Problem Statement
Answering conjunctive queries

query: latte macchiato

Latte \[ \begin{array}{c}
3 \rightarrow 7 \rightarrow 9 \rightarrow 14 \rightarrow 19 \rightarrow 23 \rightarrow 41 \rightarrow 47
\end{array} \]

Macchiato \[ \begin{array}{c}
2 \rightarrow 4 \rightarrow 7 \rightarrow 11 \rightarrow 19 \rightarrow 41 \rightarrow 57 \rightarrow 62
\end{array} \]
Answering conjunctive queries

**query:** latte macchiato

Latte: 3 → 7 → 9 → 14 → 19 → 23 → 41 → 47

Macchiato: 2 → 4 → 7 → 11 → 19 → 41 → 57 → 62

Compute the intersection of 2 sorted lists
Merging

Latte

Macchiato
Merging

Latte

Macchiato
Merging

Latte: 3 → 7 → 9 → 14 → 19 → 23 → 41 → 47

Macchiato: 2 → 4 → 7 → 9 → 19 → 41 → 57 → 62
Merging

Latte

Macchiato
Merging

Latte

Macchiato
Skips

Latte 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8

Macchiato 7 → 8 → 17 → 18 → 19 → 41 → 57 → 62
Conventional WSDM

Skips are placed every $\sqrt{N}$ many positions
Question

If we know the query distribution, can we place skips better?
Problem statement

If we know the query distribution, can we place skips in order to minimize the expected time of a merge?
If we know the query distribution, can we place skips in order to minimize the expected time of a merge?

Is the assumption realistic?
The Power of the Law

![Graph showing query count against query rank with a power law fit.](image)
The query distribution contains a lot of information. Can we **provably** take advantage of it?
Algorithms to follow work with any query distribution whatsoever.
Algorithms to follow work with any query distribution whatsoever.

..and can be extended to deal with soft conjunctions.
Outline

• Skip placement policies
• A matter of definitions
• Algorithms
• Experiments
Skip Placement Policies
Spaghetti Skips
Spaghetti Skips
Simple Skips
Simple Skips

This is the most interesting case
A Matter of Definitions
Useful Documents: 1

$q : \text{world cup}$

Relevant docs are "useful"
But usefulness does not coincide with relevance
Useful Documents: 2

$q : \text{world cup}$

Is the skip useful for $q$?
Is the skip useful for $q$?
Useful Documents: 2

q : world cup

world

14 → 15 → ? → 18 → 47 → ? → ? → ?

cup

13 → 19 → 41 → 43 → 62 → ? → ? → ?

Is the skip useful for q?
Useful Documents: 2

$q$: world cup

The skip is useful
Useful Documents: 2

q : world cup

world

cup
Useful Documents: 2

\textbf{q : world cup}


\textbf{cup} 13 → 19 → 41 → 43 → 62 → ? → ? → ?
Useful Documents: 2

$q : \text{world cup}$

world

14 → 15 → 16 → ? → 47 → ? → ? → ?

cup

13 → 19 → 41 → 43 → 62 → ? → ? → ?
Useful Documents: 2

q : world cup

world

14 → 15 → 16 → 18 → 47 → ? → ? → ?

cup

13 → 19 → 41 → 43 → 62 → ? → ? → ?
Useful Documents: 2

q : world cup

world

14 → 15 → 16 → 18 → 47 → ? → ? → ?

cup

13 → 19 → 41 → 43 → 62 → ? → ? → ?
Useful Documents: 2

$q : \text{world cup}$
Useful Documents: 2

$q : \text{world cup}$

world

14 → 15 → 16 → 18 → 47 → ? → ? → ?

cup

13 → 19 → 41 → 43 → 62 → ? → ? → ?

The skip is useless
Useful Documents: 2

q : world cup

world

14 → 15 → 16 → 18 → 47 → ? → ? → ?

cup

13 → 19 → 41 → 43 → 62 → ? → ? → ?

18 cannot be skipped

The skip is useless
Useful Documents: 2

Useful documents are those that cannot be avoided during a merge.
Induced Distributions

The query distribution induces another distribution on the postings

platypus

1 → 2 → 3 → i → j → k → n

p₁ p₂ pᵢ pₖ pₙ
Induced Distributions

The query distribution induces another distribution on the postings

platypus 1 → 2 → 3 → ... → i → j → k → n

\[ p_i = \Pr(i \text{ useful for } q \mid \text{platypus } \in q) \]
Induced Distributions

The query distribution induces another distribution on the postings

platypus

\[ 1 \rightarrow 2 \rightarrow 3 \rightarrow i \rightarrow j \rightarrow k \rightarrow n \]

\[ p_1 \rightarrow p_2 \rightarrow p_i \rightarrow p_j \rightarrow p_k \rightarrow p_n \]

We will assume this distribution to be known
Induced Distributions

In practice these probabilities can be approximated using a small sample of the query universe
Making Life Simple

Events like “a is useful” and “b is useful” are not independent

..but from now on we will assume that they are
Making Life Simple

Events like “a is useful” and “b is useful” are not independent.

..but from now on we will assume that they are.

This simplifying assumption will be vindicated by our experiments.
Algorithms
Algorithms

**Input:** a list with, for each doc, the probability that it is useful

**Output:** skip placement that minimizes the expected time to merge
Input: a list with, for each doc, the probability that it is useful
Output: skip placement that minimizes the expected time to merge

cost of a merge = #elements read in posting list
Algorithms

• $O(nt)$ algorithm for spaghetti skips, where $t$ is the average length of a skip
• $O(n \log n)$ for simple skips
Algorithms

• $O(nt)$ algorithm for spaghetti skips, where $t$ is the average length of a skip

• $O(n \log n)$ for simple skips

$O(n \log n)$ algorithm for simple skips is by far the most interesting
Simple Skips

\[ t: d_1d_2..d_i..d_n \ & \ p_1p_2..p_n \]

- Build the solution from left to right
- M(i) is best placement for prefix \( d_1..d_i \)
In computing $M(i)$ we have two choices. We either place a skip landing at position $i$ or we do not.
If we place no skip to $i$ then $M(i) = M(i-1)$
Computing $M(i)$
Computing $M(i)$

$$M(i) = \max \{ M(i-1), \max_j M(j) + G(j, i) \}$$
Computing $M(i)$

$$\max_j M(j) + G(j, i) = M(T(i)) + G(T(i), j)$$
Monotonicity of $T(i)$
Monotonicity of $T(i)$

$T(i) \leq T(i+1)$
Monotonicity of $T(i,k)$

$T(i,k)$ is best jump to $i$ under the additional constraint that it must start no later than $k$. 
Monotonicity of $T(i,k)$

Key lemma: $T(i,k) \leq T(i+1,k)$
Monotonicity of $T(i,k)$

Let $\hat{i}$ be the smallest index $i$ such that $T(i,k)=k$. Then,

\[
T(j,k) = \begin{cases} 
  k & j \geq \hat{i} \\
  T(j,k-1) & j < \hat{i}
\end{cases}
\]
Updating $T(i,k)$

$T(i,k-1)$

\[
\begin{array}{cccccccccccc}
1 & 1 & 1 & 1 & 1 & i_1 & i_1 & i_1 & i_2 & i_2 & i_2 & i_3 & i_3 & i_4 & i_4 & i_4
\end{array}
\]
Updating $T(i,k)$

$T(i,k-1)$

$1 < i_1 < i_2 < i_3 < i_4 < k-1$
Updating $T(i,k)$

The best skip to reach $j$ starts at position $i_1$. 
Updating $T(i,k)$

$T(i,k-1)$

| 1 | 1 | 1 | 1 | 1 | $i_1$ | $i_1$ | $i_1$ | $i_2$ | $i_2$ | $i_2$ | $i_3$ | $i_3$ | $i_4$ | $i_4$ | $i_4$ | $i_4$ |

$T(i,n)$ gives the optimal placement
Updating $T(i,k)$

$T(i, k-1)$

$T(i,n)$ gives the optimal placement

$T(\bullet,1) \rightarrow T(\bullet,2) \rightarrow \ldots \rightarrow T(\bullet,k) \rightarrow \ldots \rightarrow T(\bullet,n)$
Updating $T(i,k)$

$\min \{i : T(i,k)=k\}$

$T(i,k-1)$

1 1 1 1 1 i_1 i_1 i_1 i_2 i_2 i_2 i_3 i_3 i_4 i_4 i_4 i_4$
Updating $T(i,k)$

$T(i,k)$

$T(i,k-1)$

$\min \{i: T(i,k) = k\}$
The resulting algorithm takes $O(N \log N)$ where $N$ is the length of the list
Experiments
Space
Time to merge
Build up time
Size of query sample for spaghetti skips
Size of query sample for simple skips

1/256
The Bottomline

Simple skips are the solution of choice (for power law distributions):

• They merge as fast as spaghetti skips (the general case)
• They occupy less space
• Build time is much faster
• They need a smaller sample to collect statistics on document usefulness
Summing up

• First attempt to exploit in a rigorous way knowledge of the distribution
• Much work remains to be done but results are encouraging
Extensions

- Taking the cache into account
- Taking dependencies into account
- Compare against skip list and other data structures

Thanks for your attention