Sistemi Informativi LS
AA. 2009/2010

SKIP-AND-PRUNE: Cosine-based Top-K Query Processing for Efficient Context-Sensitive Document Retrieval

Jong Wook Kim K. Selçuk Candan

Gruppo 14:

Daniele Biagi - Veronica Conti - Fabio Ferretti

Obiettivi in teoria...

Modello dei dati basato su parole chiave

Contesto di ricerca dell'utente

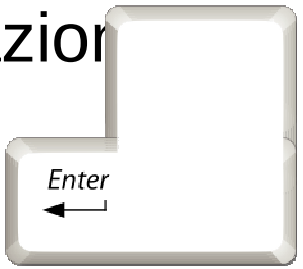
Algoritmo in grado di utilizzare questi modelli per restituire all'utente i migliori documenti che soddisfano i parametri di ricerca

...e in pratica

Facciamo un esempio…

Mettiamoci nei panni di un appassionato di musica che voglia ricercare maggiori informazion

VIOLA



Obiettivo dell'algoritmo

- Utilizzare abilmente tecniche top-k per **reinterpretare** a seconda del contesto utente, solo i documenti necessari e limitare l'esplorazione di grandi porzioni del database

- Uso di una scoring function basata sul coseno quindi non monotona, per comparare i documenti del db con la query dell'utente

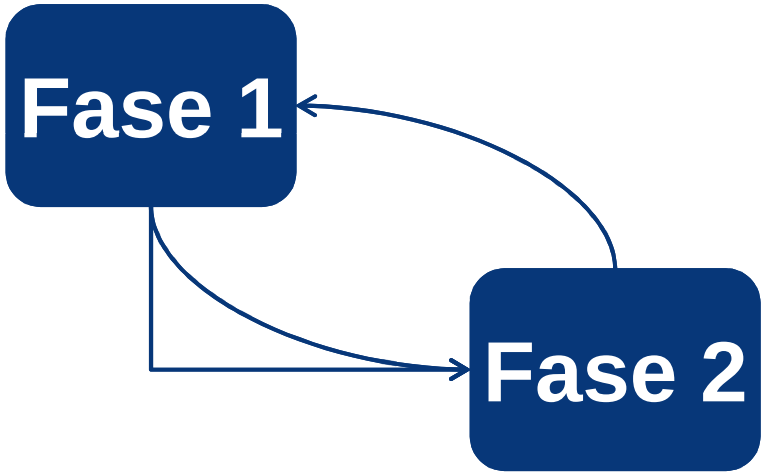
$$sim_{cos}(\vec{d},\vec{q}) = cos(\vec{d},\vec{q}) = \frac{\vec{d}\cdot\vec{q}}{\|\vec{d}\|\ \|\vec{q}\|}$$

necessità di un nuovo algoritmo di query processing

$$sim'_{cos}(\vec{d},\vec{q},U) = \cos(U\vec{d},U\vec{q})$$

Skip-and-Prune (SnP)

- L'algoritmo opera in due fasi:
  - FASE 1: selezione dei documenti secondo lo specifico contesto
  - FASE 2: scelta dei migliori k documenti in base alla query



- La soluzione al problema consiste nel trovare i k documenti in DB, con il più alto punteggio di similarità del coseno rispetto alla query considerando lo spazio specifico dei concetti dell'utente

Formalizzazione del problema

- $\mathcal{L}$→ dizionario contenente ℓ distinte parole chiave

ID_KEYWORD	KEYWORD
l1	“viola”
l2	“spartito”
l3	“inno”
l4	“cantare”
l5	“calcio”

Formalizzazione del problema

- $\mathcal{DB}$ → collezione di m documenti, rappresentati con la matrice D = m x l

D	l1	l2	l3	l4	l5
concerto	0,30	0,40	0,10	0,10	0,10
flauto_magico_mozart	0,20	0,38	0,09	0,21	0,00
musica_nello_sport	0,10	0,11	0,30	0,20	0,20
accordare_viola	0,81	0,13	0,06	0,00	0,00
fiorentina_firenze	0,29	0,00	0,08	0,00	0,40

- D quindi rappresenta quanto sia realmente presente una parola chiave in un certo documento

Formalizzazione del problema

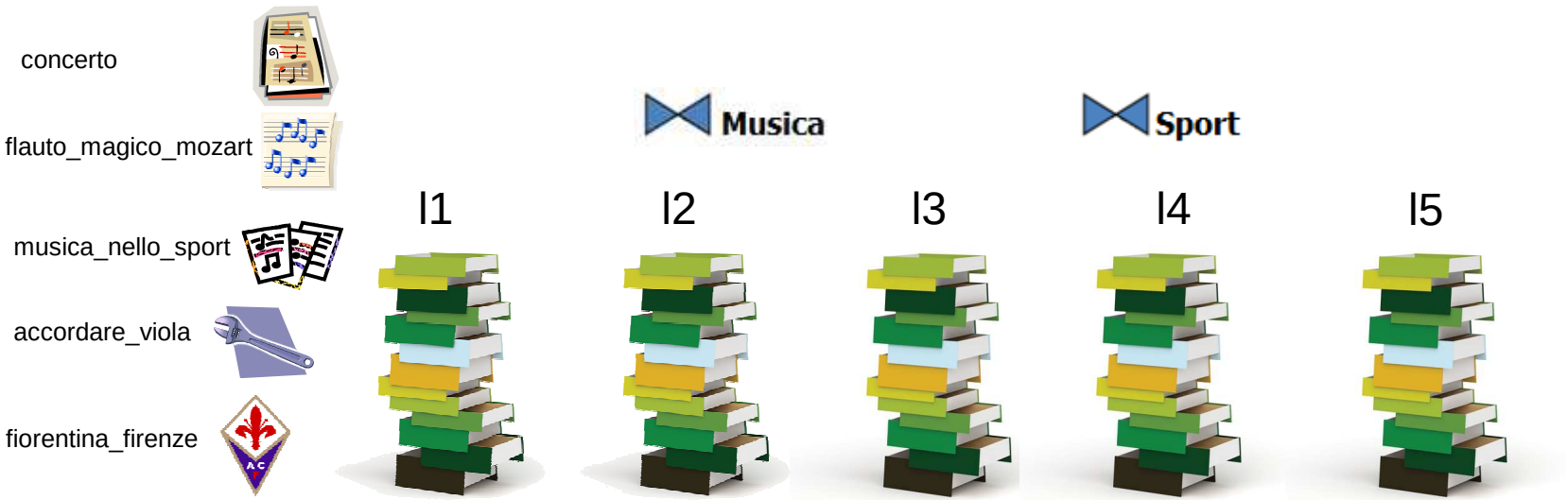
- $\mathcal{U}$ → matrice del contesto dell'utente che riassume quanto la parola chiave sia importante all'interno di un determinato concetto per lo specifico utente

U	l1	l2	l3	l4	l5
musica	0,28	0,30	0,12	0,30	0,00
sport	0,10	0,00	0,20	0,00	0,70

- Il risultato del prodotto $D_u = DU^T$ è la matrice rappresentante quanto un dato documento sia importante all'interno di un concetto in un contesto preciso
- Molto **oneroso**, le dimensioni delle matrici rendono incalcolabile
- Qui entrano in gioco le due fasi dell'algoritmo!

Skip-and-Prune: Fase 1

- L'obiettivo è cercare di estrarre per ogni concetto i documenti più rilevanti nel contesto dell'utente, senza considerare ancora la query!
- Esecuzione degli operatori di Ranked Join su tanti stream di documenti ordinati quante sono le parole chiave rilevanti per il concetto a cui sono associati



Skip-and-Prune: Fase 1

U	l1	l2	l3	l4	l5
musica	0,28	0,30	0,12	0,30	0,00
sport	0,10	0,00	0,20	0,00	0,70

D	l1	l2	l3	L4	l5
concerto	0,30	0,40	0,10	0,10	0,10
flauto_magico_mozart	0,20	0,38	0,09	0,21	0,00
musica_nello_sport	0,10	0,11	0,30	0,20	0,20
accordare_viola	0,81	0,13	0,06	0,00	0,00
fiorentina firenze	0,29	0,00	0,08	0,00	0,40

Fase 2

accordare_viola	0,276
concerto	0,247
flauto_magico_mozart	0,244
musica_nello_sport	0,156

τ=0,274

Musica

l1

documento	score
accordare_viola	0,230
concerto	0,085
fiorentina_firenze	0,082
flauto_magico_mozart	0,057

l2

documento	score
concerto	0,120
flauto_magico_mozart	0,114
accordare_viola	0,039
musica_nello_sport	0,033

l3

documento	score
musica nello sport	0,035
concerto	0,012
flauto_magico_mozart	0,010
fiorentina_firenze	0,009

l4

documento	score
flauto_magico_mozart	0,063
musica_nello_sport	0,060
concerto	0,030
accordare_viola	0,000

Skip-and-Prune: Fase 1

U	l1	l2	l3
musica	0,28	0,30	0,12
sport	0,10	0,00	0,20

ATTENZIONE:
non abbiamo ancora considerato la query, stiamo solo inviando alla fase 2 i documenti più rilevanti di ogni concetto

IDEA:
lo score di un documento è la sua coordinata nella dimensione di quel concetto

flauto_magico_mozart	0,244
musica_nello_sport	0,156

l2

documento	score
concerto	0,120
flauto_magico_mozart	0,114
accordare_viola	0,039
musica_nello_sport	0,033

l3

documento	score
musica nello sport	0,035
concerto	0,012
flauto_magico_mozart	0,010
fiorentina_firenze	0,009

l4

documento	score
flauto_magico_mozart	0,063
musica_nello_sport	0,060
concerto	0,030
accordare_viola	0,000

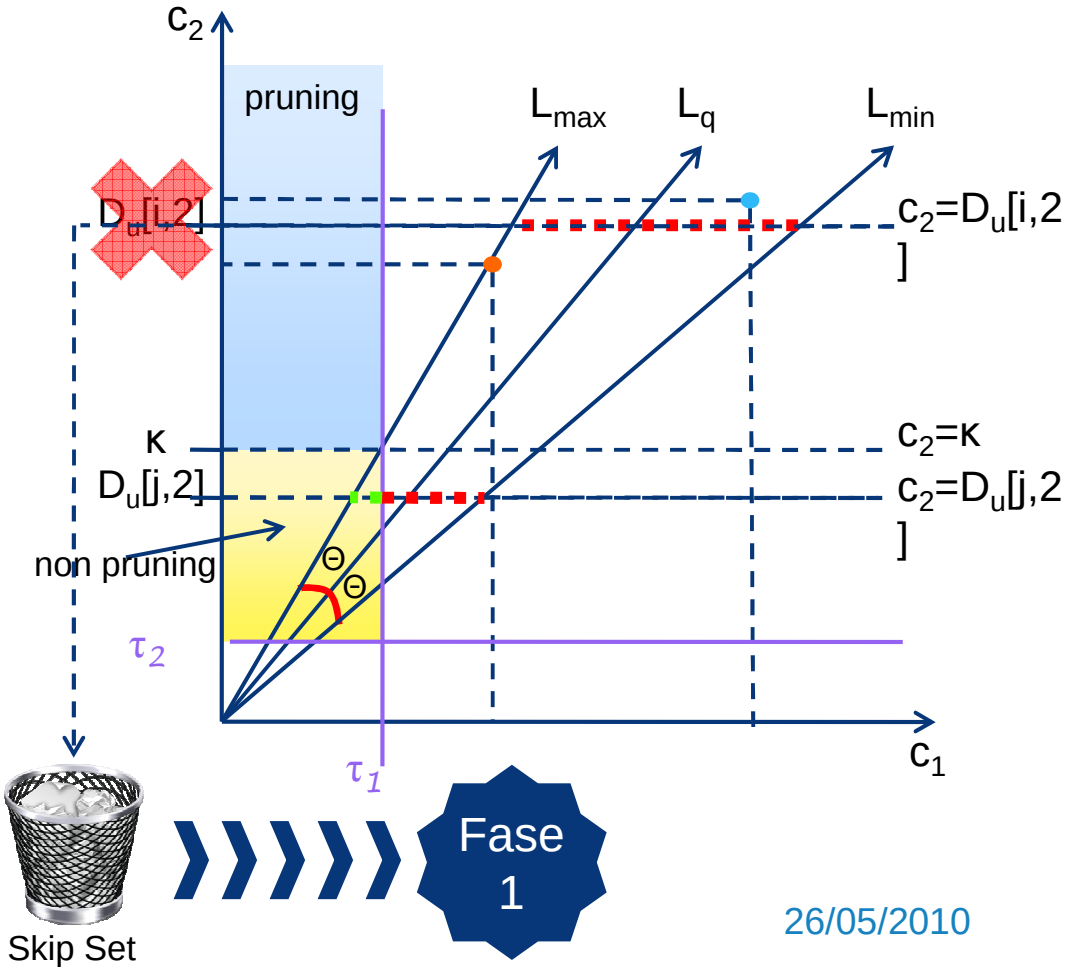
	score
	0,230
	0,085
	082
	7

Skip-and-Prune: Fase 1

- L'operatore di Ranked Join restituisce i documenti che ottengono un punteggio oltre una certa soglia denominata τ
- La soglia tiene conto sia di D che di U consentendo di avanzare ai documenti con score di rilevanza per quel concetto maggiore di essa. Si aggiorna ad ogni Ranked Join
- I documenti in fondo alle InvertedLists sono meno importanti per quel concetto e, a meno che il loro ID sia all'interno di un insieme detto SkipSet che viene riempito dai documenti eliminati nella fase 2 (retroazione!), supereranno più tardi la soglia per accedere alla fase successiva

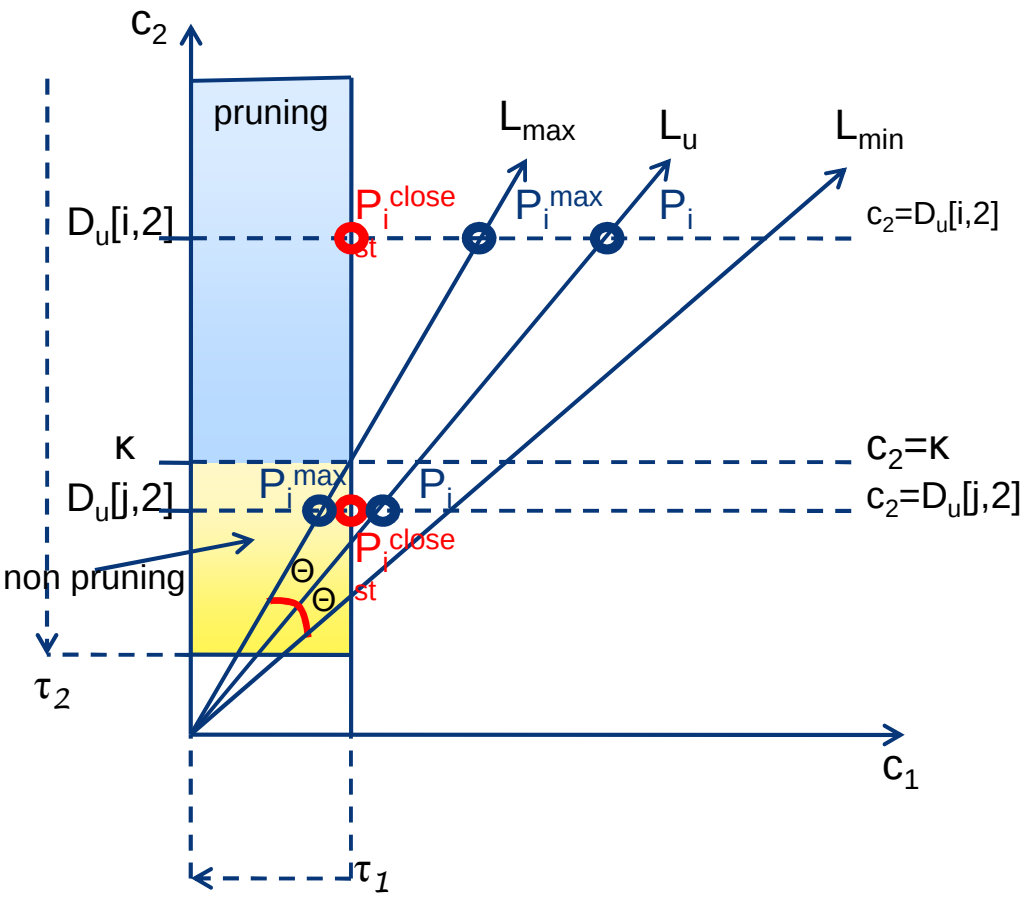
Skip-and-Prune: Fase 2

- Spazio bidimensionale dei concetti
- La query dell'utente considerata nel suo specifico contesto è rappresentata con il vettore L_q. Obiettivo di questa fase è scegliere documenti vicini alla query nel piano
- Ogni documento uscente dalla fase 1 ha associato il punteggio maggiore alla soglia, rappresentante la coordinata nell'asse del concetto a cui è associato il Ranked Join. Se il documento ha superato entrambi i Ranked Join è un punto nel piano
- Supponiamo di considerare una query top-2; ricevuti i primi due documenti posso determinare i vettori L_{max} ed L_{min} che definiscono il triangolo dei risultati accettabili
- Consideriamo un nuovo documento d_i che supera il Ranked Join di c_2: ha una sola coordinata nello spazio e c1 è incognito. d_i è accettabile solamente se è posizionato sulla linea tratteggiata
- Sia K l'ordinata del punto dove L_{max}Interseca τ_1; d_i può essere eliminato dato che la fase 1 limita le sua ascisse a valori inferiori a τ_1: non potrò mai posizionarsi nella zona tratteggiata! Il suo id viene inserito nello SkipSet ed il Ranked Join di c_1 non lo considererà



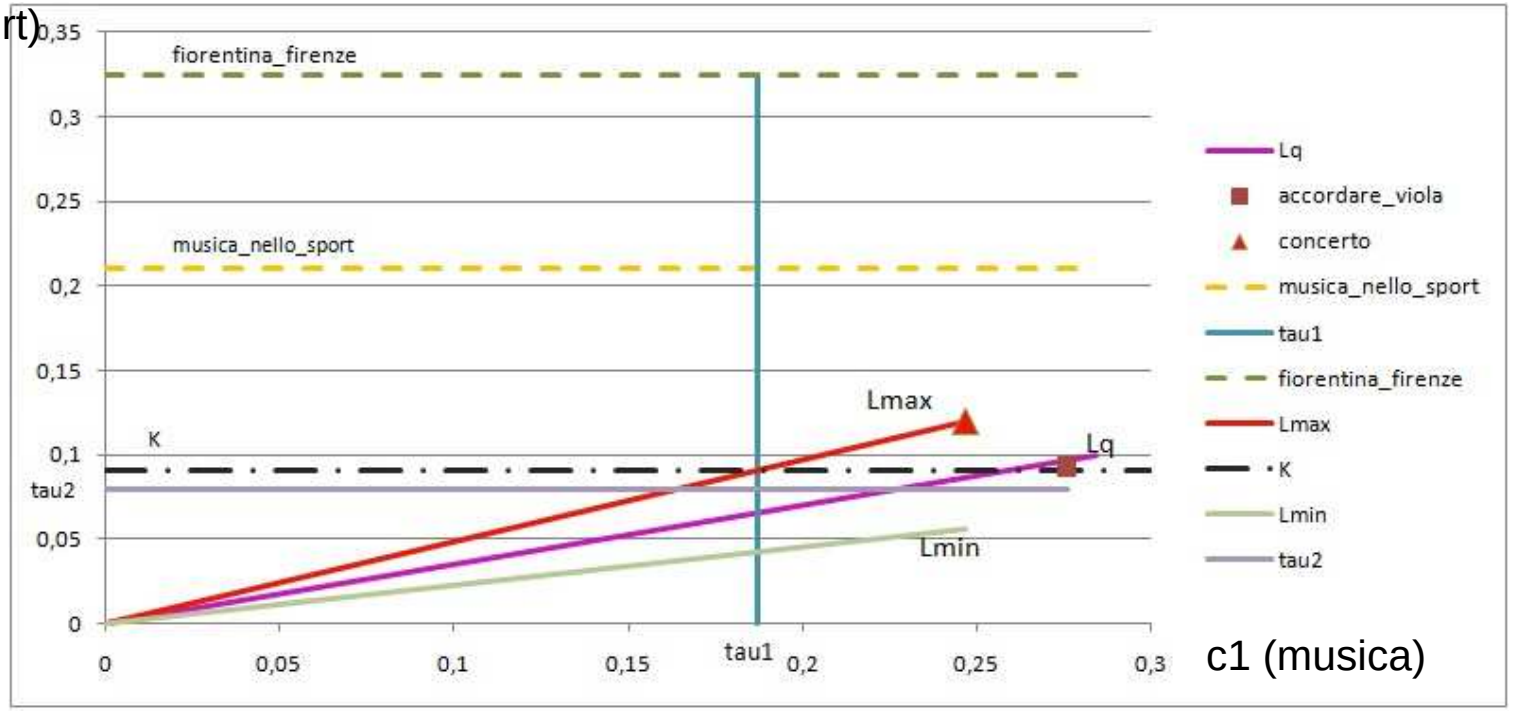
Skip-and-Prune: Fase 2
Criteri di Pruning

- P_i rappresenta la query in corrispondenza della retta del documento H_i, tracciata grazie all'ordinata ottenuta dalla fase 1
- $P_i^{closest}$ rappresenta l'ascissa massima che $D_u[i,2]$ potrà avere su H_i
- P_i^{max} è il punto limite accettabile per far si che d_i sia tra i correnti top-k candidati
- d_i verrà eliminato se:
  $\Delta(P_i^{closest},P_i) \geq \Delta(P_i^{max},P_i)$
- Altrimenti il suo score sarà maggiore di *min_score*, il più basso tra quelli dei top-k documenti
- *min_score*consente di fare le considerazioni geometriche: la regione accettabile è definita dal triangolo tramite l'equazione:
  cos(Θ) = *min_score*



Che fine ha fatto l'esempio??

c2 (sport)

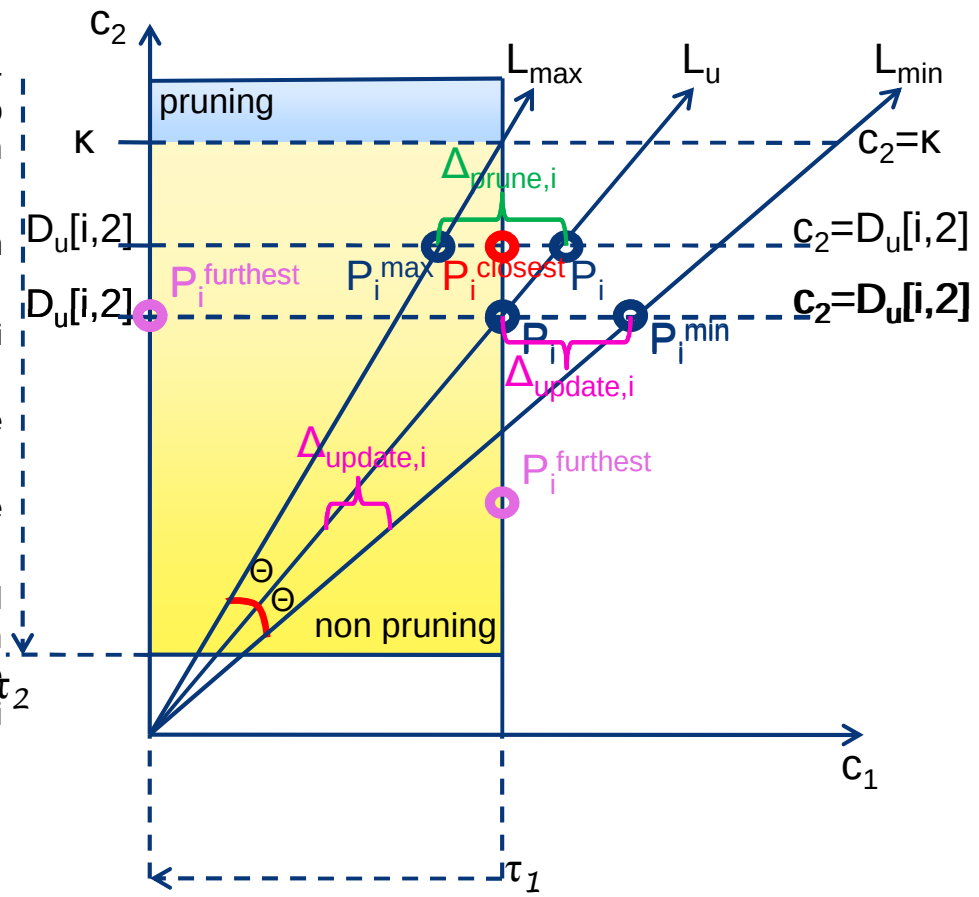


c1 (musica)

- Sui documenti "fiorentina_firenze" e "musica_nello_sport" viene fatto pruning→ il loro ID viene inserito nello SkipSet
- Come atteso, i documenti **migliori** sono "concerto" e "accordare_viola"

Skip-and-Prune: ulteriori raffinamenti...

- Il raffinamento proposto consiste nel ridefinire *min_score* di modo che possa riflettere i punteggi dei documenti visti solo parzialmente
- Prendiamo il documento d_k nella lista corrente dei top-k col minimo punteggio. Questo potrà essere rimosso e rimpiazzato con un documento d_i se quest'ultimo non avrà mai un punteggio inferiore
- **Come esserne certi???** Consideriamo un ragionamento analogo al precedente, però invertito:
  - sia $P_i^{furthest}$ il punto in H_i più lontano da P_i all'interno di τ_1
  - P_i^{min}è il punto dove H_ii nterseca L_{min} e $\Delta(P_i^{min},P_i) = \Delta_{update,i}$
  - d_i non potrà mai avere punteggi inferiori a d_k se $\Delta(P_i^{furthest},P_i) < \Delta_{update,i}$
- Questo autorizza a rimuovere d_ke sostituirlo con d_i, il cui punteggio totale verrà calcolato grazie ai random access. Il valore di *min_score* verrà ricalcolato in base al nuovo punteggio più basso presente nella lista dei top-k



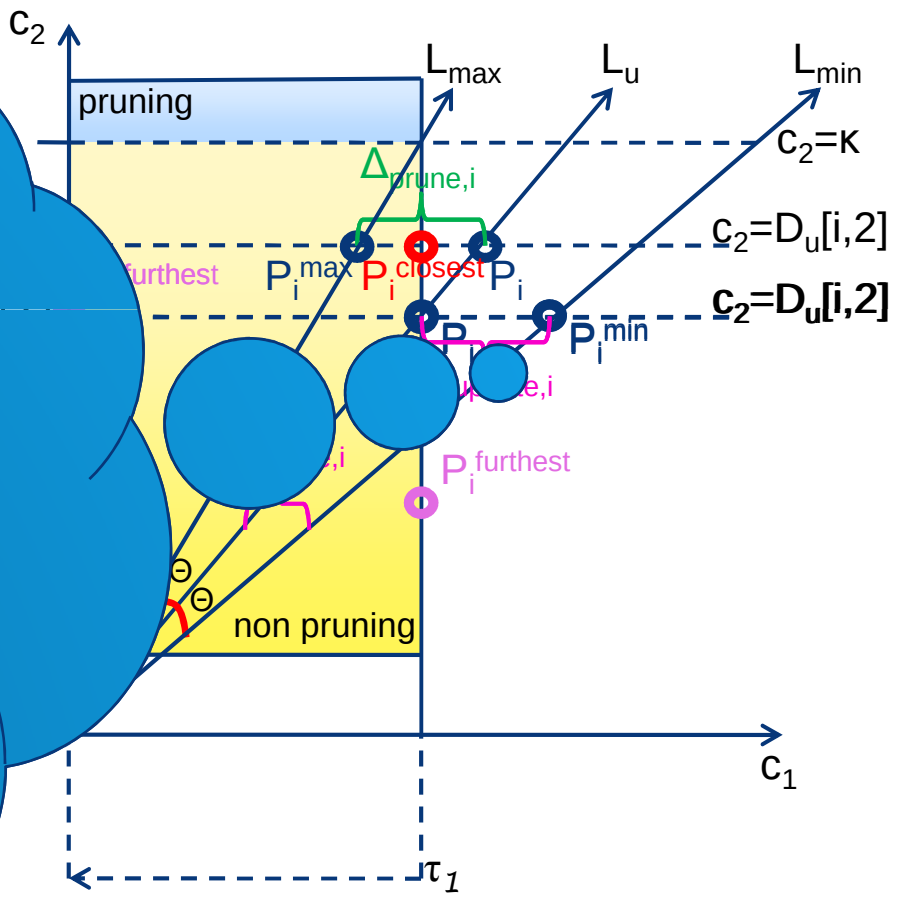
Skip-and-Prune: ulteriori raffinamenti...

- Il raffinamento proposto consiste nel ridefinire *min_score* di modo che possa riflettere i punteggi dei documenti visti solo parzialmente
- Prendiamo il doc[illegible] dei top-k col[illegible]

L'efficienza dell'algoritmo di pruning dipende da quanto stretti sono i valori della distanza di pruning

$$\Delta_{prune,i} = \Delta(P_i^{max}, P_i)$$

per i documenti candidati. Questo valore dipende da *min_score*, che, specialmente negli instanti iniziali del processo, sarà troppo basso per l'assenza di documenti completi

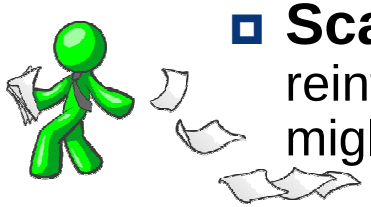


Skip-and-Prune: considerazioni

- Il meccanismo dello *skip set* assicura che i documenti vengano rimossi il prima possibile dagli stream di input per avere un minore tempo di elaborazione totale
- A differenza di TA, questo processo non può terminare finché gli InvertedFiles non siano stati completamente scanditi, a causa della non monotonicità della funzione coseno

Valutazioni Sperimentali

- Sono stati fatti esperimenti per valutare l'**efficienza** dell'approccio proposto, confrontando i risultati con le soluzioni precedenti:



  - **Scan-and-Choose** (SC): scandisce tutti i documenti, li reinterpreta secondo il contesto dell'utente restituendo i migliori k

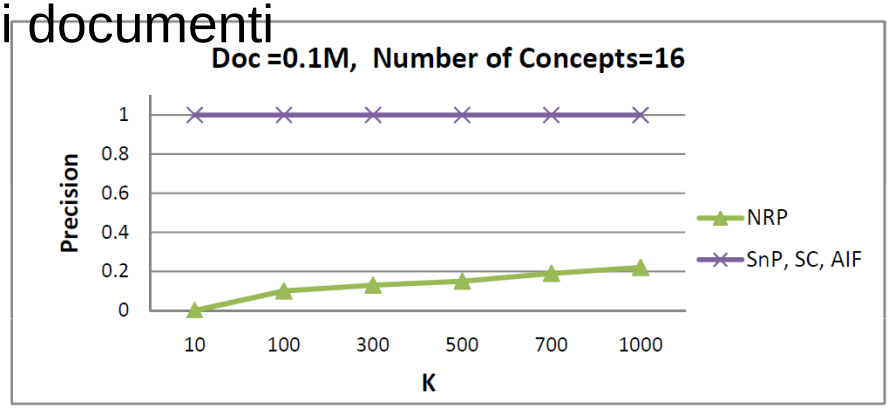
  - **Accumulator-basedInverted File** (AIF): utilizza InvertedFiles e accumulatori per calcolare il punteggio globale di ogni documento

  - **NaiveRanked Processing** (NRP): simile a SnP perché considera solo i documenti rilevanti per ogni concetto, ma utilizzando una funzione di scoring monotona

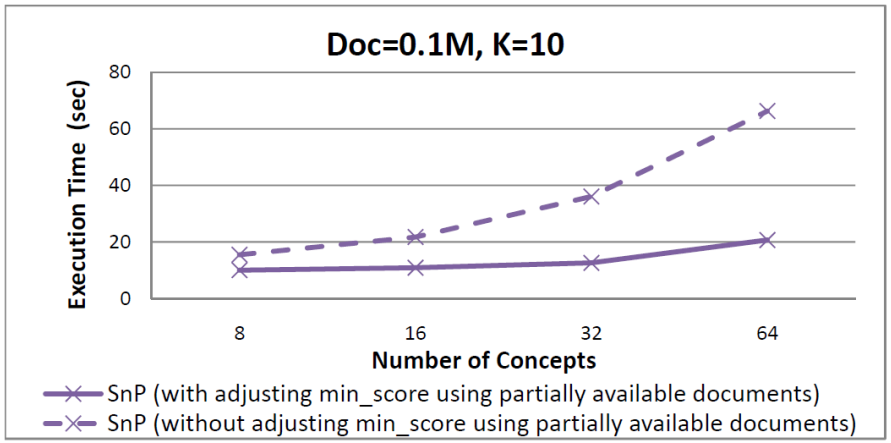
Valutazioni Sperimentali: dati sintetici

Dati sintetici generati sistematicamente, variando parametri come k, il numero dei concetti e il numero di documenti

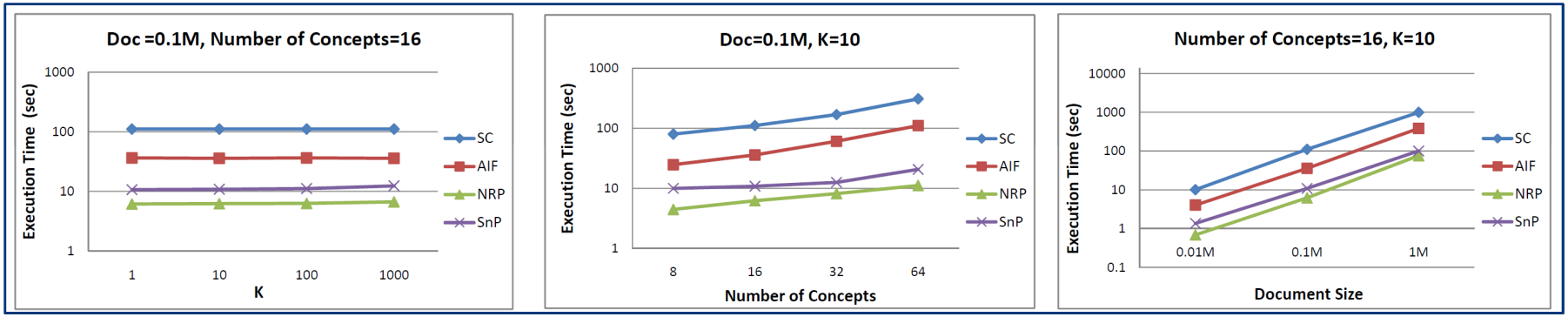
Confronto della precisione media dei top-k risultati in funzione di k, numero dei documenti restituiti all'utente.



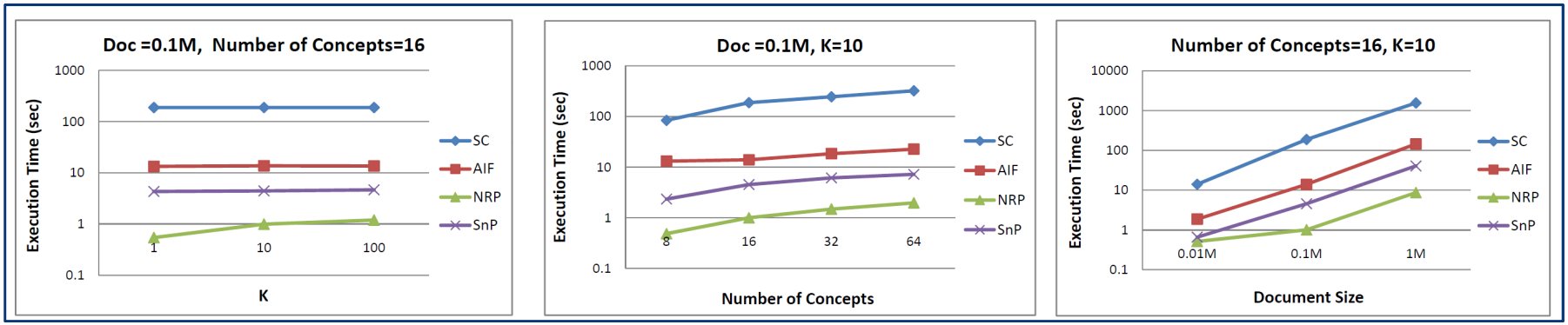
Impatto dell'aggiustamento del *min_score* considerando documenti visti parzialmente



Valutazioni Sperimentali: dati sintetici



Matrici D dense: tempi di esecuzione al variare di differenti parametri

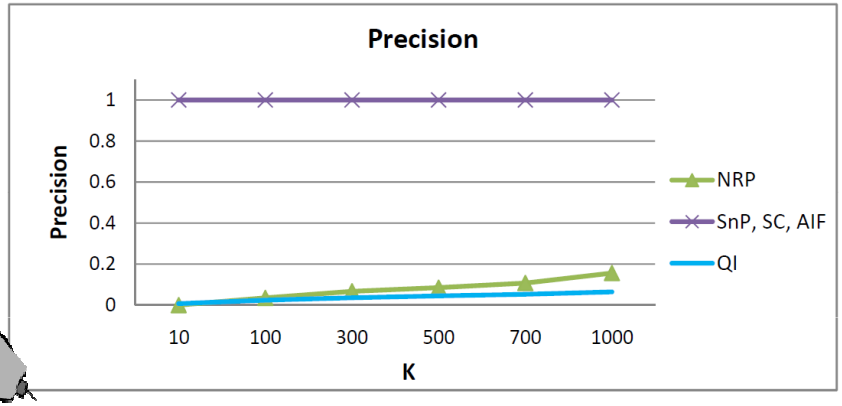


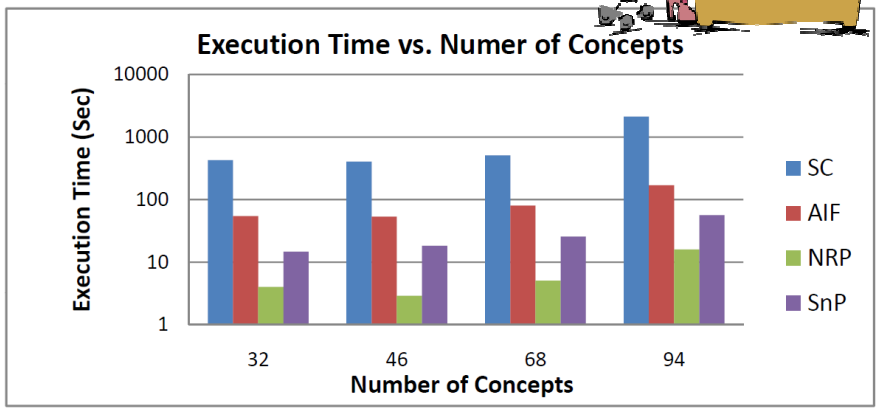
Matrici D sparse: tempi di esecuzione al variare di differenti parametri

Valutazioni Sperimentali: dati reali

Dati reali estratti da ACM digitallibrary e ScienceDirect

Precisioni medie dei top-k risultati sui dati di insiemi reali. Ulteriore algoritmo di confronto: **Q**uery-only**I**nterpretation





Tempi di esecuzione dei differenti algoritmi al variare del numero di concetti significativi.

L'algoritmo SnP qui proposte di uno o due ordini più veloce di SC e AIF

Conclusioni

- L'algoritmo qui presentato, consente efficientemente di calcolare query Top-k in presenza di un contesto utente con una funzione di scoring non monotona basata sul coseno

Vantaggi principali:

- È efficiente e scala molto bene al crescere della dimensione dei documenti e del numero di dimensioni nello spazio dei concetti
- Ha tempi di esecuzione migliori degli algoritmi precedenti dato che consente di eliminare documenti non inerenti al contesto e alla query
- Fornisce risultati ad alta precisione

- Estensioni future possibili ad altre funzioni di scoring non monotone

FINE