https://vvtesh.sarahah.com/

Information Retrieval

Venkatesh Vinayakarao

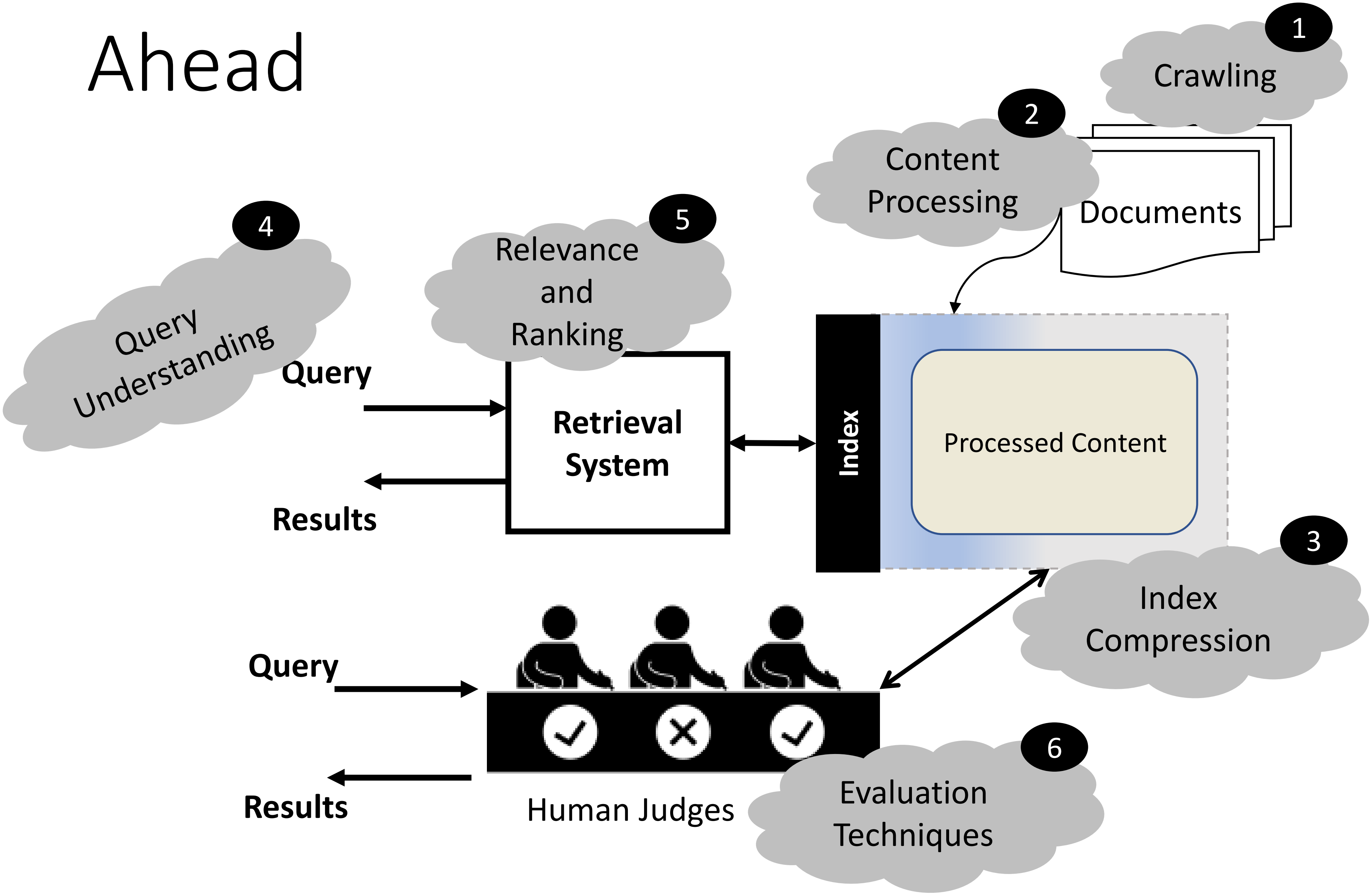
Term: Aug – Sep, 2019

Chennai Mathematical Institute

Computers understand very little of the meaning of human language. This profoundly limits our ability to give instructions to computers, the ability of computers to explain their actions to us, and the ability of computers to analyse and process text. Vector space models (VSMs) of semantics are beginning to address these limits.

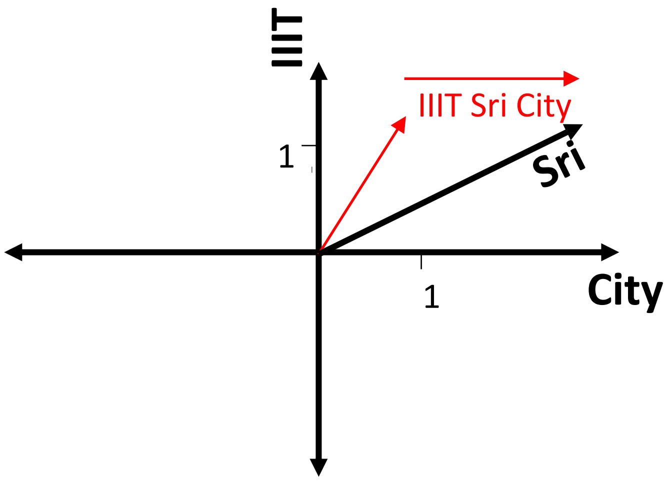
– Turney and Pantel, JAIR 2010.

Information Retrieval – Road Ahead



Sentences are Vectors

- "IIIT Sri City" is a 3-dimensional vector



Example

Let query q = “BITS Pilani”.
Let document, d_1 = “BITS Pilani Goa Campus” and d_2 = “IIIT Delhi”.

	BITS	Pilani	Goa	Campus	IIIT	Delhi
q	1	1	0	0	0	0
d_1	1	1	1	1	0	0
d_2	0	0	0	0	1	1

In our VSM, q = (1,1,0,0,0,0), d_1= (1,1,1,1,0,0) and d_2 = (0,0,0,0,1,1)

similarity(d_1, q) $= \frac{d_1.q}{||d_1||\ ||q||} = \frac{1.1 + 1.1}{\sqrt{1^2+1^2+1^2+1^2}\ \sqrt{1^2+1^2}} = 0.71.$

similarity(d_2, q) $= \frac{d_2.q}{||d_2||\ ||q||} = 0.$

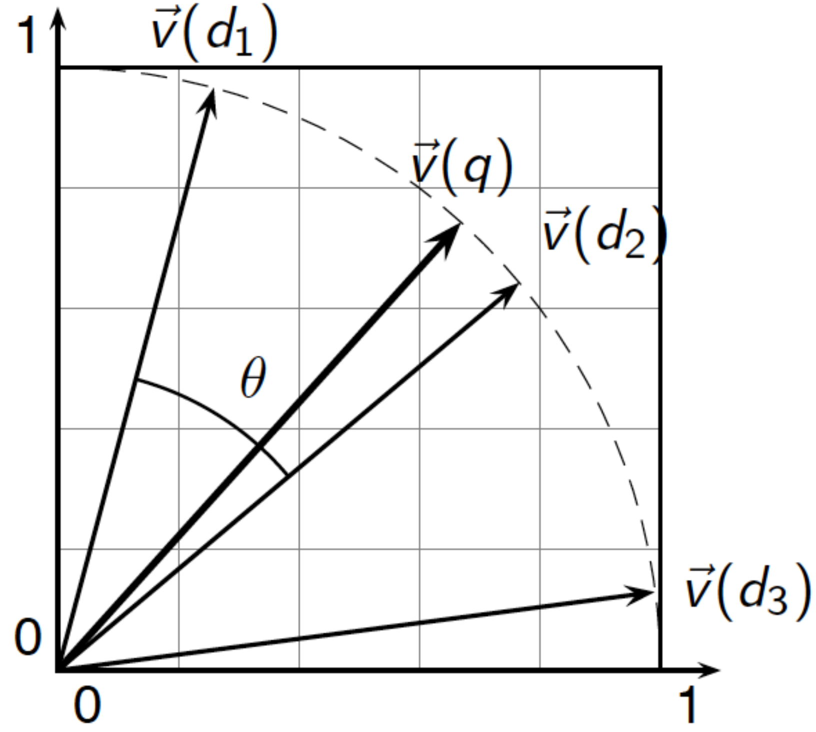
Converting to Unit Vectors

- *Normalization*
  - $\frac{d_2.q}{||d_2||||q||} = \frac{d_2}{||d_2||} \times \frac{q}{||q||}$
  - $\frac{d_2}{||d_2||}$ and $\frac{q}{||q||}$ are unit vectors.

Unit Vectors

- Now, Similarity:

sim(d1,d2) = v(d1).v(d2)



Quiz

- Can you length normalize the vector (1,0,1,2) ?

Answer: (0.41, 0, 0.41, 0.82)

Hint: Normalization Factor = $\sqrt{1^2 + 0 + 1^2 + 2^2}$ = $\sqrt{6}$

Normalized Vector = ($1/\sqrt{6}$, 0, $1/\sqrt{6}$, $2/\sqrt{6}$)

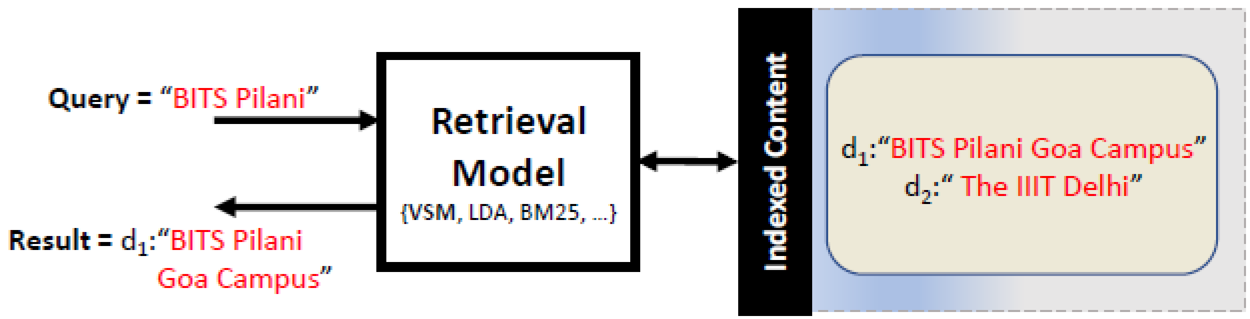
Quiz

- What is the cosine similarity between the **unit** vectors:
  - (0.996, 0.087, 0.017) and
  - (0.993, 0.120, 0)

Answer: 0.999

Hint: Simply take the dot product between the two unit vectors.

Not Every Term is Important



Let us add Term Weights

	BITS	the (* 0)	Pilani	Goa	Campus	IIIT	Delhi
q	1	1 * 0 = 0	1	0	0	0	0
d_1	1	0 * 0 = 0	1	1	1	0	0
d_2	0	1 * 0 = 0	0	0	0	1	1

$sim(q,d_1) = 0.71$

$sim(q,d_2) = 0$

Inverse Document Frequency

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

where N = |D| = Total no. of documents.

$$idf(\text{"}\textit{the}\text{"}, \{\mathbf{d_1}, \mathbf{d_2}\}) = \log \frac{2}{2} = 0$$

$$idf(\text{"}\textit{batsmen}\text{"}, \{\mathbf{d_1}, \mathbf{d_2}\}) = \log \frac{2}{1} = 0.3$$

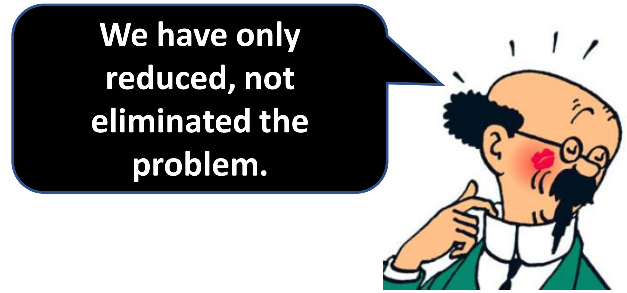
But, do you see any problem? Clue… divide by zero.

Indexed Content

d_1: *"An inverse document frequency factor is incorporated which diminishes* ***the*** *weight of terms that occur very frequently in* ***the*** *document set and increases* ***the*** *weight of terms that occur rarely."*

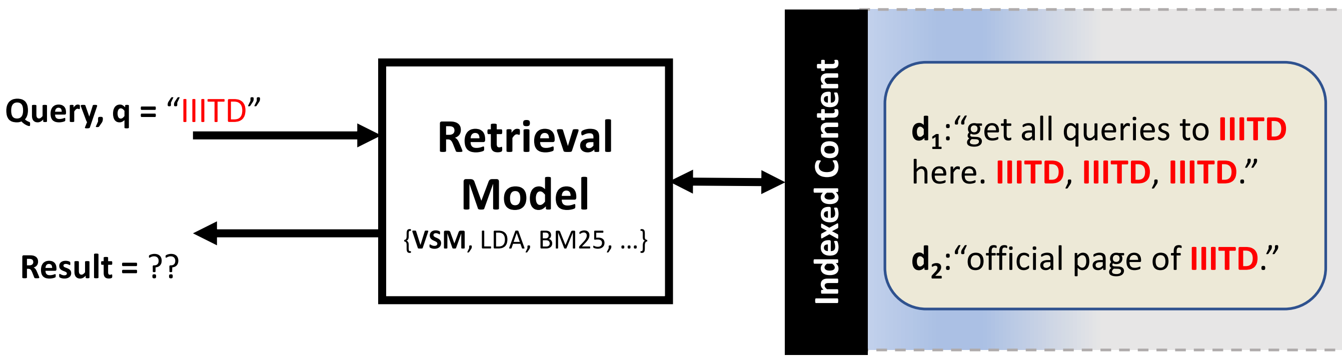
d_2: *"Sachin Ramesh Tendulkar is a former Indian international cricketer and a former captain of* ***the*** *Indian national team, regarded as one of* ***the*** *greatest batsmen of all time."*

Length Normalization



$$tf_L(t, dj) = tf(t, dj) \ * \log(1 + \frac{Avg.DL}{len(dj)})$$

$$Avg.DL = \frac{9+4}{2} = 6.5$$



$$tf_L(\text{"IIITD"}, d_1) = 4 \ * \log(1 + \frac{6.5}{9}) = 0.94$$

$$tf_L(\text{"IIITD"}, d_2) = 1 \ * \log(1 + \frac{6.5}{4}) = 0.42$$

Distinct Terms

Let | $\mathbf{d_1}$| = | $\mathbf{d_2}$ | = 20.

|{distinct terms in $\mathbf{d_1}$}| = 5 , tf("IIITD", $\mathbf{d_1}$) = 4.

|{distinct terms in $\mathbf{d_2}$}| = 15, tf("IIITD", $\mathbf{d_2}$) = 4.

Which document will you prefer?

d_1 = IIITD works in IR. Venkatesh works in IIITD. In IIITD, Venkatesh works in IR. Venkatesh works. IR works. IIITD works.

d_2 = Welcome to the official page of IIITD. IIITD has four departments. IIITD has research focus. Come to IIITD. Schedule visit.

Prefer d_2: Has Less Distinct Terms

Length Regulated TF, LRTF = $tf_L(t, dj) = tf(t, d_j) \ * \log(1 + \frac{Avg.DL}{len(d_j)})$

$tf_L(\text{"IIITD"}, d_1) = 4 \ * \log(1 + \frac{20}{20})$ = 1.2

$tf_L(\text{"IIITD"}, d_2) = 4 \ * \log(1 + \frac{20}{20})$ = 1.2

Relative Intra-Term TF, RITF = $tf_R(t, dj) = \frac{tf(t,dj)}{Avg.TF(d_j)}$

$tf_R(\text{"IIITD"}, d_1) = \frac{20}{4}$ = 5

$tf_R(\text{"IIITD"}, d_2) = \frac{20}{1.3}$ = 15.4

A Case Where RITF Fails

Let | $\mathbf{d_1}$| = 20. | $\mathbf{d_2}$ | = 200.

|{distinct terms in $\mathbf{d_1}$}| = 15, tf("IIITD", $\mathbf{d_1}$) = 4.

|{distinct terms in $\mathbf{d_2}$}| = 150, tf("IIITD", $\mathbf{d_2}$) = 4.

Which document will you prefer?

- RITF = $\mathrm{tf_R}(t, dj) = \frac{tf(t,dj)}{Avg.TF(dj)}$
- In this case, $\mathrm{tf_R}(\text{"IIITD"}, d_1) = \mathrm{tf_R}(\text{"IIITD"}, d_2)$
- We prefer $\mathrm{tf_L}$ instead.

The Okapi Ranking Function

- First implemented in the Okapi Information Retrieval System at the London City University
- Many variants evolved since then
  - BM11 (Stands for Best Match)
  - BM15
  - BM25
  - BM25F

The BM25 Ranking Function

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

K1 and b are free variables chosen empirically. Typical values are:

$k_1 \in [1.2, 2.0]$ and $b = 0.75$.

VSM is expensive to be applied on all documents for a query

If we are only interested in top-k documents, how can we improve our matching algorithm?

In-exact Top-K Retrieval

- Guess the candidate set
- Apply VSM on that set

Candidate set need not contain the top scoring document

Index Elimination

Consider only those documents that have IDF > a threshold

(and/or)

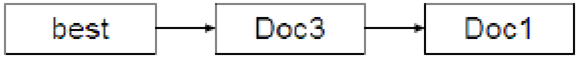
Consider only those documents that have several query terms

Use a Champions List

Pre-compute top docs for certain (popular) query terms

In-Exact Top-K Retrieval

Order Postings List by Popularity



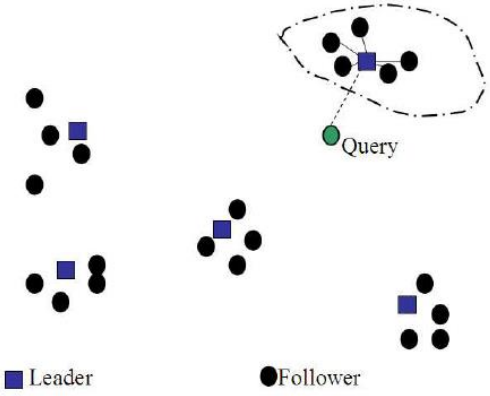
$$\text{net-score}(q,d) = g(d) + \frac{\vec{V}(q)\cdot\vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|}$$

g(d) is the popularity of document d

Cluster Pruning

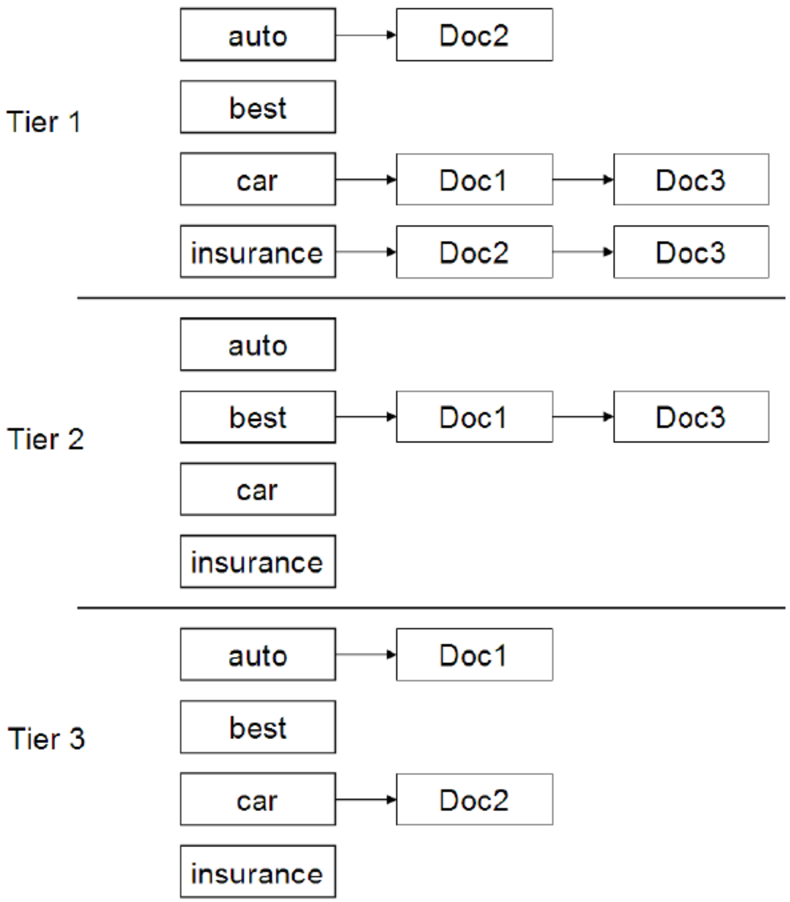
At query time, consider only a small number of clusters.

But, how to cluster the documents?



Tiered Indexes

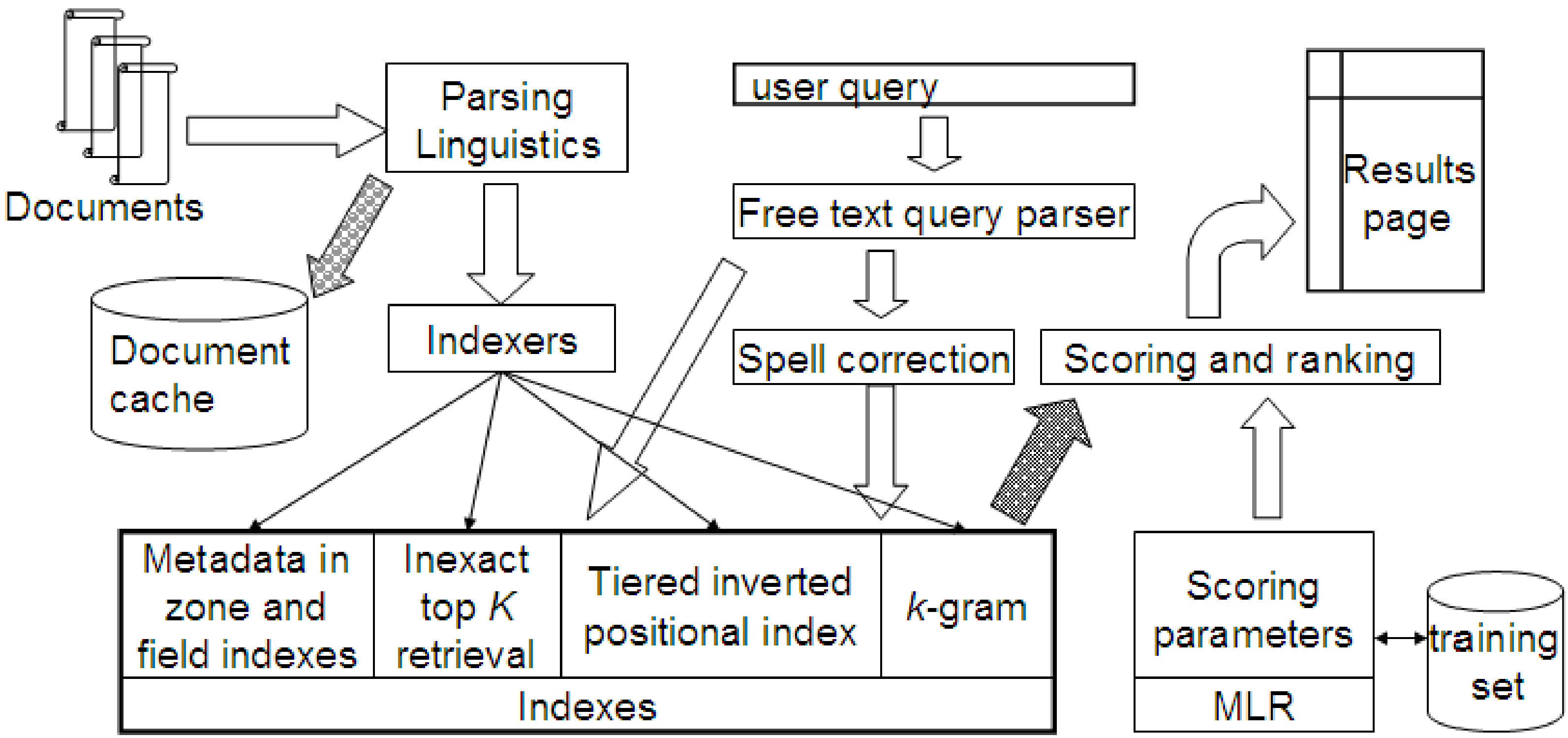
- Extends the idea of champion lists
  - Tiers are ordered by term frequency.
    - E.g., Tier 1 has docIDs with tf > 20
    - Tier 2 has docIDs with tf > 10
    - …



Proximity Weighting

- Ideally, query terms should occur closer to each other in the document.
  - Say w is the smallest window containing all query terms in some document d.
  - If d has the text “CMI is located in Siruseri” and if the query is “CMI Siruseri”, w is 5.
- Design a scoring function that rewards smaller w (or punishes bigger w values).

An Overview of a Complete Search System



Reading

- A novel TF-IDF weighting scheme for effective ranking, Jiaul Paik, SIGIR, 2013.
- The Probabilistic Relevance Framework: BM25 and Beyond, Stephen Robertson & Hugo Zaragoza, 2009.