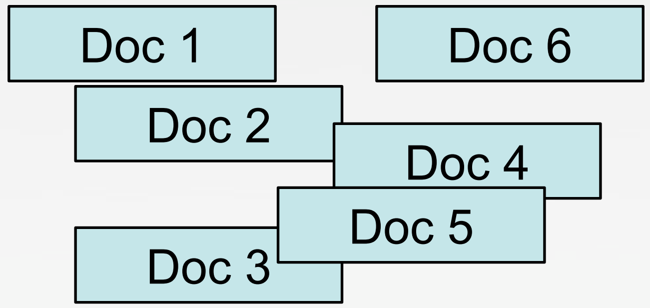
Top-k Aggregation Using Intersections

Ravi Kumar	Yahoo! Research
Kunal Punera	Yahoo! Research
Torsten Suel	Yahoo! Research / Brooklyn Poly
Sergei Vassilvitskii	Yahoo! Research

Top-k retrieval

Given a set of documents:

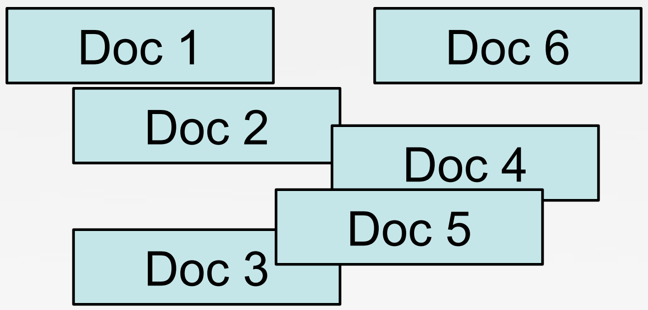


And a query: "*New York City*"

Find the k documents best matching the query.

Top-k retrieval

Given a set of documents:



And a query: “*New York City*”

Find the k documents best matching the query.

Assume: decomposable scoring function:

Score(“New York City”) = Score(“New”) + Score(“York”)+Score(“City”).

Introduction: Postings Lists

Data Structures behind top-k retrieval.

Create posting lists:

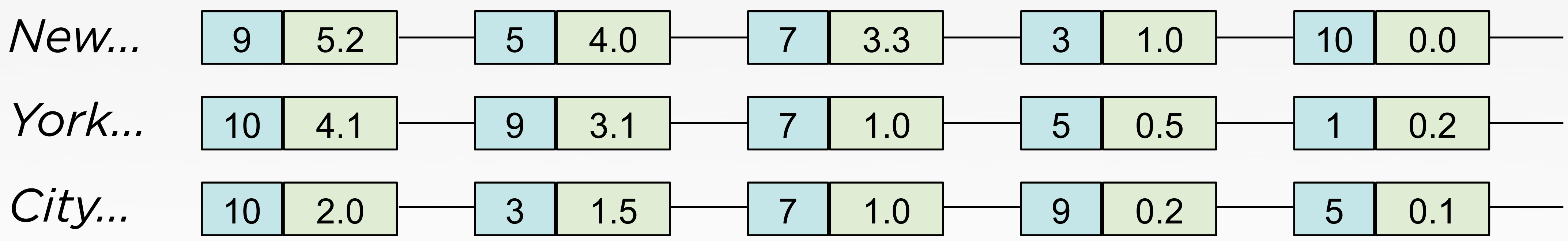
Doc ID	Score

Introduction: Postings Lists

Data Structures behind top-k retrieval.

Create posting lists: | Doc ID | Score |

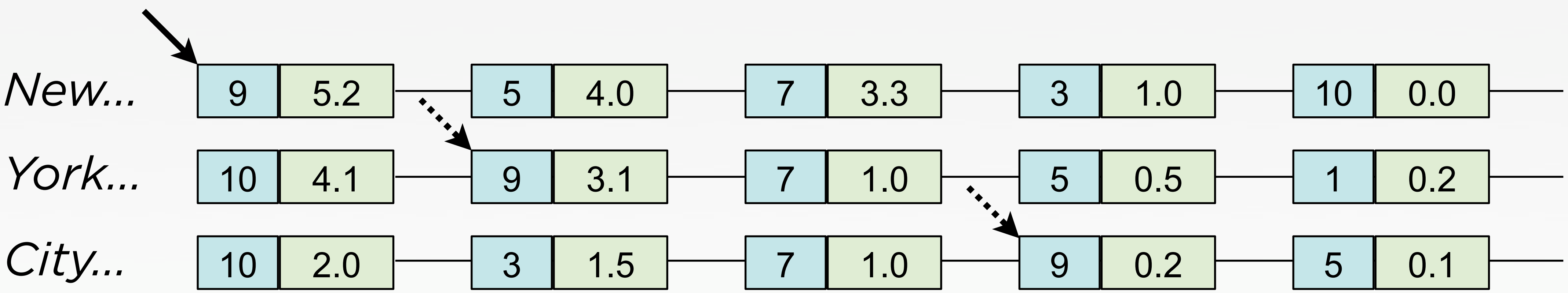
Query: *New York City*



Introduction: Postings Lists

(Offline) Sort each list by decreasing score.

Query: *New York City*



Retrieval: Start with document with highest score in any list.

Look up its score in other lists.

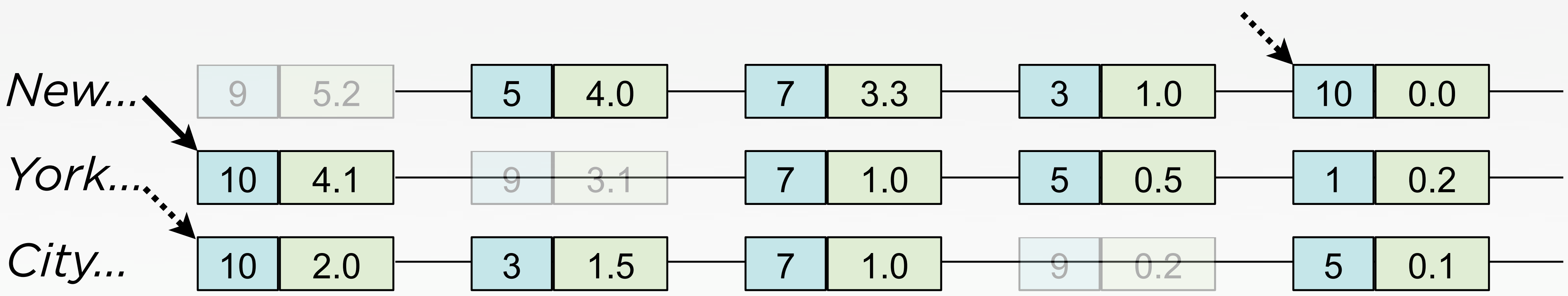
Top: 9 | 5.2+3.1+0.2=8.5

Introduction: Postings Lists

Data Structures behind top-k retrieval:

Arrange each list by decreasing score.

Query: *New York City*



Continue with next highest score.

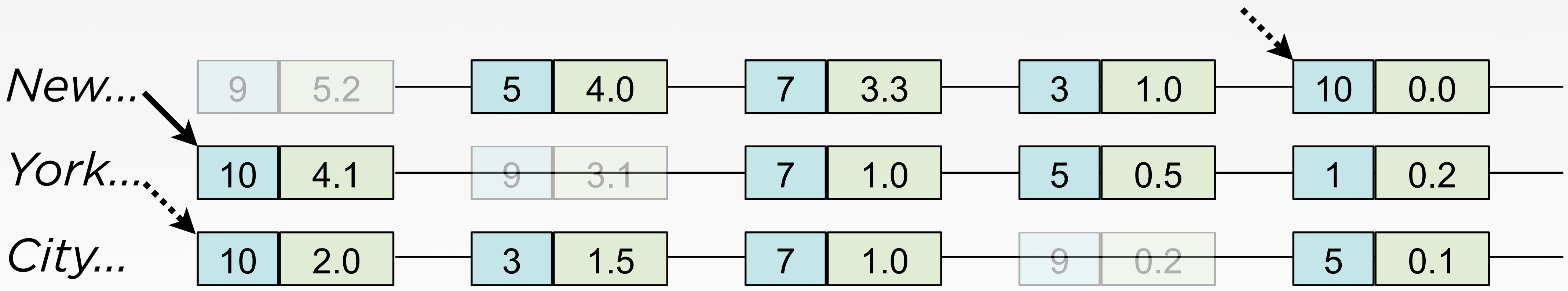
Top: | 9 | 8.5 | Candidate: | 10 | 4.1+2.0+0.0 = 6.1 |

Introduction: Postings Lists

Data Structures behind top-k retrieval:

Arrange each list by decreasing score.

Query: *New York City*



Continue with next highest score.

Top: 9 | 8.5

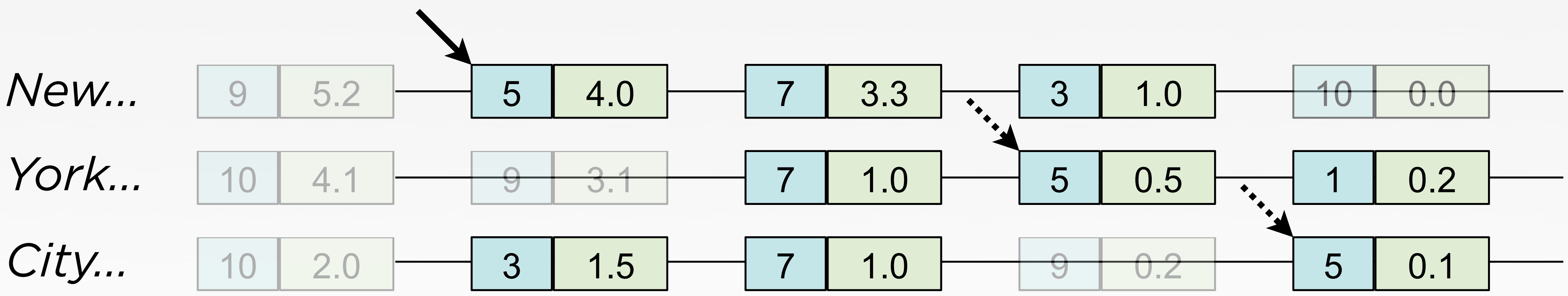
Candidate: 10 | 4.1+2.0+0.0 = 6.1

Introduction: Postings Lists

Data Structures behind top-k retrieval:

Arrange each list by decreasing score.

Query: *New York City*



Continue with next highest score.

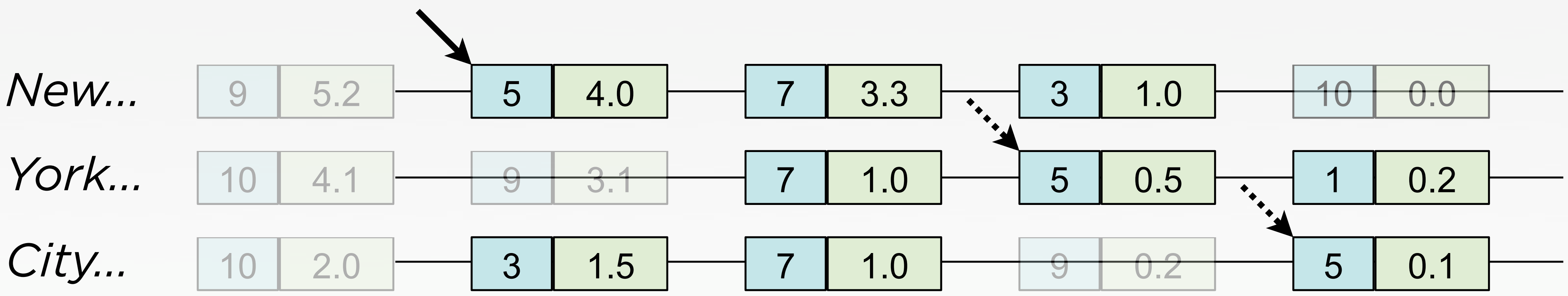
Top: 9 | 8.5   Candidate: 5 | 4.0+0.5+0.1=4.6

Introduction: Postings Lists

Data Structures behind top-k retrieval:

Arrange each list by decreasing score.

Query: *New York City*



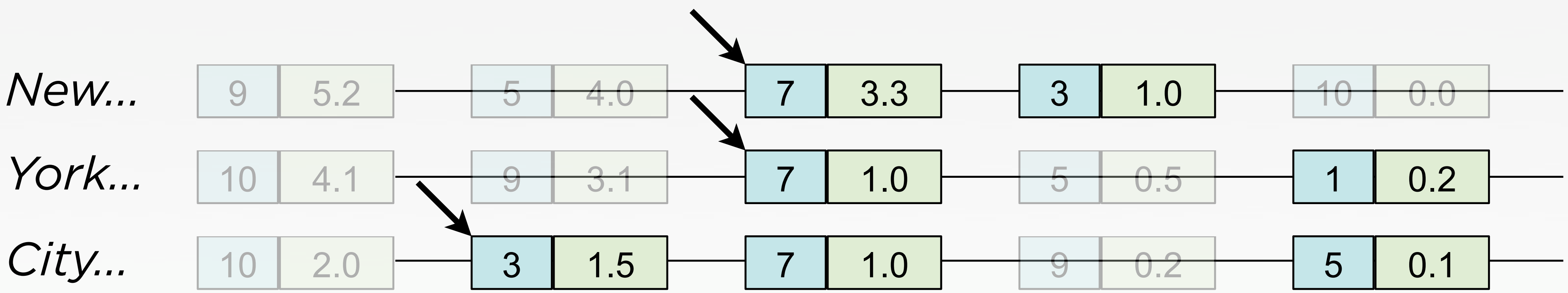
Continue with next highest score.

Introduction: Postings Lists

Data Structures behind top-k retrieval:

Arrange each list by decreasing score.

Query: *New York City*



When can we stop?

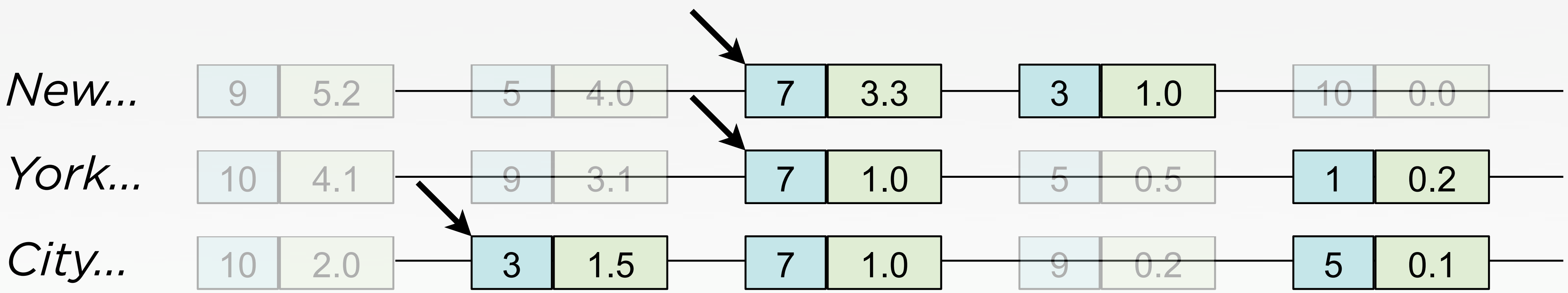
Top: | 9 | 8.5 | Best Possible Remaining: | * | 3.3+1.5+1.0=5.8 |

Introduction: Postings Lists

Data Structures behind top-k retrieval:

Arrange each list by decreasing score.

Query: *New York City*



When can we stop?

Top: 9 | 8.5 Best Possible Remaining: * | ~~3.3+1.5+1.0=5.8~~

Threshold Algorithm

Threshold Algorithm (TA)

- Instance optimal (in # of accesses) [Fagin et al]
- Performs random accesses

No-Random-Access Algorithm (NRA)

- Similar to TA
- Keep a list of all seen results
- Also instance optimal

Introducing bi-grams

Introducing bi-grams

Certain words often occur as phrases. Word association:

Introducing bi-grams

Certain words often occur as phrases. Word association:

- Sagrada ...

Introducing bi-grams

Certain words often occur as phrases. Word association:

- Sagrada …
- Barack …

Introducing bi-grams

Certain words often occur as phrases. Word association:

- Sagrada ...
- Barack ...
- Latent Semantic...

Introducing bi-grams

Certain words often occur as phrases. Word association:

- Sagrada ...
- Barack ...
- Latent Semantic...

Pre-compute posting lists for intersections

- Note, this is not query-result caching

Tradeoffs:

- Space: extra space to store the intersection (though it's smaller)
- Time: Less time upon retrieval

Bi-grams & TA

Query: New York City

All aggregations -- 6 lists.

[New] [York] [City] [New York] [New City] [York City]

Bi-grams & TA

Query: New York City

All aggregations -- 6 lists.

[New] [York] [City] [New York] [New City] [York City]

New	9	5.2	5	4.0	7	3.3	3	1.0	10	0.0
York	10	4.1	9	3.1	7	1.0	5	0.5	1	0.2
City	10	2.0	3	1.5	7	1.0	9	0.2	5	0.1
NY	9	8.3	5	4.5	7	4.3	10	4.1	3	1.0
NC	9	5.4	7	4.3	5	4.1	3	2.5	10	2.0
YC	10	6.1	9	3.3	7	2.0	3	1.5	5	0.6

Bi-grams & TA

Query: New York City

All aggregations -- 6 lists.

[New] [York] [City] [New York] [New City] [York City]

New	9	5.2	5	4.0	7	3.3	3	1.0	10	0.0
York	10	4.1	9	3.1	7	1.0	5	0.5	1	0.2
City	10	2.0	3	1.5	7	1.0	9	0.2	5	0.1
NY	9	8.3	5	4.5	7	4.3	10	4.1	3	1.0
NC	9	5.4	7	4.3	5	4.1	3	2.5	10	2.0
YC	10	6.1	9	3.3	7	2.0	3	1.5	5	0.6

Top: 9 | 8.5

Bi-grams & TA

Query: New York City

All aggregations -- 6 lists.

[New] [York] [City] [New York] [New City] [York City]

New	9	5.2	5	4.0	7	3.3	3	1.0	10	0.0
York	10	4.1	9	3.1	7	1.0	5	0.5	1	0.2
City	10	2.0	3	1.5	7	1.0	9	0.2	5	0.1
NY	9	8.3	5	4.5	7	4.3	10	4.1	3	1.0
NC	9	5.4	7	4.3	5	4.1	3	2.5	10	2.0
YC	10	6.1	9	3.3	7	2.0	3	1.5	5	0.6

Top: 9 | 8.5

Can we stop now?

TA Bounds Informal

New	9	5.2	5	4.0	7	3.3	3	1.0	10	0.0
York	10	4.1	9	3.1	7	1.0	5	0.5	1	0.2
City	10	2.0	3	1.5	7	1.0	9	0.2	5	0.1
NY	9	8.3	5	4.5	7	4.3	10	4.1	3	1.0
NC	9	5.4	7	4.3	5	4.1	3	2.5	10	2.0
YC	10	6.1	9	3.3	7	2.0	3	1.5	5	0.6

Top: 9 | 8.5

Bounds on any unseen element:

N + Y + C = 10.1

TA Bounds Informal

New	9	5.2	5	4.0	7	3.3	3	1.0	10	0.0
York	10	4.1	9	3.1	7	1.0	5	0.5	1	0.2
City	10	2.0	3	1.5	7	1.0	9	0.2	5	0.1
NY	9	8.3	5	4.5	7	4.3	10	4.1	3	1.0
NC	9	5.4	7	4.3	5	4.1	3	2.5	10	2.0
YC	10	6.1	9	3.3	7	2.0	3	1.5	5	0.6

Top: | 9 | 8.5 |

Bounds on any unseen element:

N + Y + C = 10.1

NY + C = 6.5

TA Bounds Informal

New	9 5.2	5 4.0	7 3.3	3 1.0	10 0.0
York	10 4.1	9 3.1	7 1.0	5 0.5	1 0.2
City	10 2.0	3 1.5	7 1.0	9 0.2	5 0.1
NY	9 8.3	5 4.5	7 4.3	10 4.1	3 1.0
NC	9 5.4	7 4.3	5 4.1	3 2.5	10 2.0
YC	10 6.1	9 3.3	7 2.0	3 1.5	5 0.6

Top: 9 8.5

Bounds on any unseen element:

N + Y + C = 10.1

NY + C = 6.5

NC + Y = 8.4

YC + N = 10.1

TA Bounds Informal

New	9	5.2	5	4.0	7	3.3	3	1.0	10	0.0
York	10	4.1	9	3.1	7	1.0	5	0.5	1	0.2
City	10	2.0	3	1.5	7	1.0	9	0.2	5	0.1
NY	9	8.3	5	4.5	7	4.3	10	4.1	3	1.0
NC	9	5.4	7	4.3	5	4.1	3	2.5	10	2.0
YC	10	6.1	9	3.3	7	2.0	3	1.5	5	0.6

Top: | 9 | 8.5 |

Bounds on any unseen element:

N + Y + C = 10.1
NY + C = 6.5
NC + Y = 8.4
YC + N = 10.1
1/2 (NY + YC + NC) = 7.45

TA Bounds Informal

New	9	5.2	5	4.0	7	3.3	3	1.0	10	0.0
York	10	4.1	9	3.1	7	1.0	5	0.5	1	0.2
City	10	2.0	3	1.5	7	1.0	9	0.2	5	0.1
NY	9	8.3	5	4.5	7	4.3	10	4.1	3	1.0
NC	9	5.4	7	4.3	5	4.1	3	2.5	10	2.0
YC	10	6.1	9	3.3	7	2.0	3	1.5	5	0.6

Top: | 9 | 8.5 |

Bounds on any unseen element:

N + Y + C = 10.1
NY + C = 6.5
NC + Y = 8.4
YC + N = 10.1
1/2 (NY + YC + NC) = 7.45

Thus best element has score < 6.5. So we are done!

TA: Bounds Formal

Can we write the bounds on the next element?

x_i : score of document x in list i.

b_i : bound on the score in list i (score of next unseen document)

Combinations: b_{ij} bound on $x_i + x_j$

Simple LP for bound on unseen elements:

$$\max \sum_i x_i$$

$$x_i \leq b_i$$

$$x_i + x_j \leq b_{ij}$$

In theory: Easy! Just solve an LP every time.

In reality: You're kidding, right?

Solving the LP

Need to solve the LP:

$$\max \sum_i x_i$$

$$x_i \leq b_i$$

$$x_i + x_j \leq b_{ij}$$

Same as solving the dual

$$\min \sum y_{ij} b_{ij} + \sum y_i b_i$$

$$y_i + \sum_j y_{ij} \geq 1$$

$$y_i, y_{ij} \geq 0$$

The dual as a graph

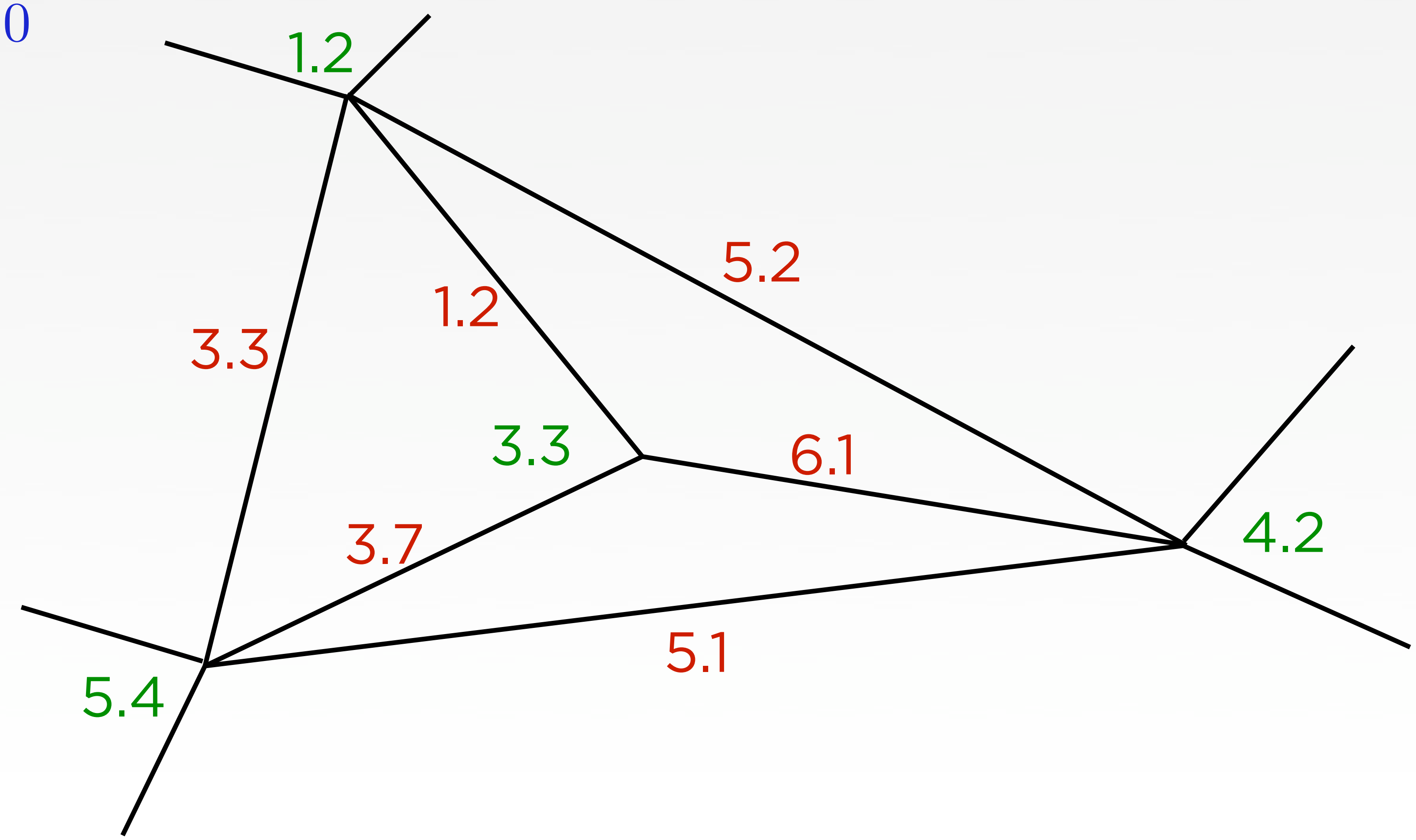
$$\min \sum y_{ij} b_{ij} + \sum y_i b_i$$

$$y_i + \sum_j y_{ij} \geq 1$$

$$y_i, y_{ij} \geq 0$$

Add one node for each y_i with weight b_i

Add one edge for each y_{ij} with weight b_{ij}



The dual as a graph

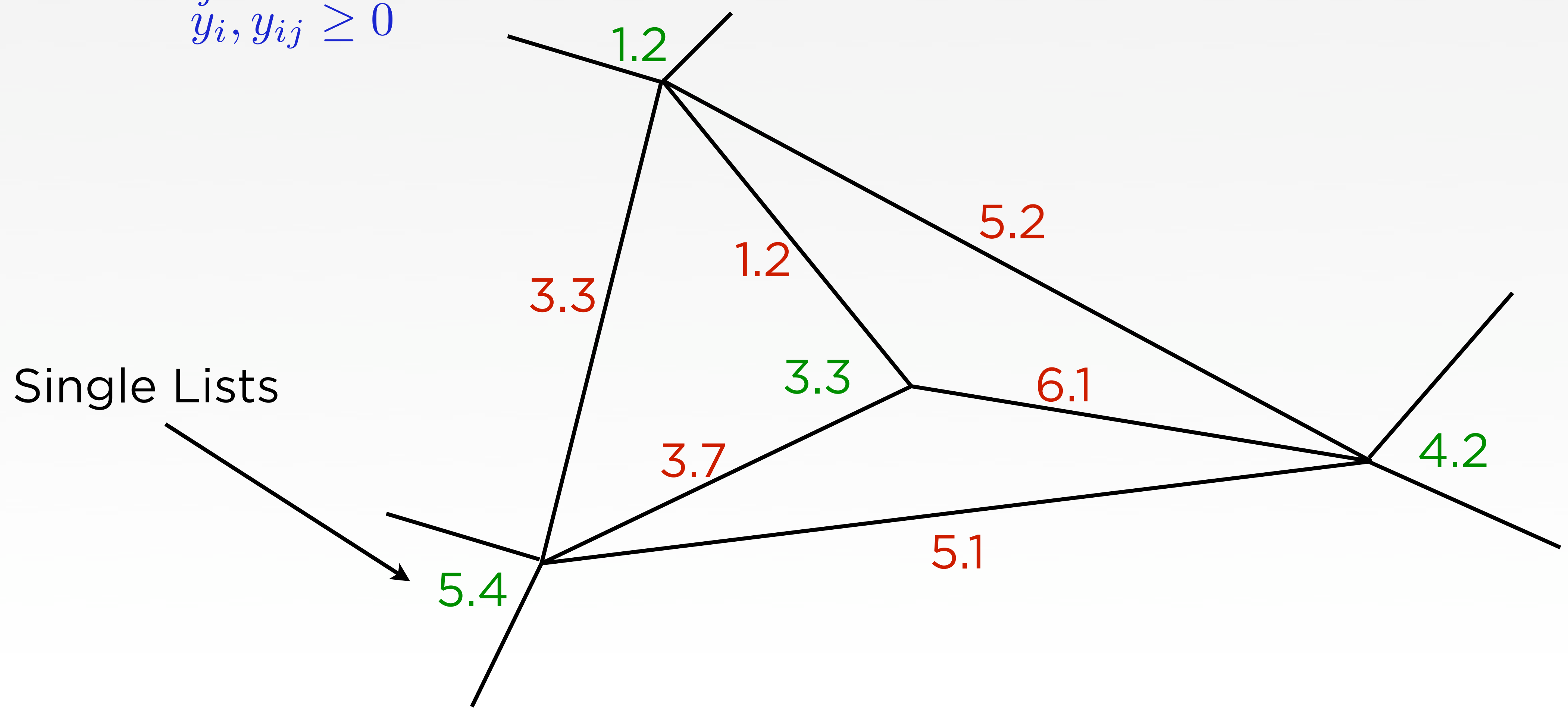
$$\min \sum y_{ij} b_{ij} + \sum y_i b_i$$

$$y_i + \sum_j y_{ij} \geq 1$$

$$y_i, y_{ij} \geq 0$$

Add one node for each y_i with weight b_i

Add one edge for each y_{ij} with weight b_{ij}



The dual as a graph

$$\min \sum y_{ij} b_{ij} + \sum y_i b_i$$

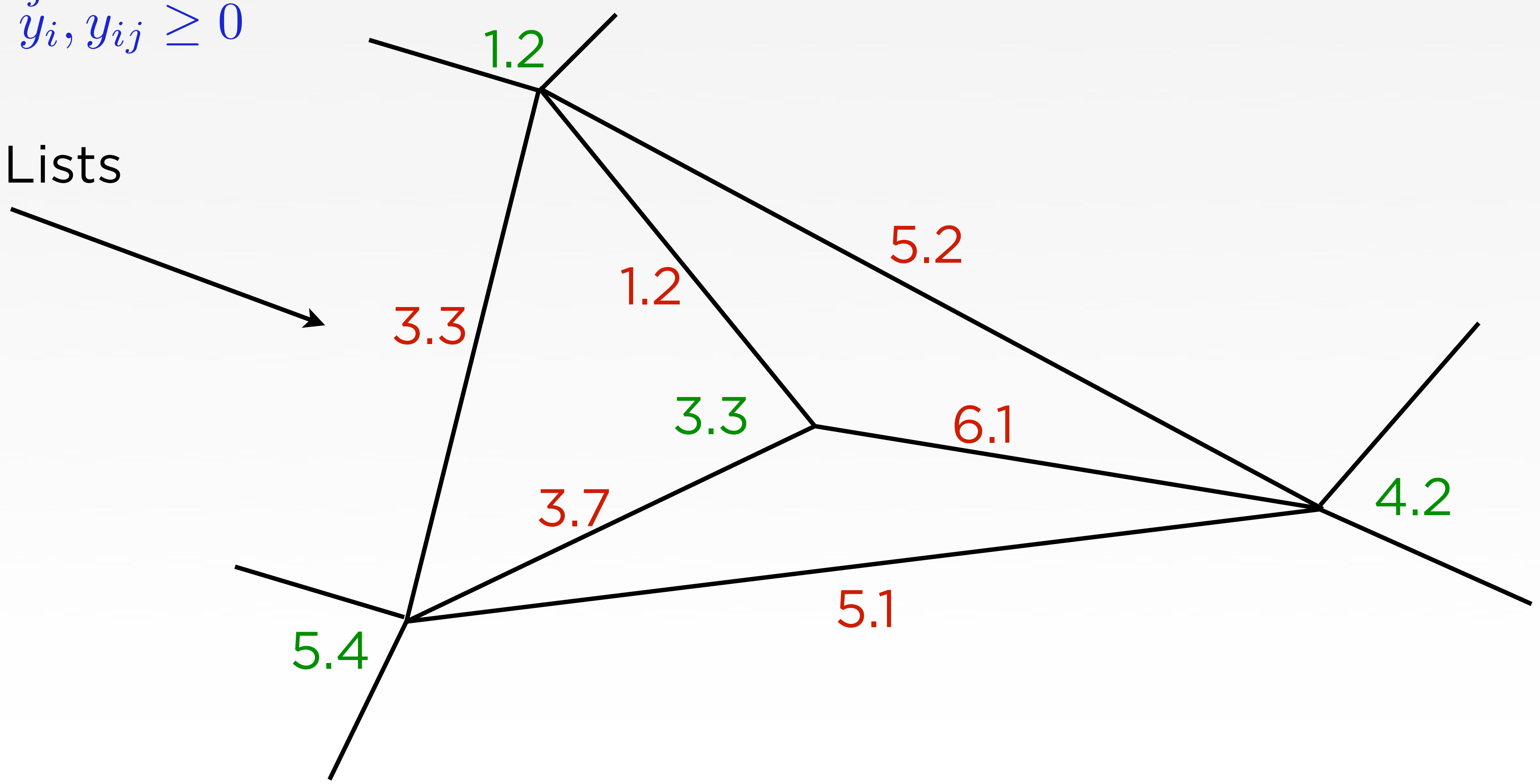
$$y_i + \sum_j y_{ij} \geq 1$$

$$y_i, y_{ij} \geq 0$$

Add one node for each y_i with weight b_i

Add one edge for each y_{ij} with weight b_{ij}

Paired Lists



The dual as a graph

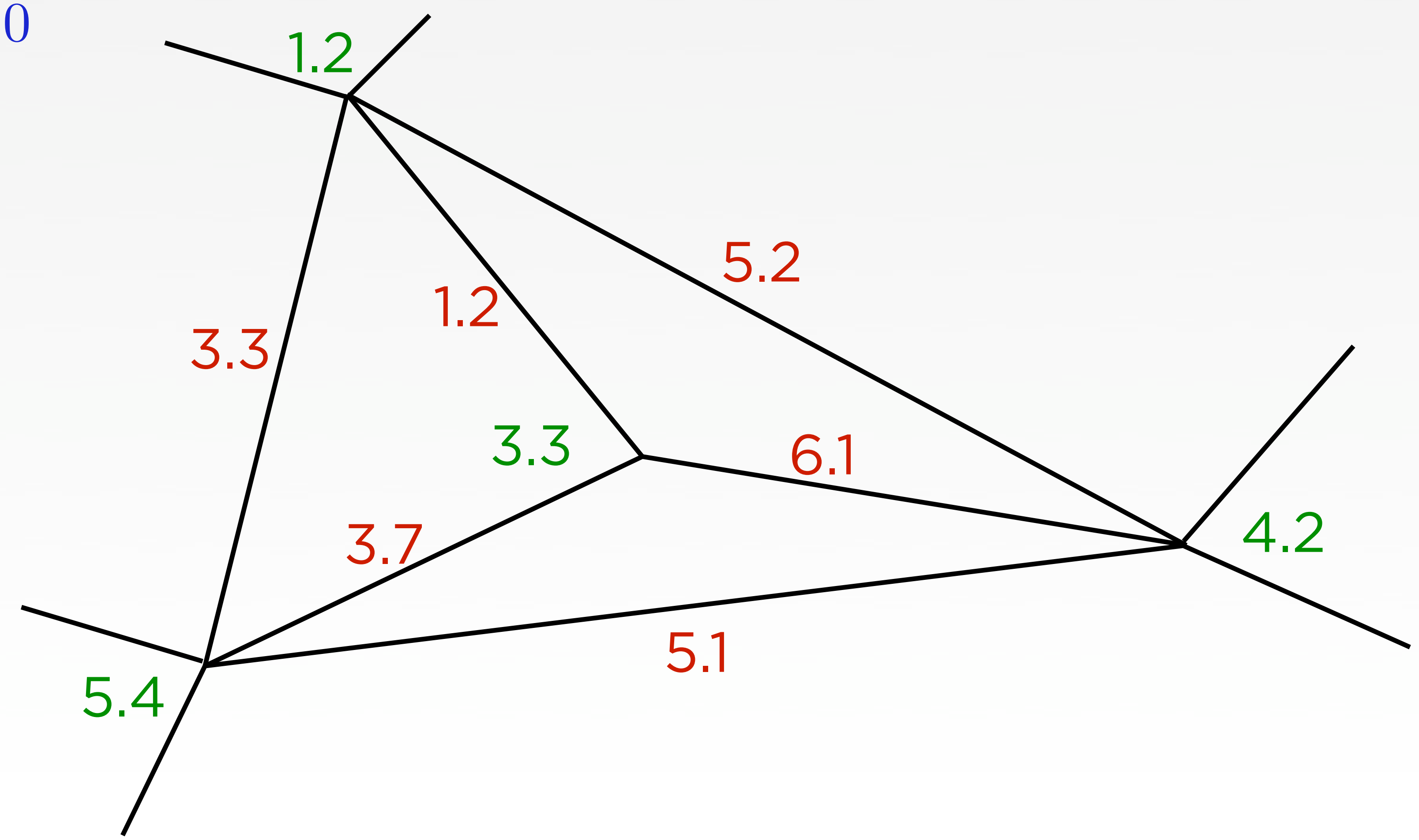
$$\min \sum y_{ij} b_{ij} + \sum y_i b_i$$

$$y_i + \sum_j y_{ij} \geq 1$$

$$y_i, y_{ij} \geq 0$$

Add one node for each y_i with weight b_i

Add one edge for each y_{ij} with weight b_{ij}



Goal: select a (fractional) subset of edges and vertices, so that each vertex has (in total) a weight of 1 selected.

The dual as a graph

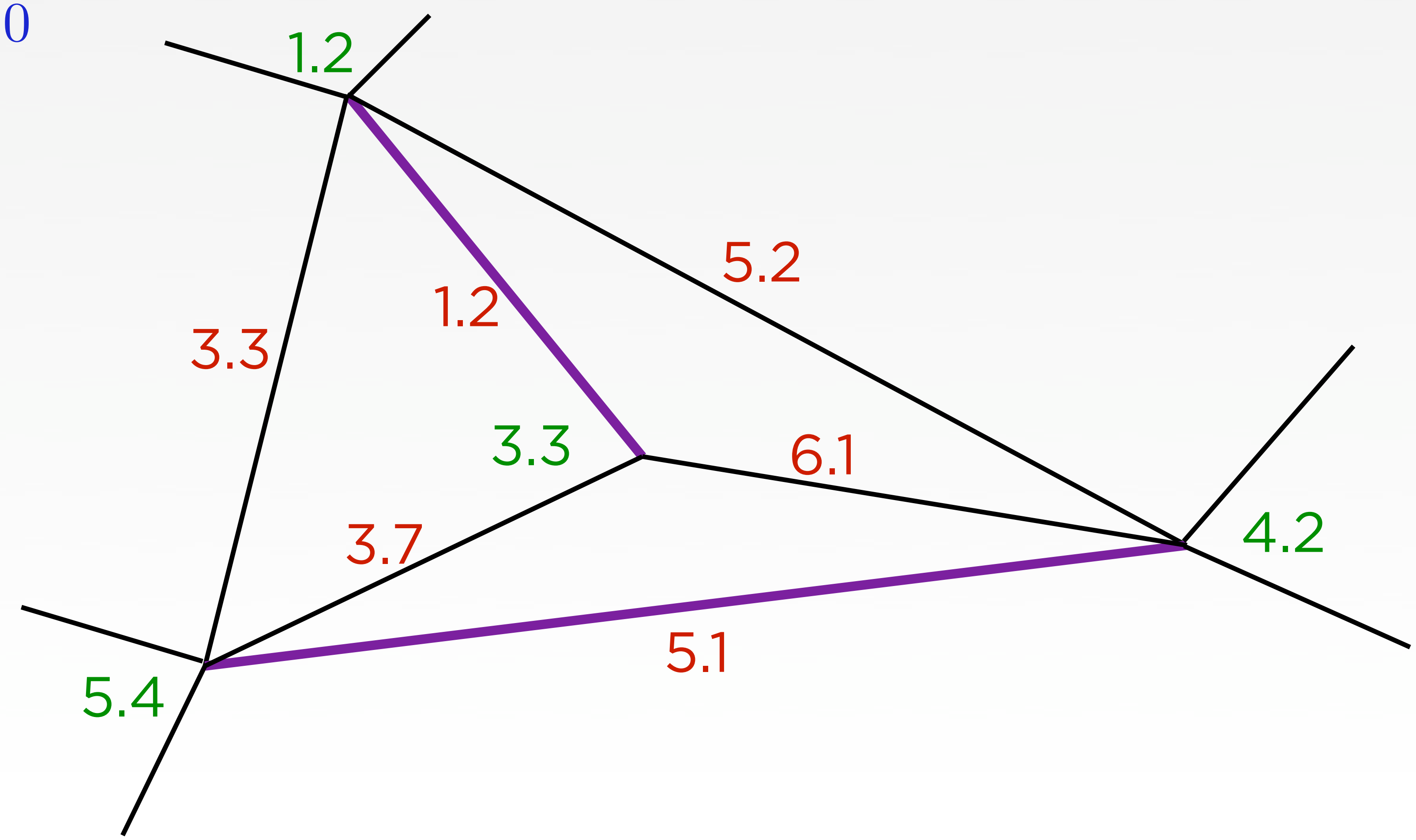
$$\min \sum y_{ij} b_{ij} + \sum y_i b_i$$

$$y_i + \sum_j y_{ij} \geq 1$$

$$y_i, y_{ij} \geq 0$$

Add one node for each y_i with weight b_i

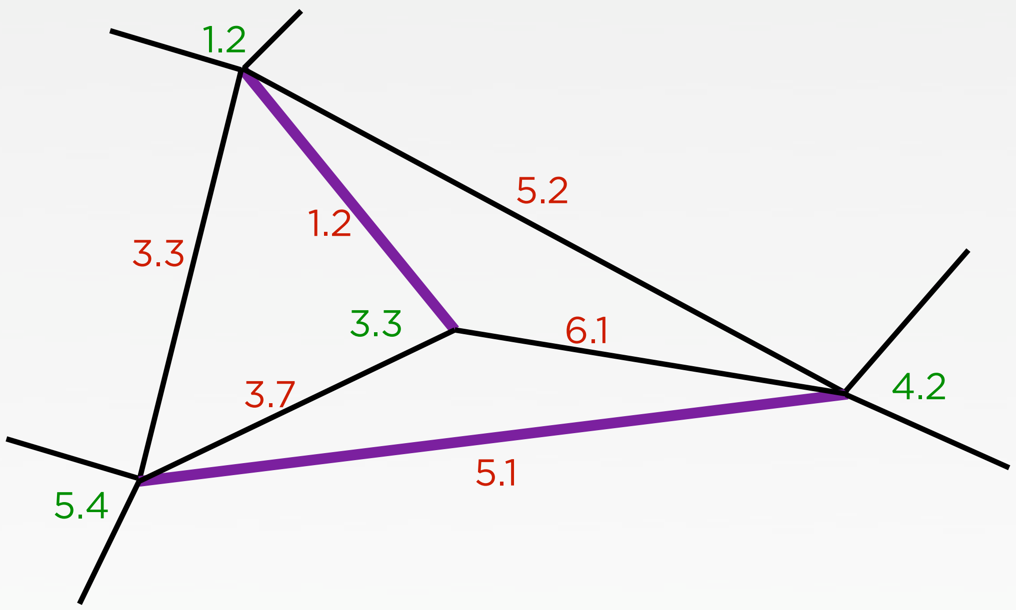
Add one edge for each y_{ij} with weight b_{ij}



Goal: select a (fractional) subset of edges and vertices, so that each vertex has (in total) a weight of 1 selected.

Solving the problem...

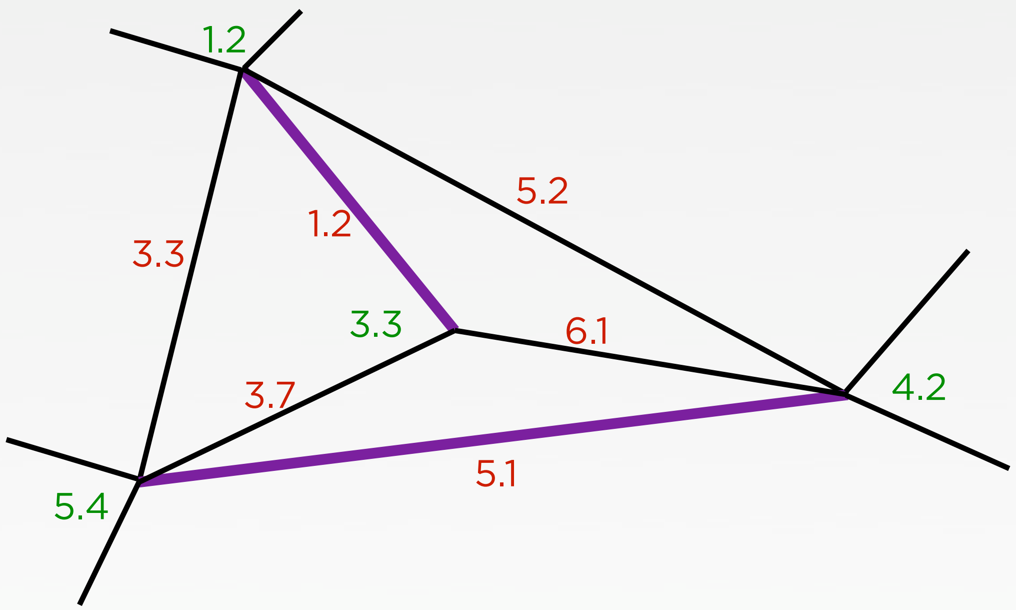
Goal: select a subset of edges and vertices, so that each vertex has a weight of 1 selected.



This looks like the classical edge cover problem only with vertices.

Solving the problem...

Goal: select a subset of edges and vertices, so that each vertex has a weight of 1 selected.



This looks like the classical edge cover problem only with vertices.

We show how to solve this problem by computing min cost matching.

Running time: O(nm)

Checking all combinations: O(n!)

Outline

Introduction to TA

Solving the ‘upper bound’ problem

Empirical Results

Conclusion

Empirical Analysis

Datasets:

- Trec (25M pages), 100k queries
- Yahoo! (16M pages), 10k queries (random subset in each)
  - result caching enabled

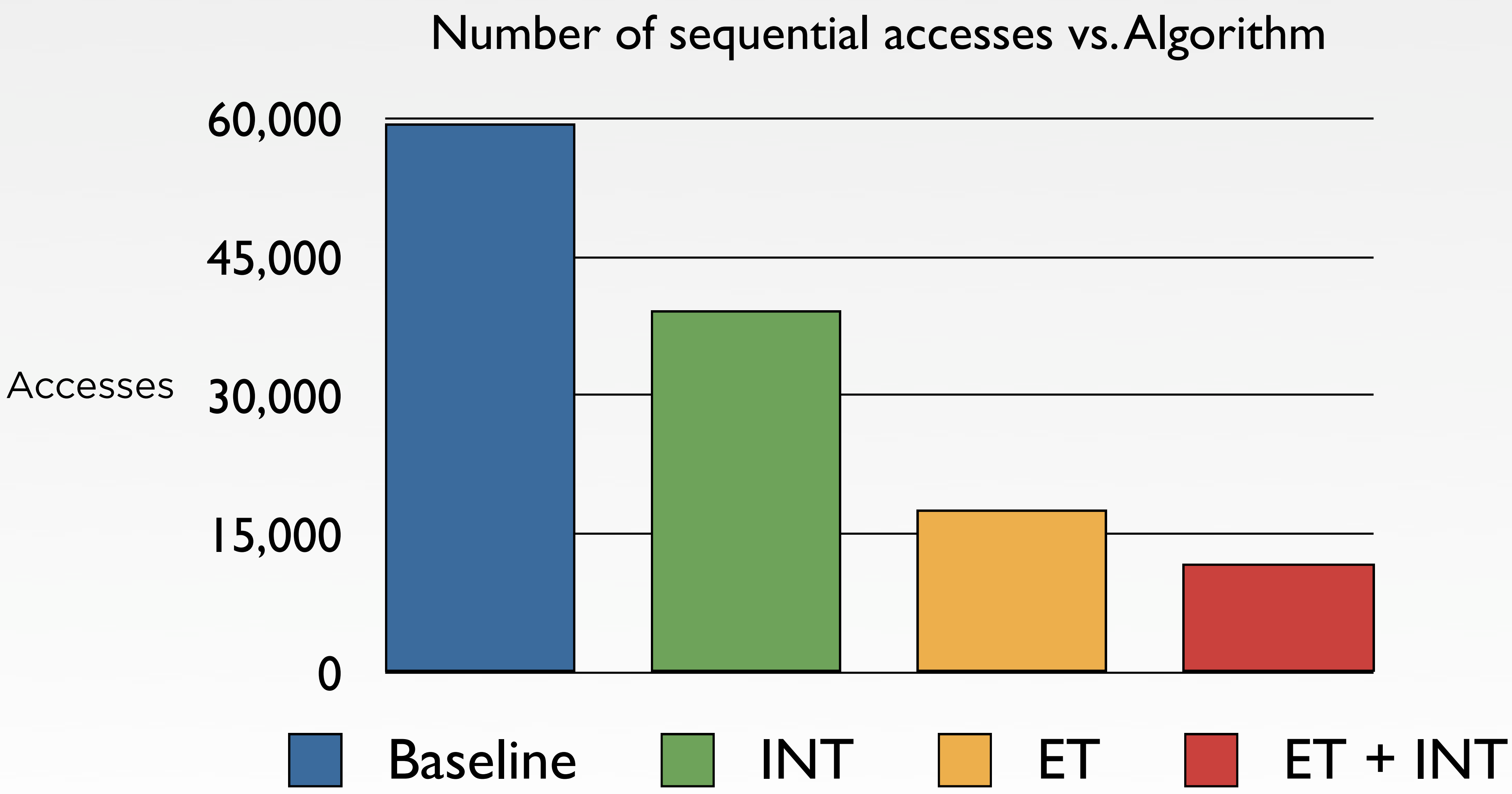
Metrics:

- Number of Random and Sequential Accesses
- Index size

Which bigrams to select?

- Query oblivious manner
- Greedily based on size of intersection versus size of original lists

Empirical Results



Baseline: traverse full list

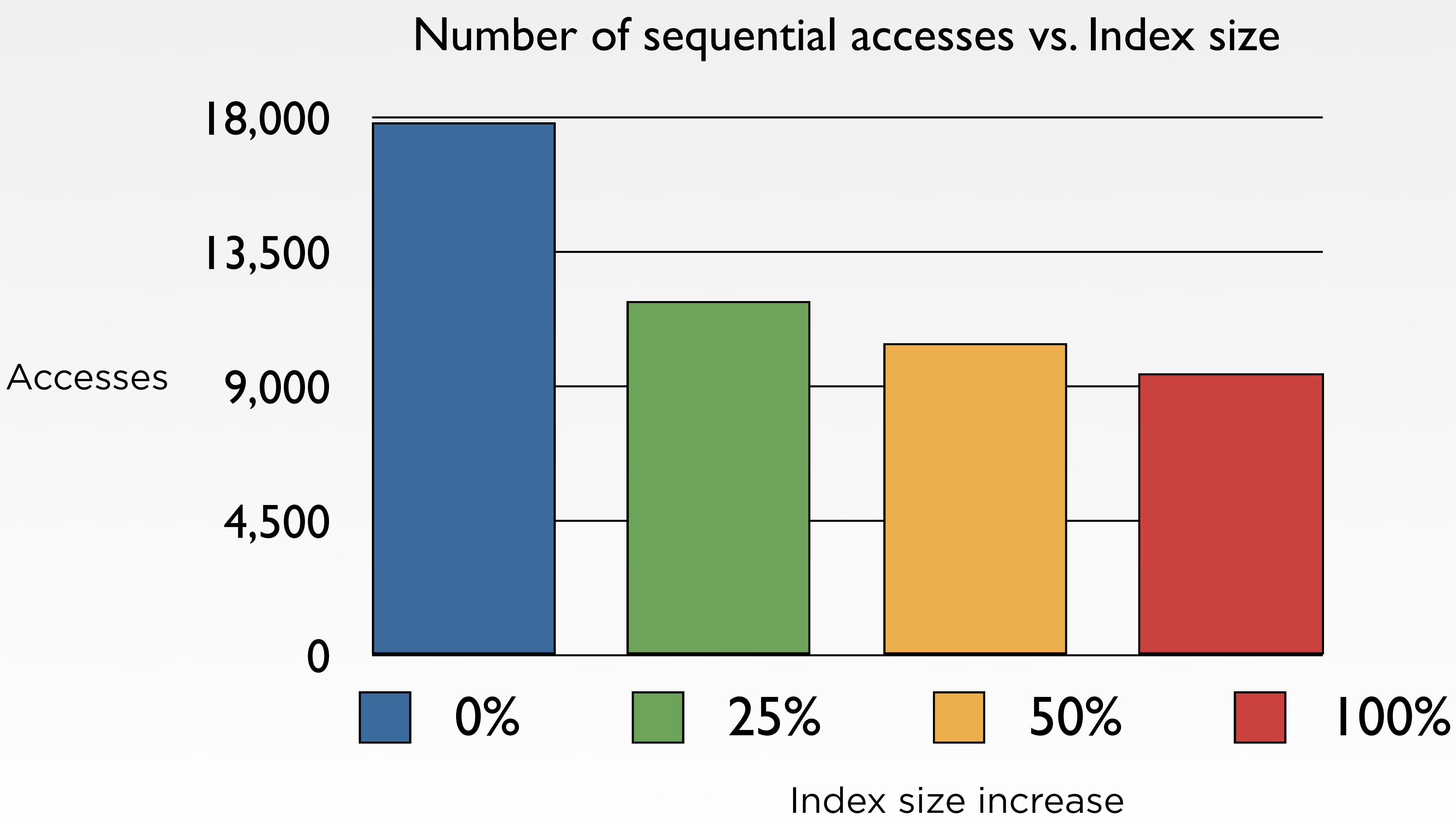
INT: Use intersection lists, but still no Early Termination

ET: Use early termination, but without intersection lists

ET + INT: Use both early termination & intersection lists

Total index growth: 25%

Empirical Results (2)



Immediate benefit, but diminishing returns as extra intersections added.

Results (2)

We prove that in worst case we must examine all of the lists to find the bound. (Otherwise not instance-optimal)

But is this just a theoretical result?

What if you use a simpler heuristics that focus only on intersection lists?

- For 89% of the queries:
  - Average savings 4500 random accesses
- For the 11% of the remaining queries
  - Average cost 127,000 random accesses

So the worst case does occur in practice.

Conclusions

Give a formal analysis of how to use pre-aggregated posting lists

- Solving an LP is unreasonable

Show empirically that a simple selection rule for intersections gives performance improvements.

Many questions remain:

- Extending results to tri-grams (Solving hyperedge cover)
- Better ways of selecting intersections
- …

Thank you