo' essere discreta (e.g. *binaria*) o continua (i.e. *retrieval* in ordine di rilevanza).

Modelli di IR (2)



Classi di Modelli per IR

- Modelli booleani (*set theoretic*)
  - *Extended Boolean*
- Modelli vettoriali (algebrici)
  - *Generalized Vector Space*
  - *Latent Semantic Indexing*
- Modelli Probabilistici

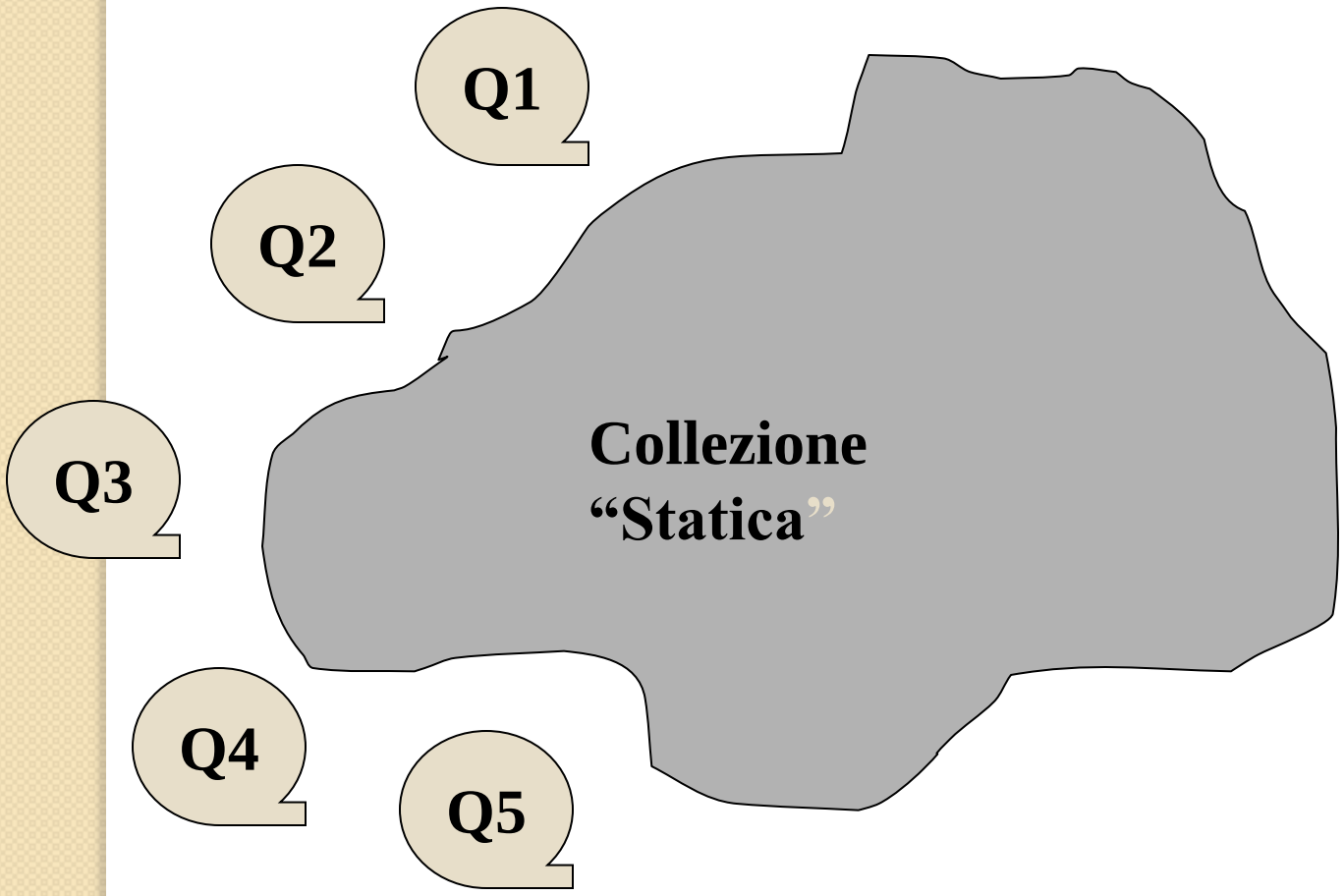
Altri criteri di classificazione

- Modello Logico dei Documenti
  - Tipologia di Indici
  - *Full text*
  - *Full text* + Struttura (e.g. ipertesti)
- Ruolo dell'utente
  - *Retrieval*
  - *Browsing*

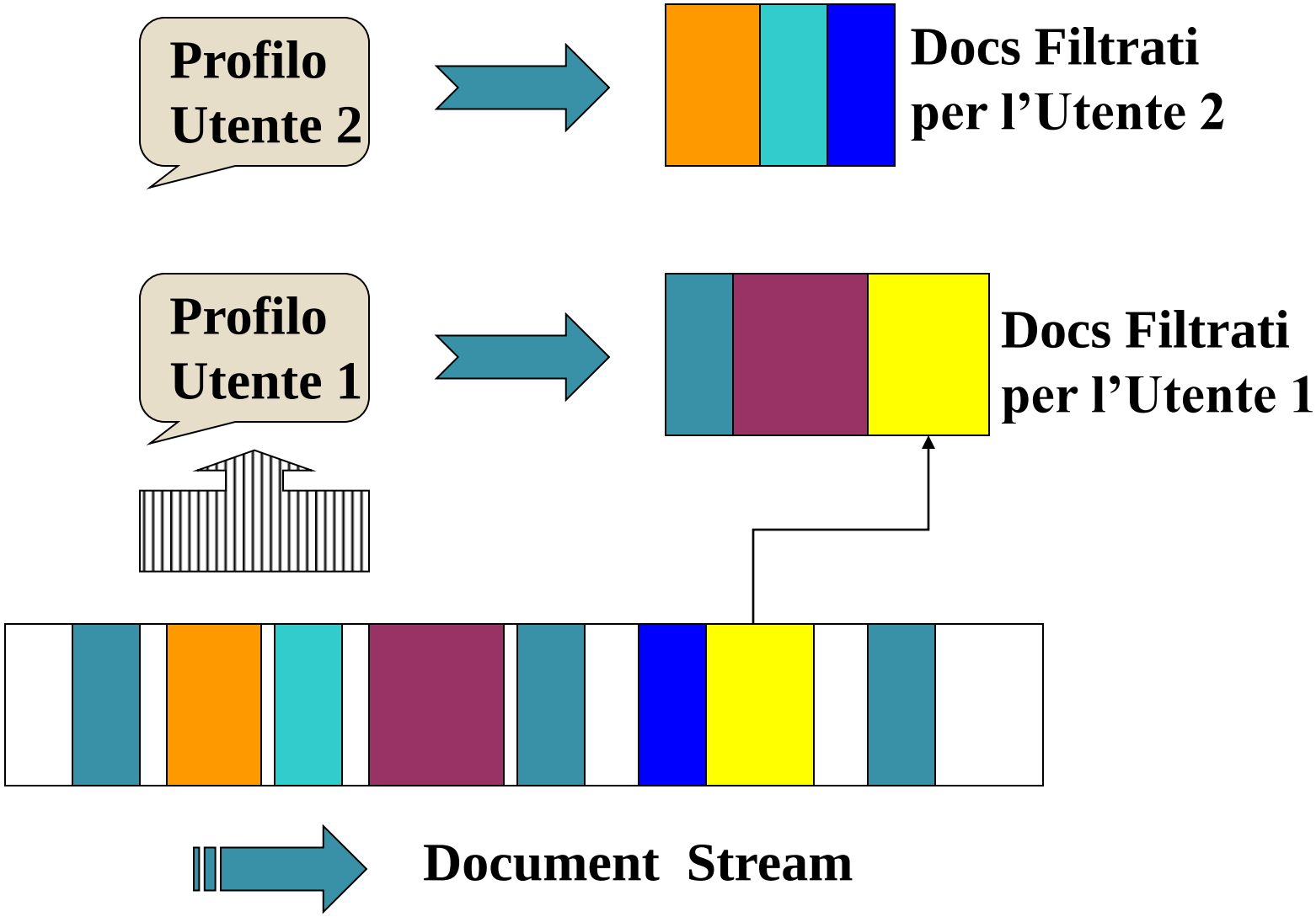
Retrieval Tasks

- *Ad hoc retrieval*: Collezione di Documenti stabile ed interrogazioni variabili.
- *Filtering*: Query (pre)fissate e flussi continui di documenti
  - Profilo Utente: un modello (statico) di preferenze relative.
  - Target: decisione binaria.
- Routing: come il *filtering* ma con funzioni di rilevanza/adesione alle preferenze non binari e generalmente dinamici.

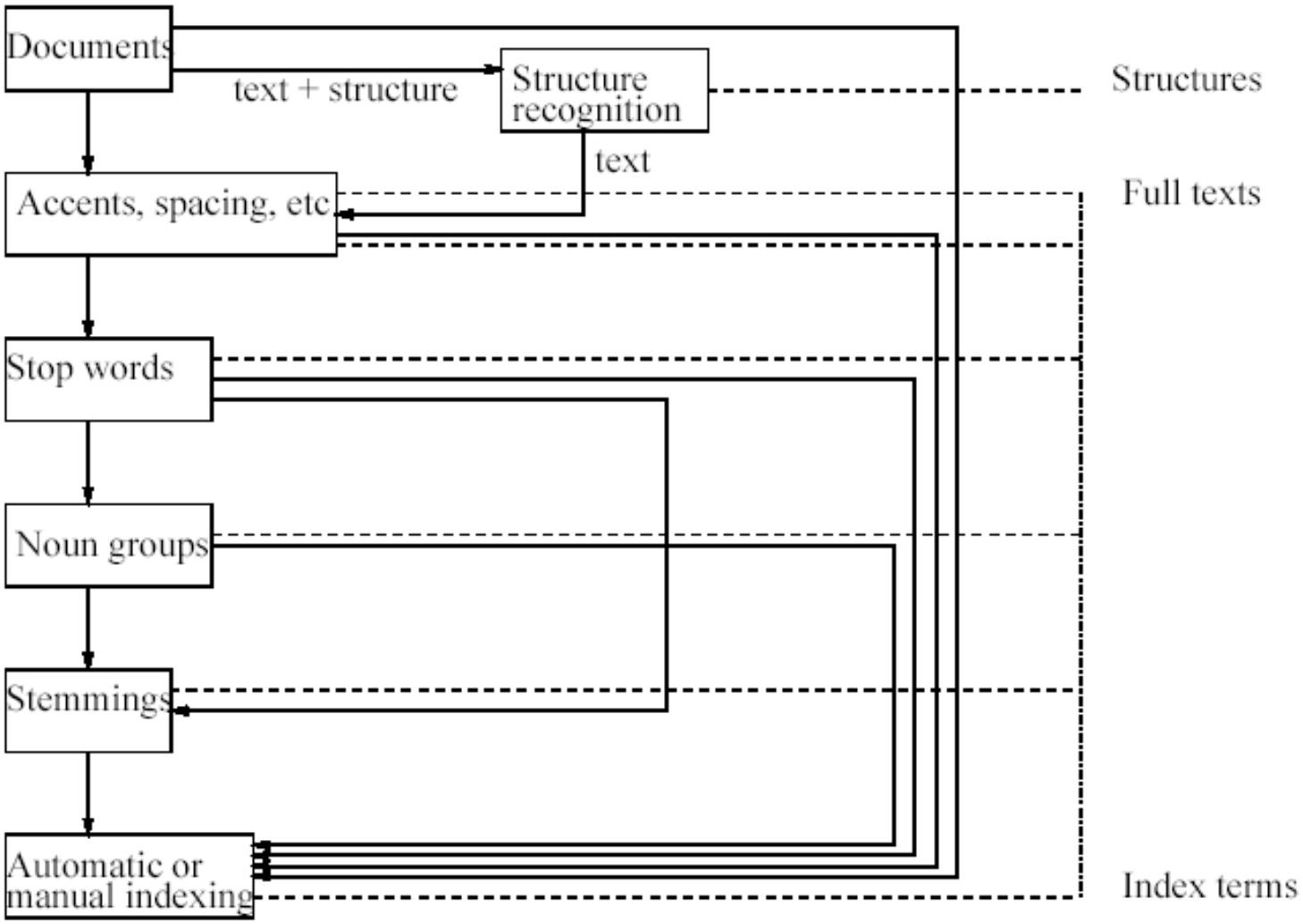
Ad Hoc Retrieval



Filtering


Profilo Utente 2
Docs Filtrati per l'Utente 2
Profilo Utente 1
Docs Filtrati per l'Utente 1
Document Stream

Uno sguardo alle attività di preprocessing



Passi di Pre-Elaborazione

- Eliminazione di caratteri o di simboli di annotazione indesiderati (e.g. etichette HTML tags, punteggiatura, …)
- Generazione delle sequenze di *token* basata su spazi.
- *Stemming* dei *tokens* in "radici"
  - *computational* → *comput*
- Eliminazione delle parole irrilevanti (*stopwords*), e.g. *un, il, esso/a*, …

Pre-Elaborazione (2)

- Rilevamento delle frasi comuni (e.g. terminologia di dominio mediante dizionari specifici).
- Costruzione di un indice inverso (*inverted index*):

  keyword → documenti che la contengono

Modello Booleano

- Un documento e' rappresentato tramite un insieme di *keyword (*parole chiave*)*.
- Le interrogazioni sono espressioni parentetiche booleane basate sulle keywords, connesse da operatori logici di tipo AND, OR e NOT:
  - [[Rio & Brazil] | [Hilo & Hawaii]] & hotel & !Hilton]
- Output: L'insieme dei documenti definiti rilevanti. Ogni decisione (appartenenzaa tale insieme) e' categoriale (e.g. binaria).

Modello Booleano

- Molto popolare (e.g. *Search Engines*):
  - Semantica facile da comprendere.
  - Formalismo semplice e chiaro.
- Una estensione che include l'ordinamento e'possibile (es. frequenza)
- Facile da implementare (indici inversi).

Limiti del modello booleano

- Scarsa flessibilita': AND≡tutti; OR≡qualsiasi.
- Poco espressivo, inadatto a interrogazioni complesse.
- Nessun controllo sul numero di documenti ritornati.
  - *Tutti* i documenti vengono restituiti.
- Nessun ordinamento:
  - *Tutti* i documenti "*soddisfano*" la interrogazione.
- E' difficile effettuare un *relevance feedback*.
  - La utilità o meno di un documento secondo un utente non ci dice come modificare la interrogazione corrispondentemente.

Modelli Statistici

- Un documento e' rappresentato come *bag of words:* (multi)insiemi di parole (con frequenze).
- *Bag* $\rightarrow$ occorrenze multiple.
- L'utente specifica i termini desiderati con un peso opzionale:
  - Termini pesati dell'interrogazione:

    Q = <*database* 0.5; *testo* 0.8; *informazione* 0.2>
  - Termini non pesati:

    Q = < *database*; *text*; *information* >
  - Nessuna condizione *booleana* nella *query*.

Retrieval Statistico

- Basato sulla *similarità* tra query e documento
- I documenti trovati sono ordinati in base alla similarità rispetto alla query.
- La similarità e' basata sulla *frequenza* di occorrenze delle keywords nella query e nel documento.
- Relevance feedback:
  - Aggiungi alla query i documenti piu' rilevanti.
  - Rimuovi gli irrelevanti dalla query.

Alcuni problemi aperti

- Come determinare le parole importanti in un documento?
  - Sensi vs. Parole
  - Sequenze (*n*-grammi di parole, espressioni idiomatiche,…) → termini
- Rilevanza dei termini in un documento vs. rilevanza dei termini nella intera collezione
- Similitudine tra interrogazioni e documenti
- Quali collezioni nel Web e qual'e' il ruolo della formattazione e del *linking*?

Il *Vector-Space Model*

- Definisci tutti i termini trovati nella base di documenti. Vocabolario (V).
- Assegna i termini ai vettori ortogonali dello spazio.

  Dimensioni = |V| = N
- Ogni termine *i-esimo* riceve un peso in un documento *j-esimo* (o query), $w_{ij.}$
- I documenti e le query sono vettori N-dimensionali:

$$d_j = (w_{1j}, w_{2j}, \ldots, w_{Nj})$$

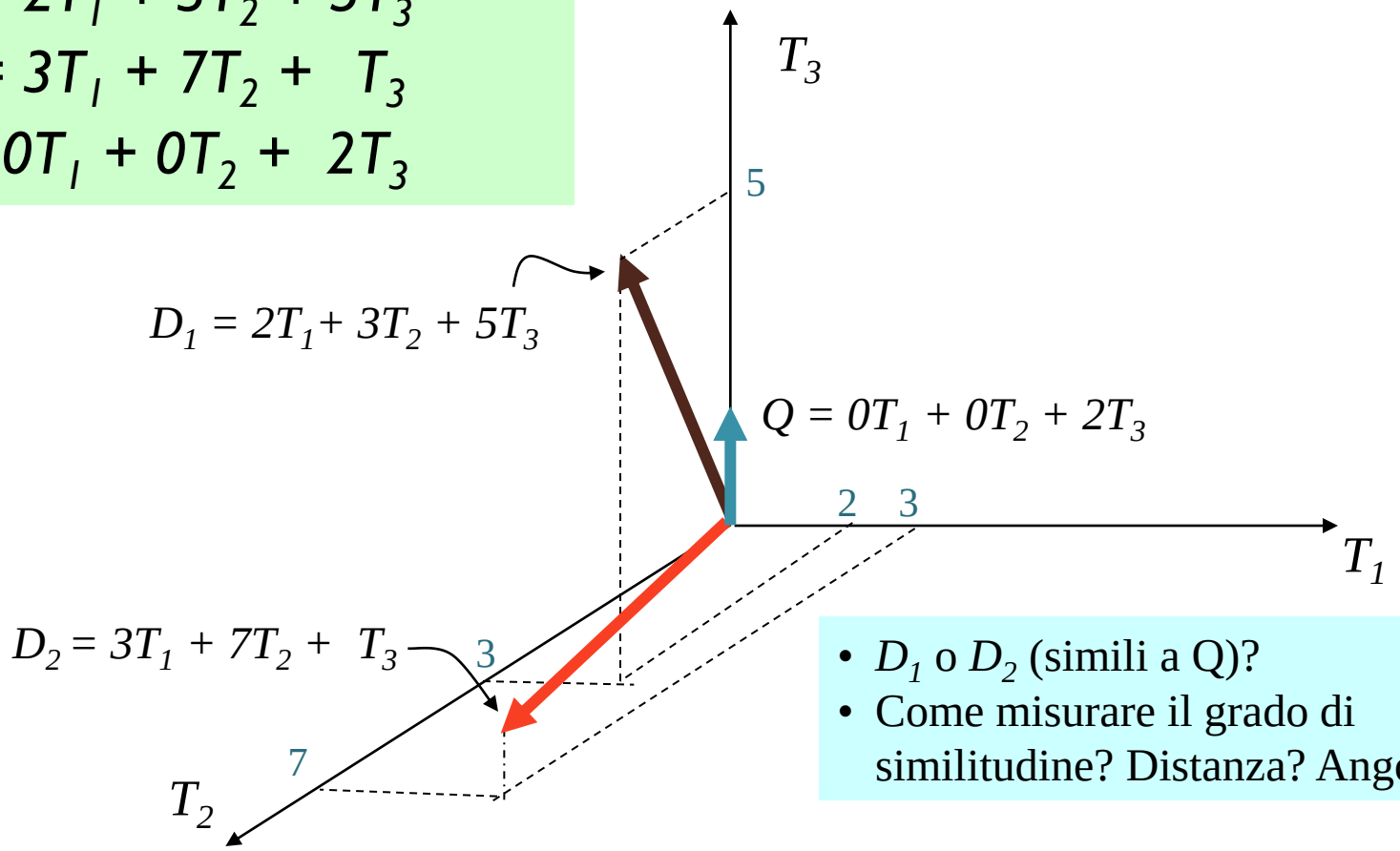
Interpretazione Geometrica

Esempio:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$



- D_1 o D_2 (simili a Q)?
- Come misurare il grado di similitudine? Distanza? Angolo?

La Collezione dei Documenti

- Una collezione di *n* documenti viene rappresentata nel VSM da una matrice termini×documenti.
- Un elemento w_{ij} di tale matrice esprime il **peso di un termine *j*-esimo nel documento *i*-esimo**; 0 ci dice che il termine non ha rilevanza o non occorre nel documento.

$$\begin{pmatrix} & T_1 & T_2 & .... & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{pmatrix}$$

Vector Space Model

- Un modello specifico di VSM si distingue da altri relativamente alle scelte di
  - Pesatura dei termini nelle interrogazioni
  - Pesatura dei termini nei documenti
  - Funzione di similarità (metrica)
  - Criterio di accettazione (i.e. soglia di rilevanza)

Pesatura: Frequenza

- Più frequenti i termini sono in un documento più importanti essi tendono ad essere, i.e. più significativi per il contenuto.

  f_{ij} = frequenza del termine *i* nel documento *j*

- Per normalizzare la frequenza (*tf*), *term frequency*, attraverso il corpus (o il documento):

  $tf_{ij} = f_{ij} / max_k\{f_{kj}\}$

Pesatura: *Inverse Document Frequency*

- I termini che appaiono in molti documenti *differenti* sono *meno indicativi* del contenuto (e.g. *a, da, per, io, questo, ha, ho, è…*):

  df_i = *document frequency* del termine *i*
  = numero di docs che contengono il termine *i*

  idf_i = inverse document frequency del termine *i*,
  = log_2 (M/ df_i)
  (M: numero totale dei documenti)

Inverse Document Frequency

- L'inverse document frequency (idf_i) e' un indice del potere di discriminazione di un termine
  - Se $df_i = M$ allora segue che $idf_i = 0$

- Il logaritmo bilancia l'effetto della frequenza tf_i.

Pesatura TF-IDF

- Un indicatore che cattura le precedenti proprietà e' il fattore *tf-idf*:

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij}\, log_2\,(M/df_i)$$

- Un termine che occorre frequentemente in un documento ma raramente nell'intera collezione riceve un peso più alto localmente a quel documento.

TF-IDF

- Esistono molte varianti che forniscono alternative funzioni di pesatura
- Sperimentalmente, *tf-idf* ha dimostrato ottime prestazioni (è un ottimo esempio di elevata adeguatezza empirica).

TF-IDF – Un esempio

Un documento ha i seguenti termini con le seguenti frequenze:

A(3), B(2), C(1)

Assumiamo una collezione di 10,000 docs in cui le frequenze globali dei termini sono:

A(50), B(1300), C(250)

Ne segue:

A: *tf* = 3/3; *idf* = log(10000/50) = 5.3; *tf-idf*=5.3

B: *tf* = 2/3; *idf* = log(10000/1300) = 2.0; *tf-idf*=1.3

C: *tf* = 1/3; *idf* = log(10000/250) = 3.7; *tf-idf*=1.2

Il vettore della domanda

- Il vettore di rappresentazione di una domanda e' tipicamente trattato come un documento e pesato tramite il fattore *tf-idf*.
- Alternativa: fornire pesi assegnati dagli utenti ai termini della query

Metrica di Similarita'

- Una metrica di similarita' e' una funzione che calcola il grado di similitudine tra due vettori.

- Grazie all'uso di una metrica di similarita' tra la query ed ogni documento e' possibile:
  - Ordinare i documenti dal piu' simile/vicino (cioe' il piu' rilevante) al meno simile.
  - Impostare una soglia al di sopra della quale respingere i documenti (per es. Per controllare la numerosita' dei risultati).

Similarità e Distanza

- Ogni funzione di distanza *d* può essere assunta come una metrica *sim* di similarità
- *sim* e *d* sono ovviamente in una relazione di proporzionalità inversa
- Un metodo per ottenere una funzione *sim* data una distanza è la seguente
  - *sim* varia tra 0 (completamente diversi) ed 1 (identici) allora vale:
  - $d(x,y) = \alpha(1-sim(x,y))^n$
  - con
    - $n, \alpha > 0,$
- Quindi: $sim(x,y) = 1-\beta d(x,y)^{-n}$

Misure di similarita' – Prodotto Interno

- La similarita' tra il documento $\boldsymbol{d}_j$ e la query $\boldsymbol{q}$ puo' essere calcolato secondo il *prodotto scalare*:

$$\text{sim}(\boldsymbol{d}_j, \boldsymbol{q}) = \boldsymbol{d}_j \bullet \boldsymbol{q} = \sum_{i=1}^{t} w_{ij} \cdot w_{iq}$$

dove w_{ij} e' il peso del termine i nel documento j e w_{iq} e' il peso del termine i nella query

- Nel caso binario, il prodotto e' la somma dei termini comuni tra query e documento (cardinalità della intersezione).
- Nel caso pesato, e' la somma dei prodotti dei pesi dei soli termini in comune.

Prodotto Interno: Proprieta'

- Il prodotto scalare non e' limitato superiormente.
- I documenti lunghi con molti termini (che non si ripetono) sono favoriti.
- Non e' sensibile alla *precisione* del *matching*:

  misura quanti termini coincidono ma NON quanti termini differiscono.

Prodotto Interno -- Esempi

Binario:

	retrieval	database	architecture	computer	text	management	information
D =	1,	1,	1,	0,	1,	1,	0
Q =	1,	0 ,	1,	0,	0,	1,	1

Taglia vettore = taglia vocabolario = 7
0 => termine non trovato nel documento (o query)

sim(D, Q) = 3

Pesato:

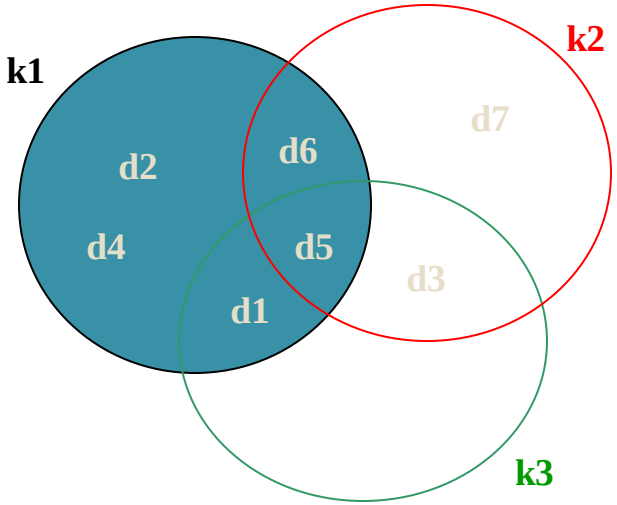
$D_1 = 2T_1 + 3T_2 + 5T_3 \qquad D_2 = 3T_1 + 7T_2 + 1T_3$

$Q = 0T_1 + 0T_2 + 2T_3$

$\text{sim}(D_1 , Q) = 2 \cdot 0 + 3 \cdot 0 + 5 \cdot 2 = 10$

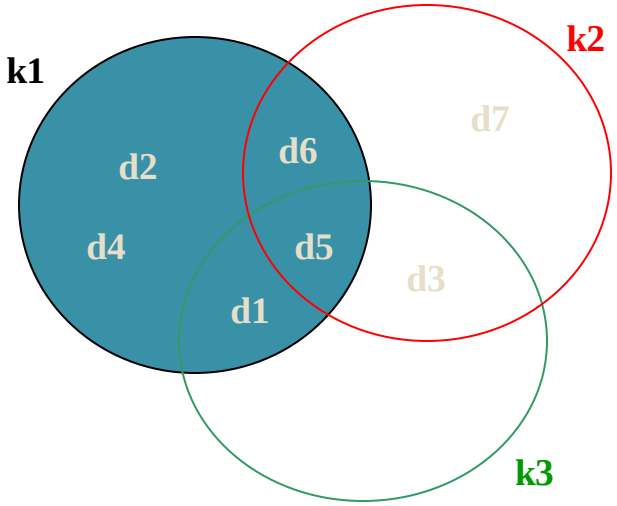
$\text{sim}(D_2 , Q) = 3 \cdot 0 + 7 \cdot 0 + 1 \cdot 2 = 2$

The Vector Model: Example I



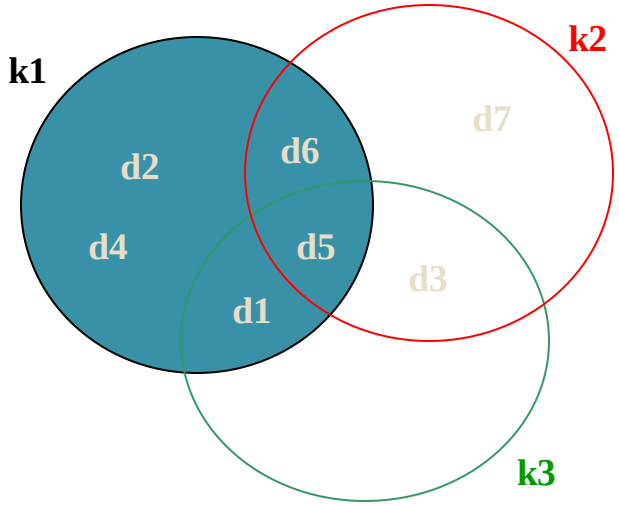
	k1	k2	k3	q • dj
d1	1	0	1	2
d2	1	0	0	1
d3	0	1	1	2
d4	1	0	0	1
d5	1	1	1	3
d6	1	1	0	2
d7	0	1	0	1
q	1	1	1	

The Vector Model: Example II



	k1	k2	k3	q • dj
d1	1	0	1	4
d2	1	0	0	1
d3	0	1	1	5
d4	1	0	0	1
d5	1	1	1	6
d6	1	1	0	3
d7	0	1	0	2
q	1	2	3	

The Vector Model: Example III

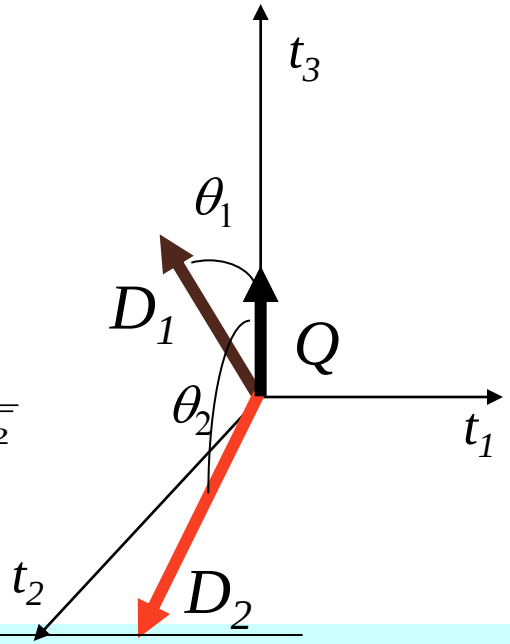


	k1	k2	k3	q • dj
d1	2	0	1	5
d2	1	0	0	1
d3	0	1	3	11
d4	2	0	0	2
d5	1	2	4	17
d6	1	2	0	5
d7	0	5	0	10
q	1	2	3	

Cosine Similarity

- Misura il coseno dell'angolo tra due vettori.
- Prodotto interno se normalizzato rispetto alla norma dei vettori.

$$\text{CosSim}(\boldsymbol{d}_j, \boldsymbol{q}) = \frac{\vec{d} \times \vec{q}}{|\vec{d}| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$



$D_1 = 2T_1 + 3T_2 + 5T_3$ $\quad \text{CosSim}(D_1, Q) = 10 / \sqrt{(4+9+25)(0+0+4)} = 0.81$

$D_2 = 3T_1 + 7T_2 + 1T_3$ $\quad \text{CosSim}(D_2, Q) = 2 / \sqrt{(9+49+1)(0+0+4)} = 0.13$

$Q = 0T_1 + 0T_2 + 2T_3$

D_1 e' 6 volte migliore di D_2 secondo la cosine similarity ma solo 5 volte usando il prodotto interno.

Altre Misure

Similarity Measure	Binary Int.	Weighted Int.
Inner Product	$\lvert X \cap Y \rvert$	$\sum_i x_i \cdot y_i$
Dice Coeffficent	$\frac{2 \cdot \lvert X \cap Y \rvert}{\lvert X \rvert^2 + \lvert Y \rvert^2}$	$\frac{2 \cdot \sum_i x_i \cdot y_i}{\sum_i (x_i^2 + y_i^2)}$
Cosine Measure	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$	$\frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i (x_i^2 + y_i^2)}}$
Jaccard Coefficent	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert - \lvert X \cap Y \rvert}$	$\frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i (x_i^2 + y_i^2 - x_i \cdot y_i)}}$

Implementazione *Naïve*

- Converti tutti i documenti della collezione *D* in vettori pesati tramite *tf-idf* $\boldsymbol{d}_j$, per le keyword nel vocabulario *V*.
- Converti la interrogazione (*query*) in un vettore $\boldsymbol{q}$ pesato tramite *tf-idf*.
- Per ogni $\boldsymbol{d}_j \in D$:

  Calcola $s_j = cosSim(\boldsymbol{d}_j, \boldsymbol{q})$
- Ordina i documenti $\boldsymbol{d}_j$ in ordine decrescente secondo s_j e mostra i primi documenti ottenuti all'utente

Time complexity: $O(|V| \cdot |D|)$

$|V| = 10{,}000; |D| = 100{,}000; |V| \cdot |D| = 1{,}000{,}000{,}000$ (!)

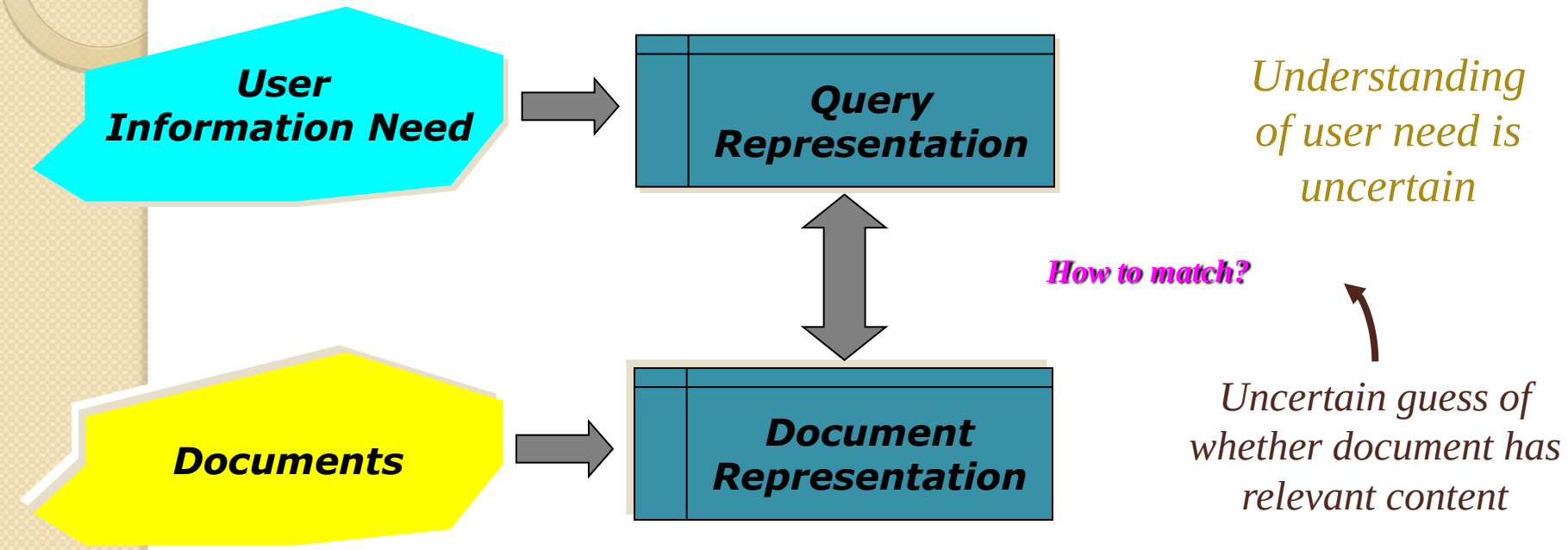
Vector Space Model

- Approccio semplice ma formalmente ben definito.
- Tiene in considerazione sia le frequenze lessicali locali (*tf*) che quelle global (*idf*).
- Supporta matching parziali e risultati ordinati.
- Sebbene alcune assunzioni siano molto deboli, la accuratezza empirica e' sorprendentemente buona.
- Consente una implementazione efficiente anche per collezioni di documenti enormi.

Limiti del *Vector Space Model*

- Manca supporto per informazione di tipo semantico (e.g. *word senses*).
- Non e' sensibile alla informazione sintattica (e.g. strutture sintagmatiche, ordinamento delle parole, informazione di prossimita').
- Viene assunta la *indipendenza tra i termini* (cfr. sinonimia).
- Manca il controllo fornito da un modello Booleano (e.g., il fatto che un termine DEVE apparire nel documento).
  - Data la query "*a b*", e' preferibile non sono distinguibili documenti in cui *a* e' frequente mentre *b* e' assente da quelli in cui sia *a* che *b* occorrono ma in modo raro.

Why probabilities in IR?



In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.

Probabilities provide a principled foundation for uncertain reasoning. Can we use probabilities to quantify our uncertainties?

The Probability Ranking Principle

"If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

The Probability Ranking Principle

"If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of **decreasing probability of relevance to the user** who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

The Probability Ranking Principle

"If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of **decreasing probability of relevance to the user** who submitted the request, where the **probabilities are estimated as accurately as possible** on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

The Probability Ranking Principle

"If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of **decreasing probability of relevance to the user** who submitted the request, where the **probabilities are estimated as accurately as possible** on the basis of whatever data have been made available to the system for this purpose, **the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data**."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

The document ranking problem

- We have a collection of documents
- User issues a query
- A list of documents needs to be returned
- **Ranking method is core of an IR system:**
  - **In what order do we present documents to the user?**
  - We want the "best" document to be first, second best second, etc….
- **Idea: Rank by probability of relevance of the document w.r.t. information need**
  - P(relevant|document$_i$, query)

Probabilistic Retrieval Strategy

- Estimate how terms contribute to relevance
  - How do things like tf, df, and length influence your judgments about document relevance?
    - One answer is the Okapi formulae (S. Robertson)
- Combine to find document relevance probability
- Order documents by decreasing probability

Probability Ranking Principle

- How do we compute all those probabilities?
  - Do not know exact probabilities, have to use estimates
  - Binary Independence Retrieval (BIR) is the simplest model
- Questionable assumptions
  - "Relevance" of each document is independent of relevance of other documents.
    - Really, it's bad to keep on returning **duplicates**
  - Boolean model of relevance
  - That one has a single step information need
    - Seeing a range of results might let user refine query

Probabilistic Ranking

Basic concept:

"For a given query, if we know some documents that are relevant, terms that occur in those documents should be given greater weighting in searching for other relevant documents.

By making assumptions about the distribution of terms and applying Bayes Theorem, it is possible to derive weights theoretically."

Van Rijsbergen

Binary Independence Model

- Traditionally used in conjunction with PRP
- **"Binary" = Boolean**: documents are represented as binary incidence vectors of terms (cf. lecture 1):

  $\vec{x} = (x_1, \ldots, x_n)$

  $x_i = 1$ iff term *i* is present in document *x*.
- **"Independence":** terms occur in documents independently
- Different documents can be modeled as same vector

- Bernoulli Naive Bayes model (cf. text categorization!)

Binary Independence Model

- Queries: binary term incidence vectors
- Given query ***q***,
  - for each document ***d*** need to compute ***p(R|q,d)***.
  - replace with computing ***p(R|q,x)*** where ***x*** is a binary term incidence vector representing ***d***
  - Interested only in ranking
- Will use odds and Bayes' Rule:

$$O(R \mid q, \vec{x}) = \frac{p(R \mid q, \vec{x})}{p(NR \mid q, \vec{x})} = \frac{\dfrac{p(R \mid q)\, p(\vec{x} \mid R, q)}{p(\vec{x} \mid q)}}{\dfrac{p(NR \mid q)\, p(\vec{x} \mid NR, q)}{p(\vec{x} \mid q)}}$$

Probabilistic Retrieval (1)

Principio (minimizzazione del rischio):
Ritorna un documento se il costo atteso di trovarlo (o rigettarlo) utilmente e' inferiore al costo atteso di trovarlo (o rigettarlo) erroneamente

$$EC_{retr} < EC_{\overline{retr}}$$

dove

$$EC_{retr} = P(rel|d)C_{retr,rel} + P(\overline{rel}|d)C_{retr,\overline{rel}}$$

Probabilistic Retrieval (2)

Formalmente il precedente principio si esprime:

$$P(rel|d)C_{retr,rel} + P(\overline{rel}|d)C_{retr,\overline{rel}} < P(rel|d)C_{\overline{retr},rel} + P(\overline{rel}|d)C_{\overline{retr},\overline{rel}}$$

cioe'

$$\frac{P(rel|d)}{P(\overline{rel}|d)}C_{retr,rel} + C_{retr,\overline{rel}} < \frac{P(rel|d)}{P(\overline{rel}|d)}C_{\overline{retr},rel} + C_{\overline{retr},\overline{rel}}$$

e quindi

$$\frac{P(rel|d)}{P(\overline{rel}|d)} < \frac{C_{\overline{retr},\overline{rel}} - C_{retr,\overline{rel}}}{C_{retr,rel} - C_{\overline{retr},rel}} = \alpha \text{ (i.e. a constant)}$$

Probabilistic Retrieval (3)

Osserva che:

$$P(rel|d) = P(d|rel)P(rel)$$

cosi' che:

$$\frac{P(rel|d)}{P(\overline{rel}|d)} = \frac{P(d|rel)}{P(d|\overline{rel})} \cdot \frac{P(rel)}{P(\overline{rel})}$$

In ogni caso $\frac{P(rel)}{P(\overline{rel})}$ e' costante per ogni documento e interrogazione, cosi' che la funzione di ordinamento puo' essere espressa come:

$$\frac{P(d|rel)}{P(d|\overline{rel})}$$

Mutua Indipendenza dei termini

Un documento *d* e' descritto da un insieme di *features* (termini)

$$d = (d_1, ..., d_t) \qquad \text{with } d_i \in D_i$$

Se le feature sono tutte mutuamente indipendenti (i.e. $P(D_i|rel,D_j)=P(D_i|rel)$) la funzione di ranking diviene:

$$\frac{P(d|rel)}{P(d|\overline{rel})} = \prod_{i=1}^{t} \frac{P(D_i=d_i|rel)}{P(D_i=d_i|\overline{rel})}$$

Features Binarie

- Siano d_i i valori binari di D_i allora

$$P(d_i|rel) = p_i^{d_i}(1-p_i)^{1-d_i}$$

$p_i = P(d_i = 1|rel)$ and $1 - p_i = P(d_i = 0|rel)$

e

$$P(d_i|\overline{rel}) = q_i^{d_i}(1-q_i)^{1-d_i}$$

con

$q_i = P(d_i = 1|\overline{rel})$ and $1 - q_i = P(d_i = 0|\overline{rel})$

Estimation (2)

- La funzione di ordinamento probabilistica (nel caso di *features* binarie) diviene:

$$\prod_{i=1}^{t} \frac{p_i^{d_i}(1-p_i)^{(1-d_i)}}{q_i^{d_i}(1-q_i)^{(1-d_i)}}$$

- Passando ai logaritmi la rilevanza del generico *j*-esimo termine e' modellabile come:

$$tr_j = log\frac{p_j(1-q_j)}{q_j(1-p_j)}$$

Stima dei parametri

- *Naive estimation*
  - Calcola p_i and q_i a partire dai primi documenti
- Stimatori Statici:
  - $q_j = P(d_j = 1 \mid \overline{rel}) = df_j / N$ (nessun doc che contiene tr_j è rilevante)
  - $p_j = P(d_j = 1 \mid rel) = 0.5$ (massima incertezza)

Stimatori (2)

- Stimatori Dinamici
  - Sia r_j il numero stimato dei documenti rilevanti che contengono t_j. (OSS: $df_j \geq r_j$)
  - $[p(D_i = 1 \mid rel) =]$ $p_j = r_j / |rel|$
  - $[p(D_i = 1 \mid \overline{rel}) =]$ $q_j = (df_j - r_j) / (N - |rel|)$
  - Un termine e' perfettamente rilevante se: $r_j = df_j$
  - … debolmente rilevante (i.e. occorre in modo bilanciato in *rel* e $\overline{rel}$):

$$r_j = df_j \cdot (|rel| / N)$$

  - caso generale

$$r_j = \alpha \cdot df_j + \beta$$

Relevance Feedback

- Processo:
  - Genera una *ranked list* attraverso un motore di IR qualsiasi (o uno stimatore di base)
  - (Passo 0) Determina il ranking dei documenti trovati attraverso una soglia *r*, uno stimatore probabilistico (p_j e q_j) ed il modello $\sum_j tr_j$
  - (Passo 1) Stima di nuovo p_j e q_j
  - (Iterazione) Ripeti i Passi 0 e 1 a convergenza

Relevance Feedback

- Stima iniziale ($k=0$)
  - $p^0_i=0.5$ (tutti i t_i equivalenti)
  - $q^0_i=df_i/N$ (tutti i doc di t_i irrilevanti)
- Passo intermedio (k-esimo)
  - $p^k_i=V^k_i/V^k$
  - $q^k_i=(df_i-V^k_i)/(N-V^k)$
  - con
    - V^k_i = #doc ritrovati al passo k che contengono t_i
    - V^k = #doc ritrovati al passo k
    - df_i = #doc che contengono t_i – doc frequency
    - N = #doc nella collezione

Relevance Feedback (2)

- Stima iniziale ($k=0$)
  - $p^0_i=0.5$
  - $q^0_i=df_i/N$
- Passo intermedio (con aggiustamenti statici)
  - $p^k_i=(V^k_i+0.5) / (V^k+1)$
  - $q^k_i=(df_i-V^k_i+0.5)/(N-V^k+1)$
- Passo intermedio (con aggiustamenti lessicali, df_i/N)
  - $p^k_i=(V^k_i+ df_i/N) / (V^k+1)$
  - $q^k_i=(df_i-V^k_i+ df_i/N)/(N-V^k+1)$

Confronto tra modelli classici

- Il modello booleano non sostiene matching parziali e' sembra il modello piu' debole
- Salton and Buckley in una serie di esperimenti dimostrano che il VSM e' su collezioni generali superiore ai modelli probabilstici
- Il vector space model (e le sue varianti) sembra(no) il modello empiricamente piu' accurato

Good and Bad News

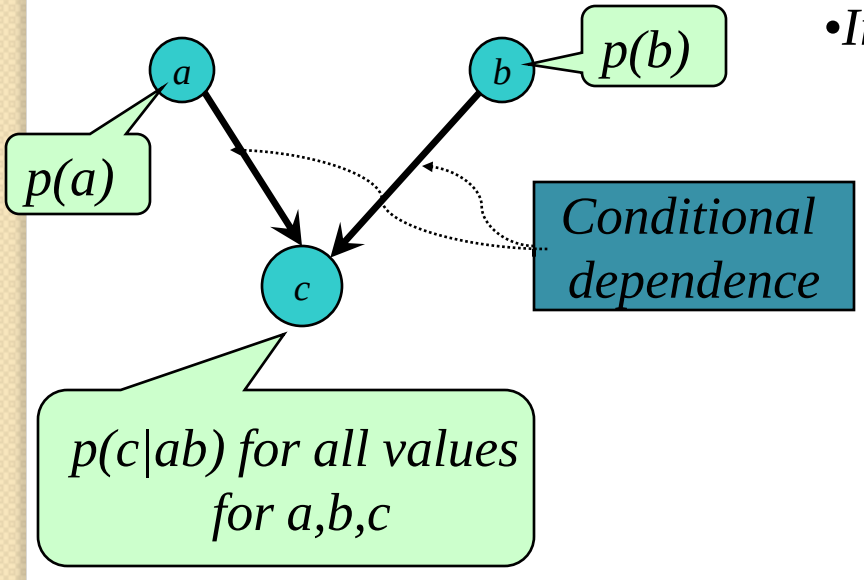
- Standard Vector Space Model
  - Empirical for the most part; success measured by results
  - Few properties provable
- Probabilistic Model Advantages
  - Based on a firm theoretical foundation
  - Theoretically justified optimal ranking scheme
- Disadvantages
  - Making the initial guess to get V
  - Binary word-in-doc weights (not using term frequencies)
  - Independence of terms (can be alleviated)
  - Amount of computation
  - Has never worked convincingly better in practice

Bayesian Networks for Text Retrieval (Turtle and Croft 1990)

- Standard probabilistic model assumes you can't estimate P(R|D,Q)
  - Instead assume independence and use P(D|R)
- But maybe you can with a Bayesian network*
- What is a Bayesian network?
  - A directed acyclic graph
  - Nodes
    - Events or Variables
      - Assume values.
      - For our purposes, all Boolean
  - Links
    - model direct dependencies between nodes

Bayesian Networks

a,b,c - propositions (events).

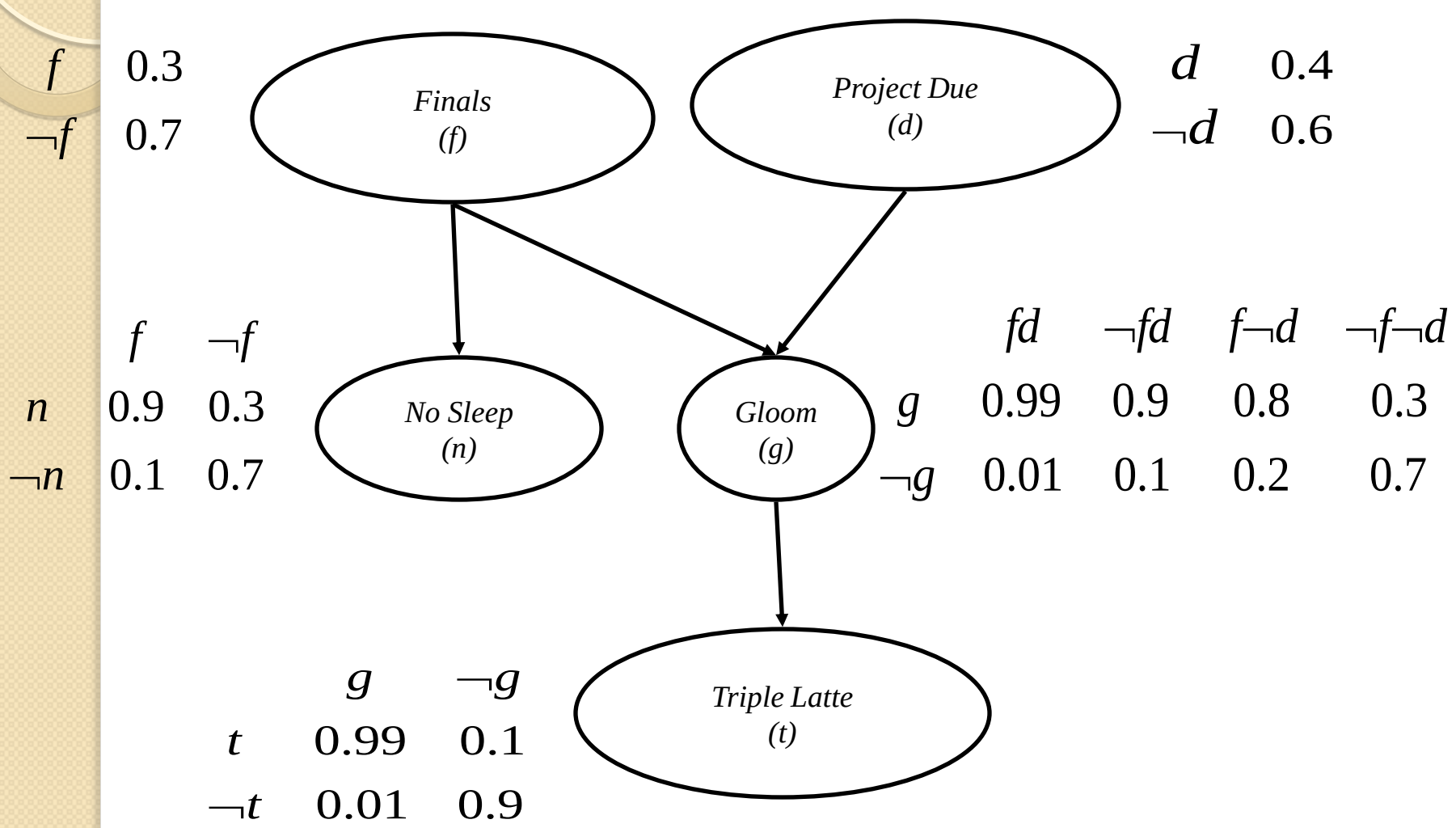


- *Bayesian networks model causal relations between events*

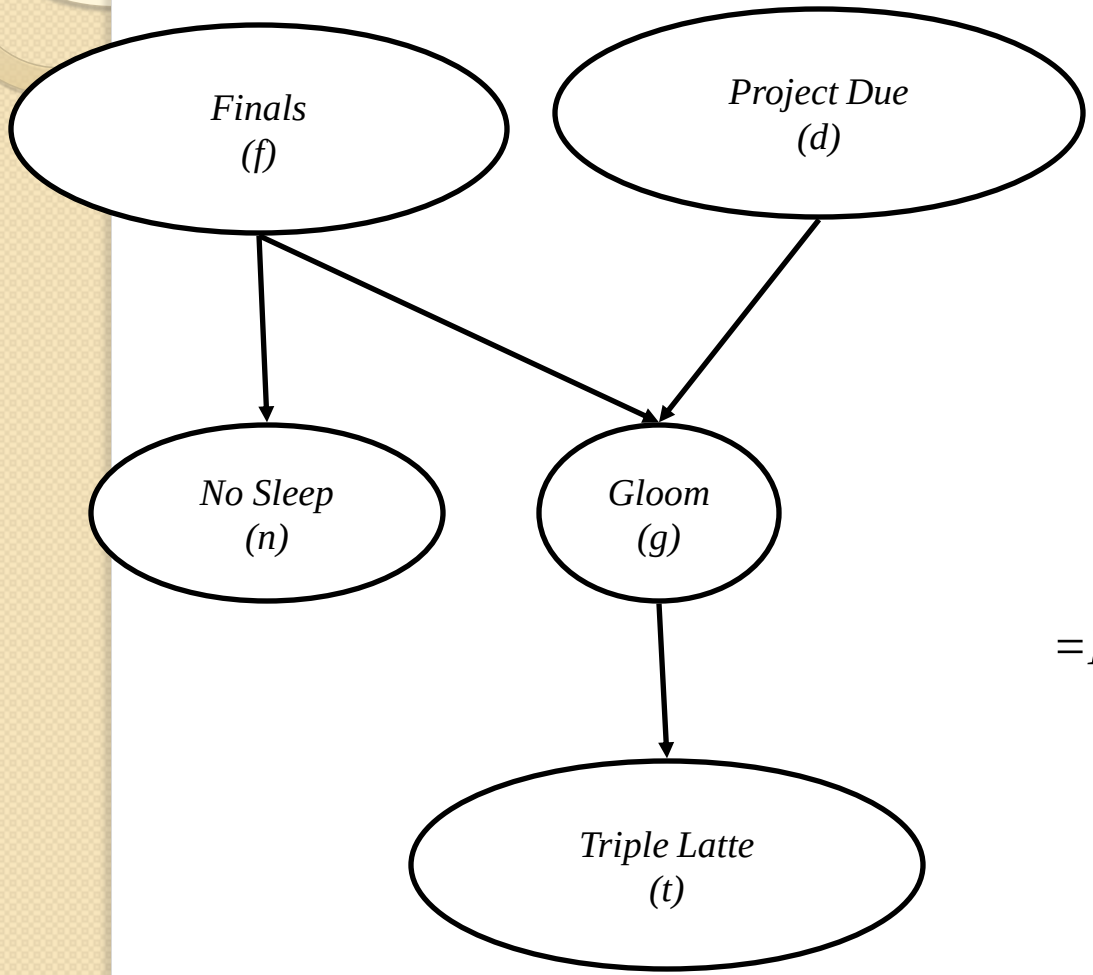
- *Inference in Bayesian Nets:*
  - *Given probability distributions for roots and conditional probabilities can compute apriori probability of any instance*
  - *Fixing assumptions (e.g., b was observed) will cause recomputation of probabilities*

For more information see:
R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. 1999. ***Probabilistic Networks and Expert Systems****. Springer Verlag.*
J. Pearl. 1988. ***Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.*** *Morgan-Kaufman.*

Toy Example


f 0.3
¬f 0.7
Finals
(f)
Project Due
(d)
d 0.4
¬d 0.6
f ¬f
n 0.9 0.3
¬n 0.1 0.7
No Sleep
(n)
Gloom
(g)
fd ¬fd f¬d ¬f¬d
g 0.99 0.9 0.8 0.3
¬g 0.01 0.1 0.2 0.7
g ¬g
t 0.99 0.1
¬t 0.01 0.9
Triple Latte
(t)

Independence Assumptions



- *Independence assumption:*
  $P(t|g, f)=P(t|g)$
- *Joint probability*

$$P(f\ d\ n\ g\ t)$$
$$=P(f)\ P(d)\ P(n|f)\ P(g|f\ d)\ P(t|g)$$

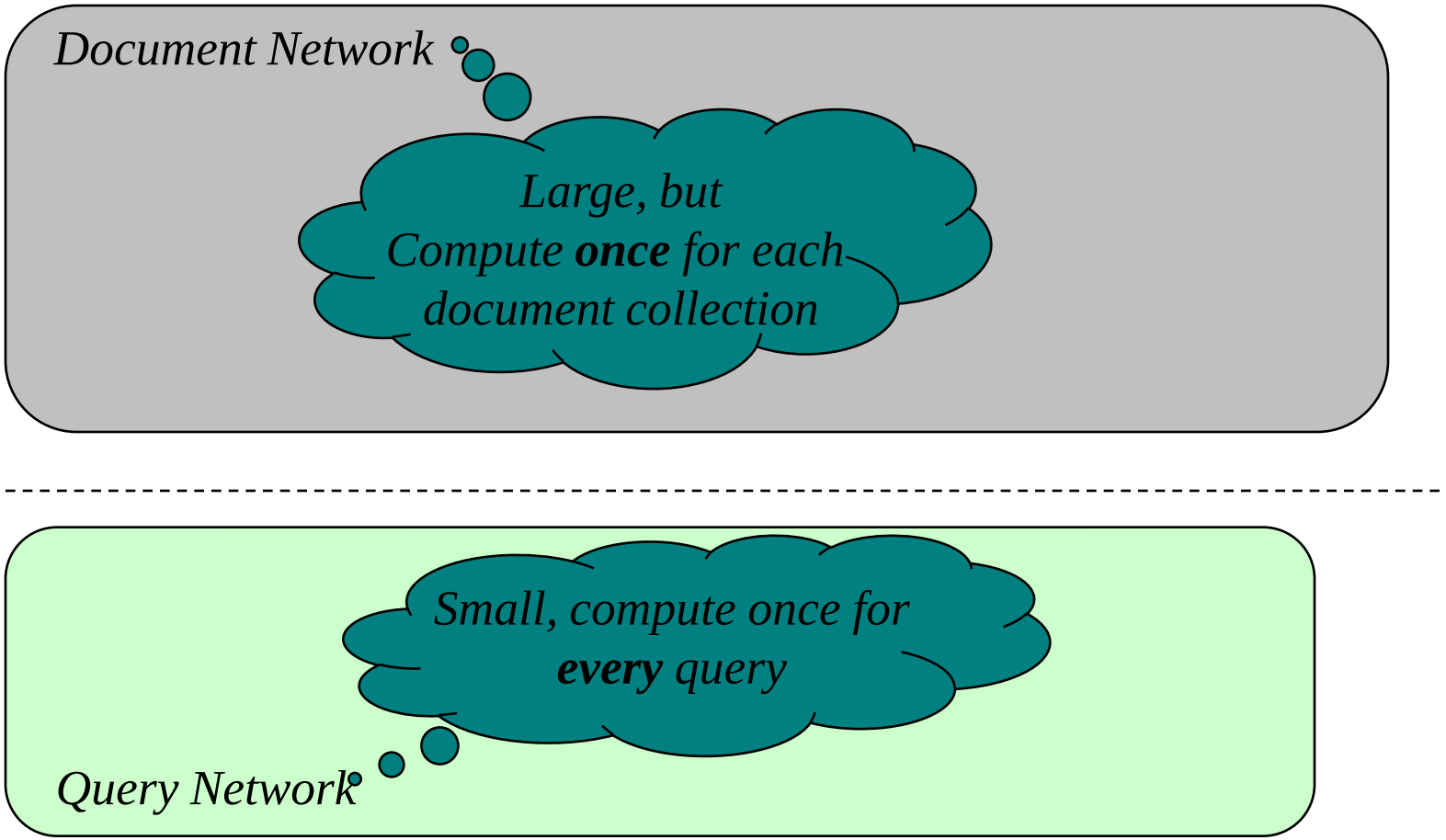
Chained inference

- Evidence - a node takes on some value
- Inference
  - Compute *belief* (probabilities) of other nodes
    - conditioned on the known evidence
  - Two kinds of inference: *Diagnostic* and *Predictive*
- Computational complexity
  - General network: NP-hard
    - Tree-like networks are easily tractable
    - Much other work on efficient exact and approximate Bayesian network inference
      - Clever dynamic programming
      - Approximate inference ("loopy belief propagation")

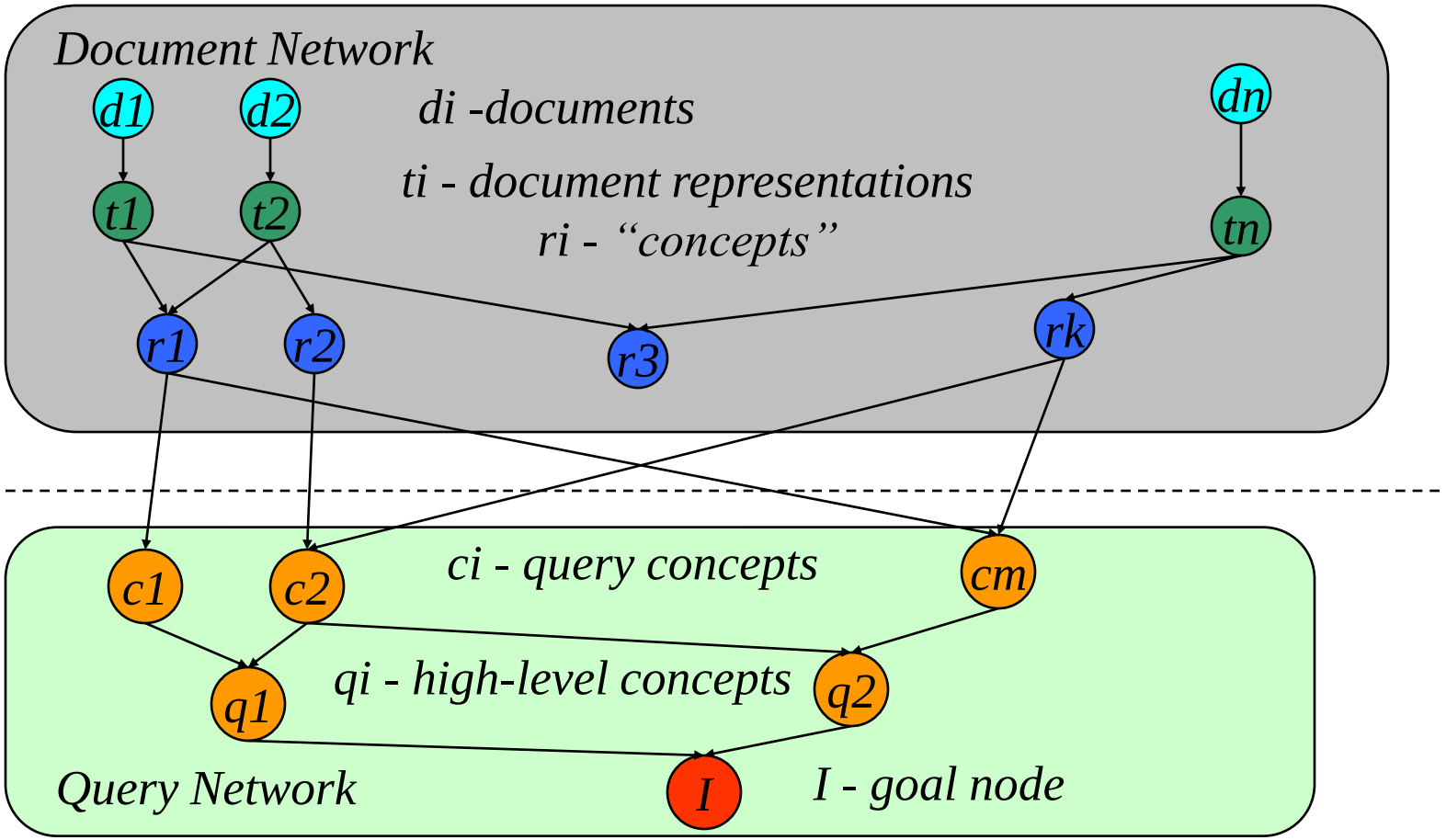
Model for Text Retrieval

- Goal
  - Given a user's information need (evidence), find probability a doc satisfies need
- Retrieval model
  - Model docs in a *document network*
  - Model information need in a *query network*

Bayesian Nets for IR: Idea



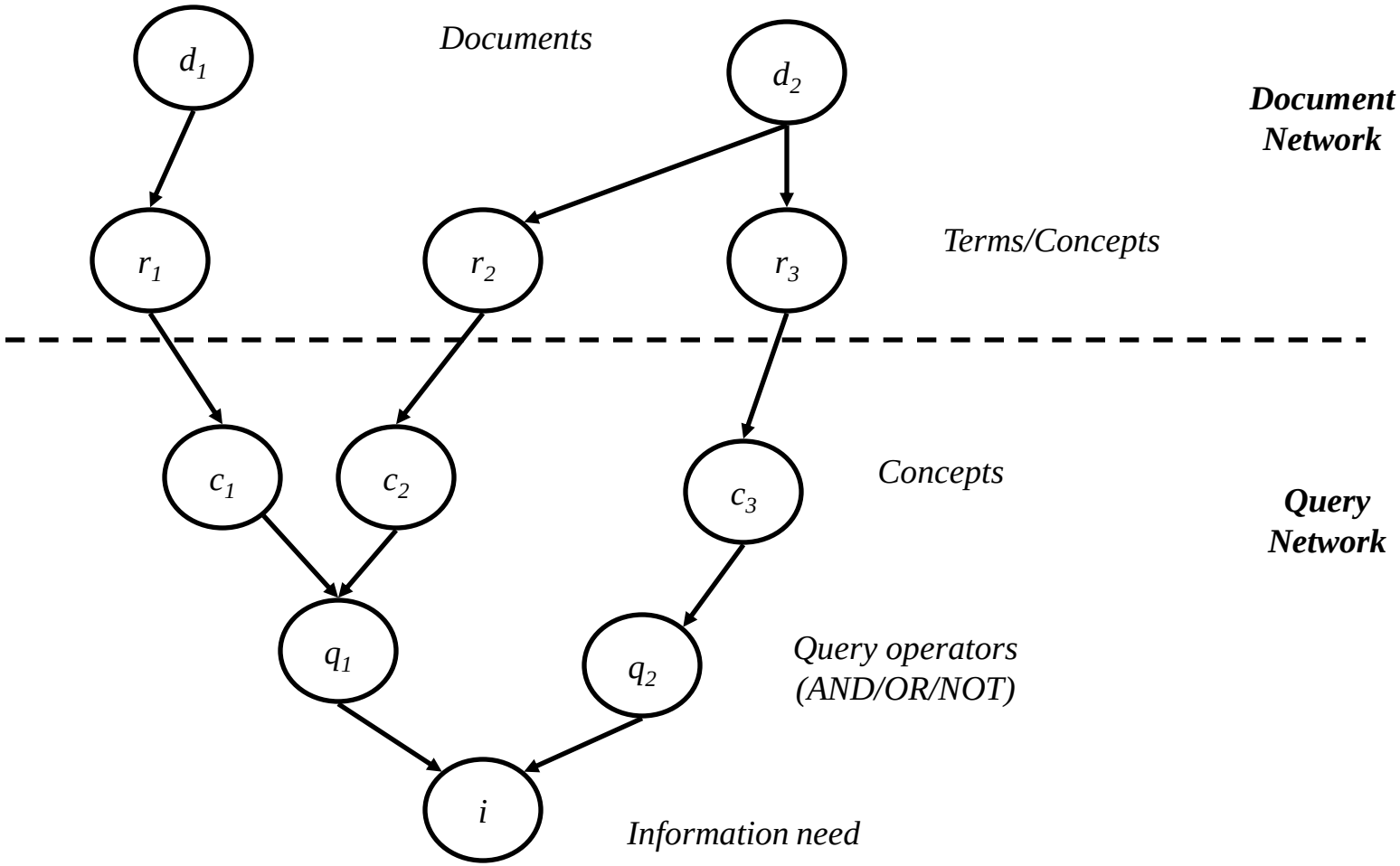
Bayesian Nets for IR: Idea



Bayesian Nets for IR

- Construct Document Network (once !)
- For each query
  - Construct best Query Network
  - Attach it to Document Network
  - Find subset of $\mathbf{d_i}$**'s** which maximizes the probability value of node **I** (best subset).
  - Retrieve these $\mathbf{d_i}$**'s** as the answer to query.

Bayesian nets for text retrieval

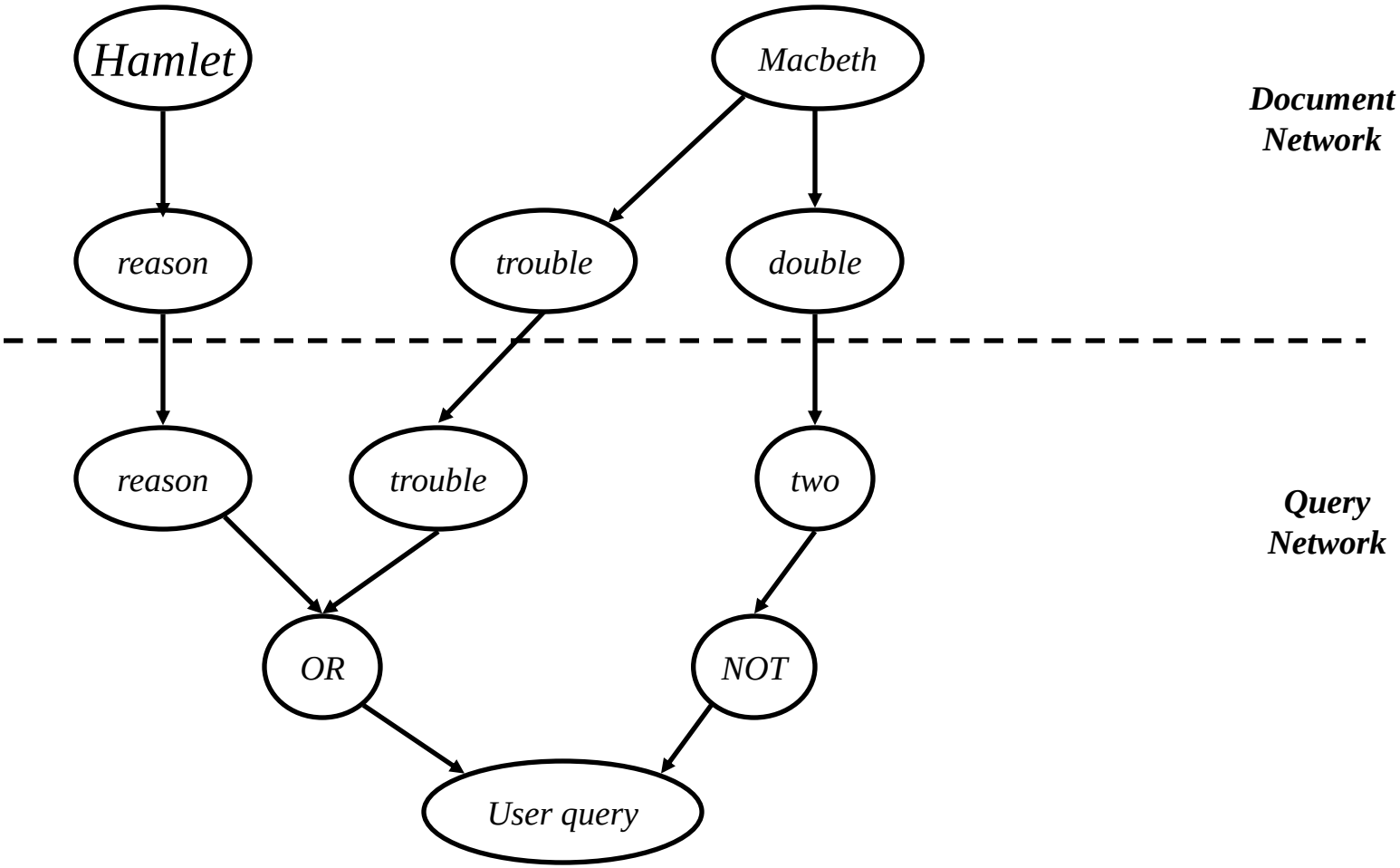


Link matrices and probabilities

- Prior doc probability $P(d) = 1/n$
- $P(r|d)$
  - within-document term frequency
  - $tf \times idf$ - based
- $P(c|r)$
  - 1-to-1
  - thesaurus
- $P(q|c)$: canonical forms of query operators
  - Always use things like AND and NOT – never store a full CPT*

*conditional probability table

Example: "*reason trouble –two*"



Extensions

- Prior probs don't have to be $1/n$.
- "User information need" doesn't have to be a query - can be words typed, in docs read, any combination …
- Phrases, inter-document links
- Link matrices can be modified over time.
  - User feedback.
  - The promise of "personalization"

Computational details

- Document network built at indexing time
- Query network built/scored at query time
- Representation:
  - Link matrices from docs to any single term are like the postings entry for that term
  - Canonical link matrices are efficient to store and compute
- Attach evidence only at roots of network
  - Can do single pass from roots to leaves

Bayes Nets in IR

- Flexible ways of combining term weights, which can generalize previous approaches
  - Boolean model
  - Binary independence model
  - Probabilistic models with weaker assumptions

- Efficient large-scale implementation
  - InQuery text retrieval system from U Mass
    - Turtle and Croft (1990) [Commercial version defunct?]

- Need approximations to avoid intractable inference
- Need to estimate all the probabilities by some means (whether more or less ad hoc)

- Much new Bayes net technology yet to be applied?

Resources

S. E. Robertson and K. Spärck Jones. 1976. Relevance Weighting of Search Terms. *Journal of the American Society for Information Sciences* 27(3): 129–146.

C. J. van Rijsbergen. 1979. *Information Retrieval.* 2nd ed. London: Butterworths, chapter 6. [Most details of math] http://www.dcs.gla.ac.uk/Keith/Preface.html

N. Fuhr. 1992. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3),243–255. [Easiest read, with BNs]

F. Crestani, M. Lalmas, C. J. van Rijsbergen, and I. Campbell. 1998. Is This Document Relevant? ... Probably: A Survey of Probabilistic Models in Information Retrieval. *ACM Computing Surveys* 30(4): 528–552.

http://www.acm.org/pubs/citations/journals/surveys/1998-30-4/p528-crestani/

[Adds very little material that isn't in van Rijsbergen or Fuhr]

Resources

H.R. Turtle and W.B. Croft. 1990. Inference Networks for Document Retrieval. *Proc. ACM SIGIR*: 1-24.

E. Charniak. Bayesian nets without tears. *AI Magazine* 12(4): 50-63 (1991). http://www.aaai.org/Library/Magazine/Vol12/12-04/vol12-04.html

D. Heckerman. 1995. A Tutorial on Learning with Bayesian Networks. Microsoft Technical Report MSR-TR-95-06 http://www.research.microsoft.com/~heckerman/

N. Fuhr. 2000. Probabilistic Datalog: Implementing Logical Information Retrieval for Advanced Applications. *Journal of the American Society for Information Science* 51(2): 95–110.

R. K. Belew. 2001. *Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge UP 2001.

MIR 2.5.4, 2.8