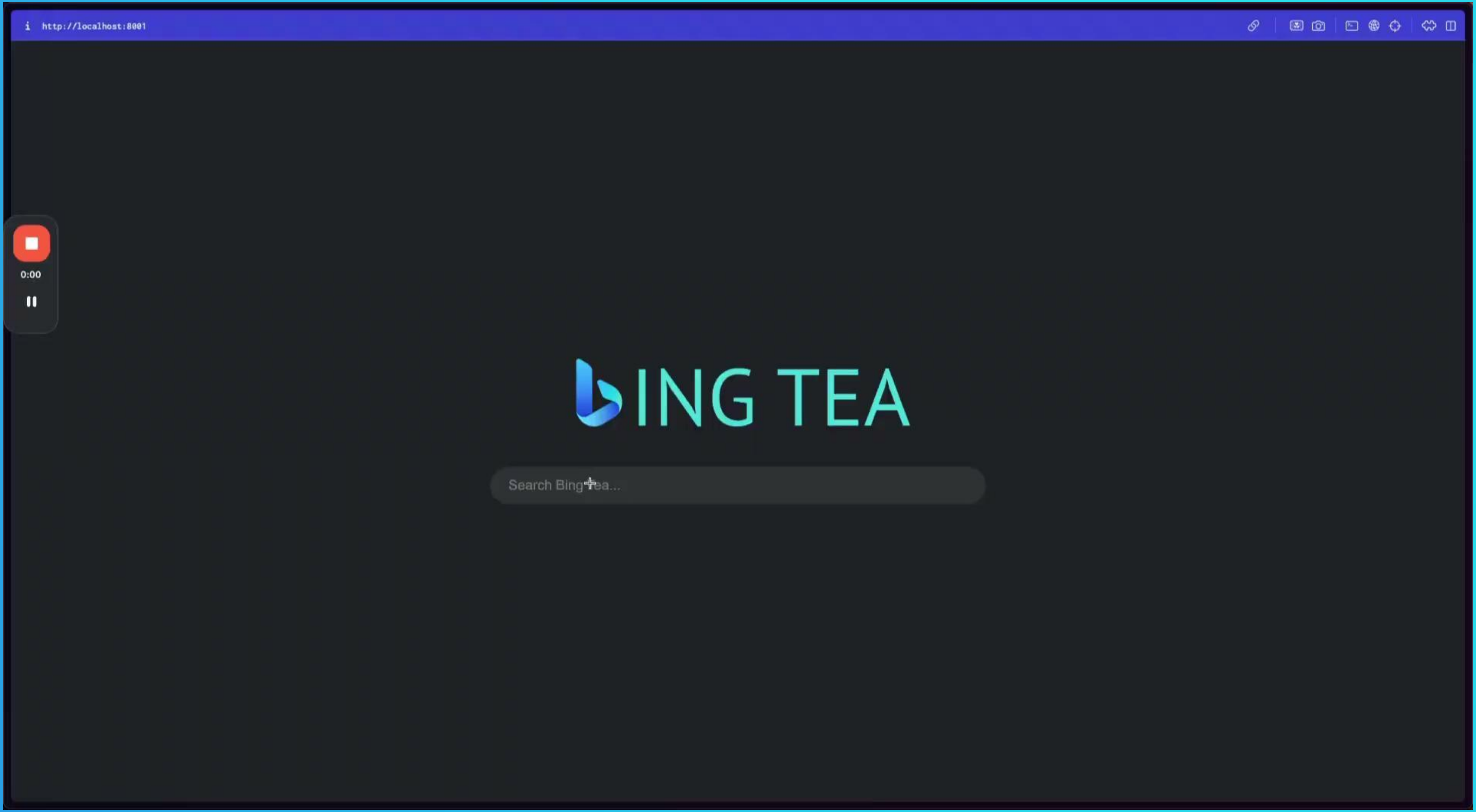
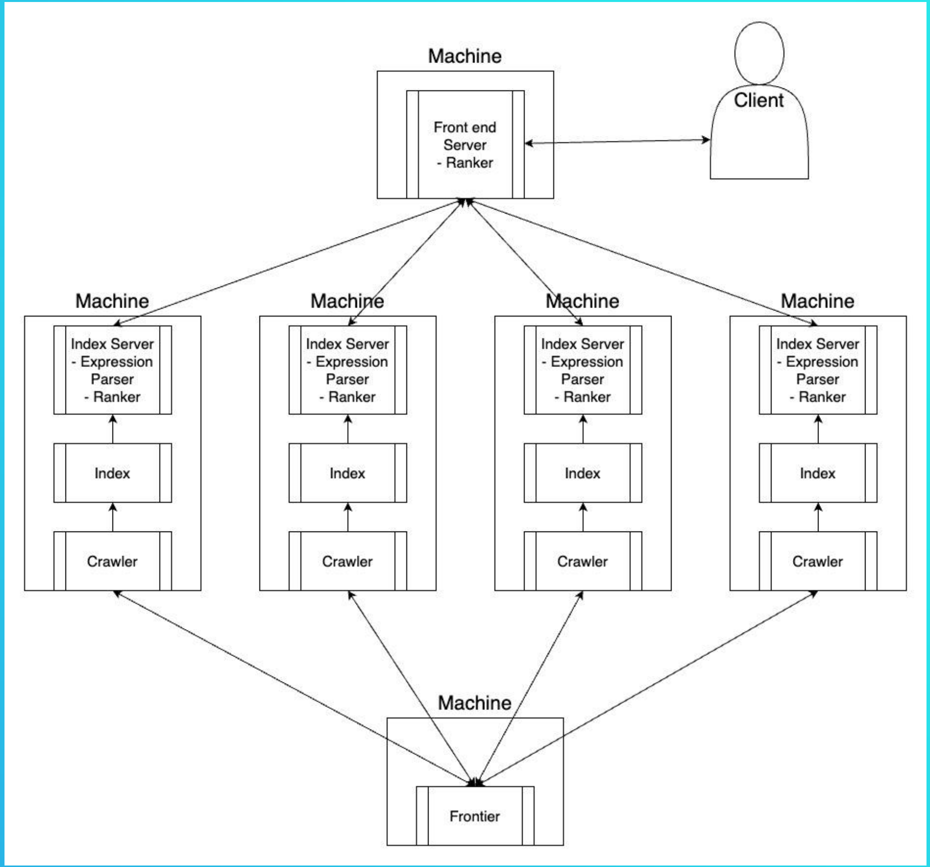
Bing Tea Search Engine

Jimmy Dai:	CSE Junior
Rickey Dong:	CSE Junior
Kevin Li:	CSE Senior
Roee Tsimhoni:	CSE Junior
Matthew Wang:	CSE + Aero Junior
Danny Kim:	CS Junior
Wonbin Jin:	CSE Junior


http://localhost:8001
0:00
BING TEA
Search Bing Tea...

Architecture



Architecture

Key Design Choices:

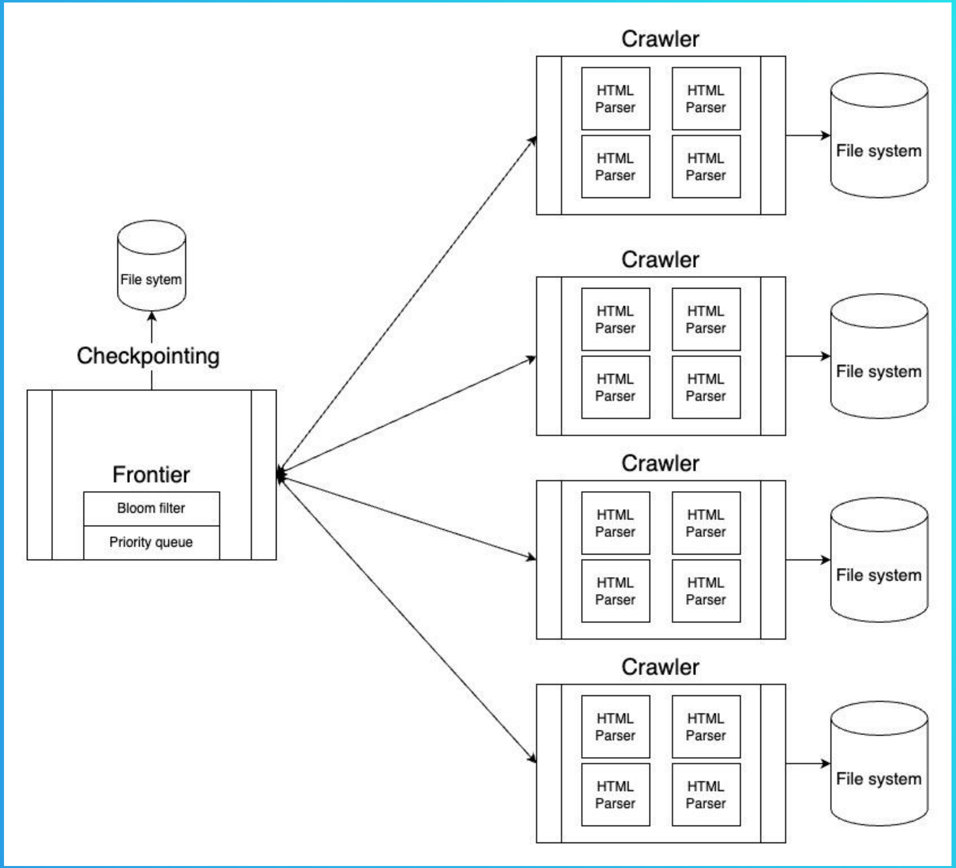
- A “manager” process for the crawler
- No “special” indexes
- Store parsed HTML files
- Closeness from seed-list static ranking
- Two stage ranking

Crawler

Statistics:

Total Pages Crawled	51,618,383
Total Crawl Time	170 hours
Average documents per second	84 documents / second
Peek documents per second	215 documents / second
Average Frontier response time	2.2 ms

Crawler



- Initial assumption was to crawl on laptops (x7)
- Used 22 VMs on GCP instead

- Average Frontier message wait time: 20ms
- Average Frontier message process time: 2ms

- Easy to add and remove machines **during** crawl

Crawler

Problems and Fixes/Potential Improvements:

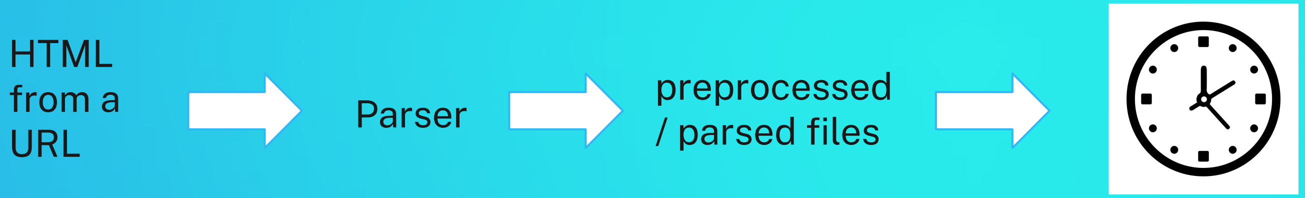
- Being stuck in a certain host → Random url crawling, inversely rank hosts by crawl count
- Document retrieval success rate is low → Retried depending on error, figure out a better retrieval method, rank hosts by success rate
- Duplicate detection → Bloom filter on Frontier, bloom filter on Crawler
- Client-side dynamic content → JS rendering or smart heuristics to discover links behind dynamic content
- Page type reach gaps → Whitelist pdf and video types and extract pdf text & video metadata

HTML Parser

- Parses input html and returns key metrics for ranking and index building
- Collects the following data:
  - Title Words
  - URLs and Anchor text
  - “Emphasized” words (italics, bold, headers)
  - Normal words
  - Number of images
  - Site language
- Does some basic text processing for the index
  - Lowercases all words
  - Removes unnecessary punctuation

Bridge Between Parsing and Indexing

One major design decision we made was to have an explicit phase / “bridge” between parsing and indexing



HTML from a URL

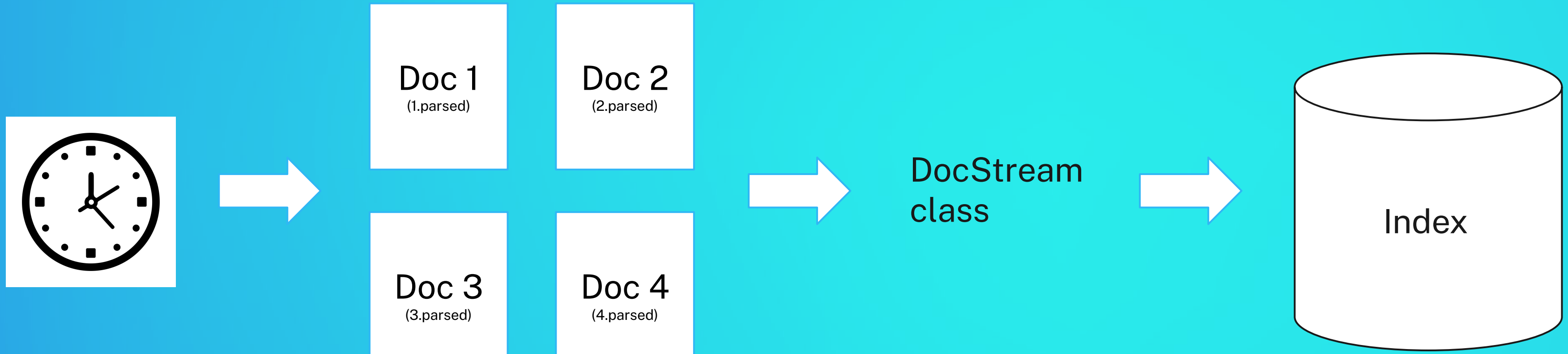
Parser

preprocessed / parsed files

Reasons why this decision was made:

- We wanted to get crawling done ASAP but the index was not yet finished
- We didn’t want the index crashing to affect the crawl
- These intermediate parsed files turned out to be good for snippets

Index



Doc 1
(1.parsed)

Doc 2
(2.parsed)

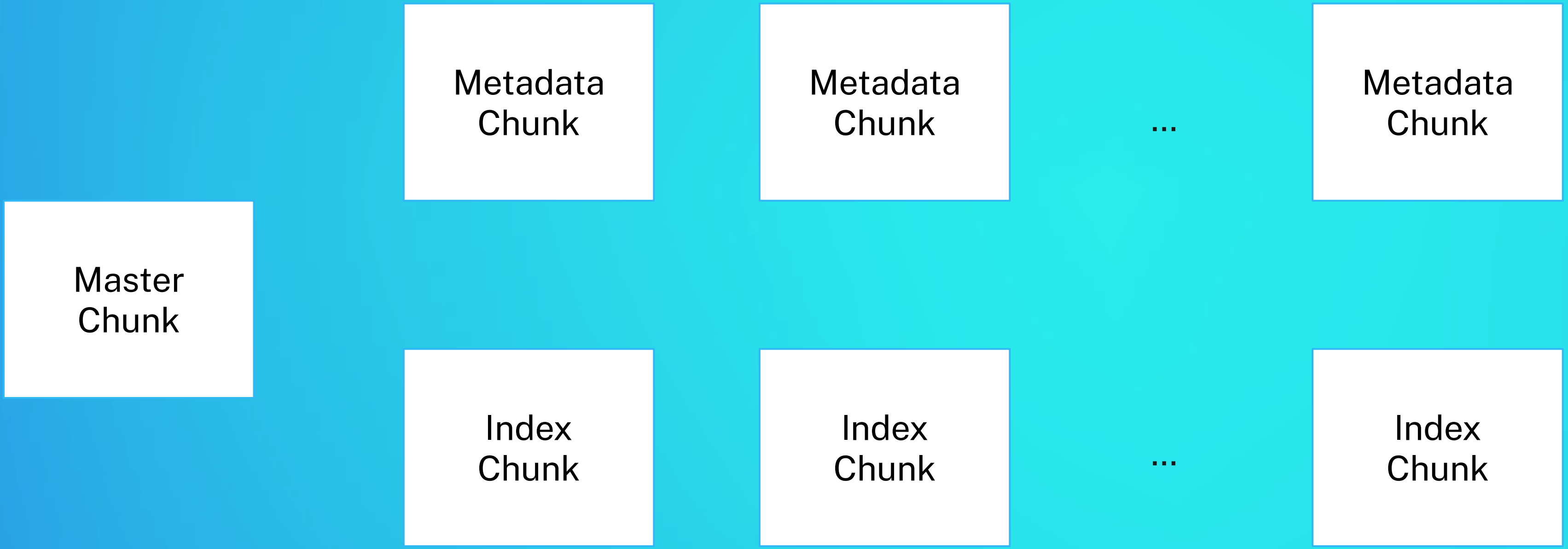
Doc 3
(3.parsed)

Doc 4
(4.parsed)

DocStream
class

Index

Index



Metadata Chunk

Metadata Chunk

…

Metadata Chunk

Master Chunk

Index Chunk

Index Chunk

…

Index Chunk

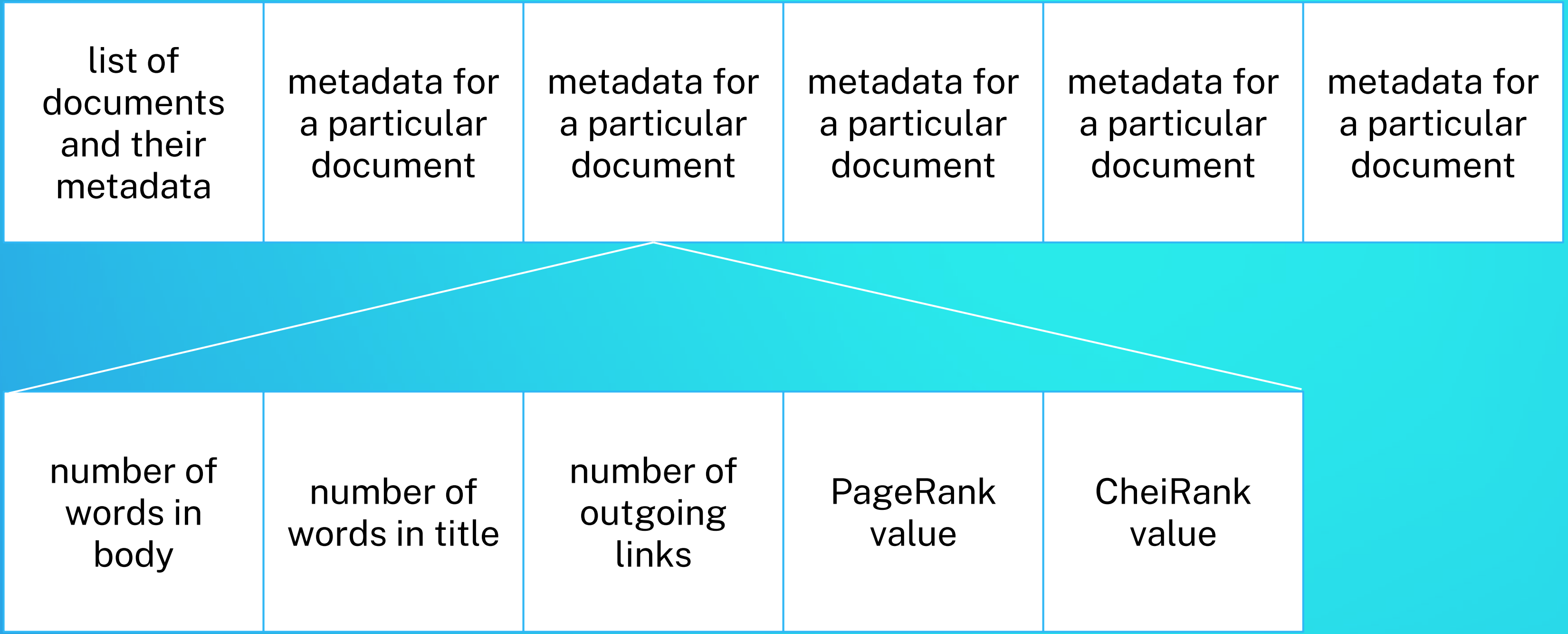
Index

Master Chunk

metadata like # of documents	list of index chunks	list of metadata chunks

Index

Metadata Chunk



list of documents and their metadata

metadata for a particular document

metadata for a particular document

metadata for a particular document

metadata for a particular document

metadata for a particular document

number of words in body

number of words in title

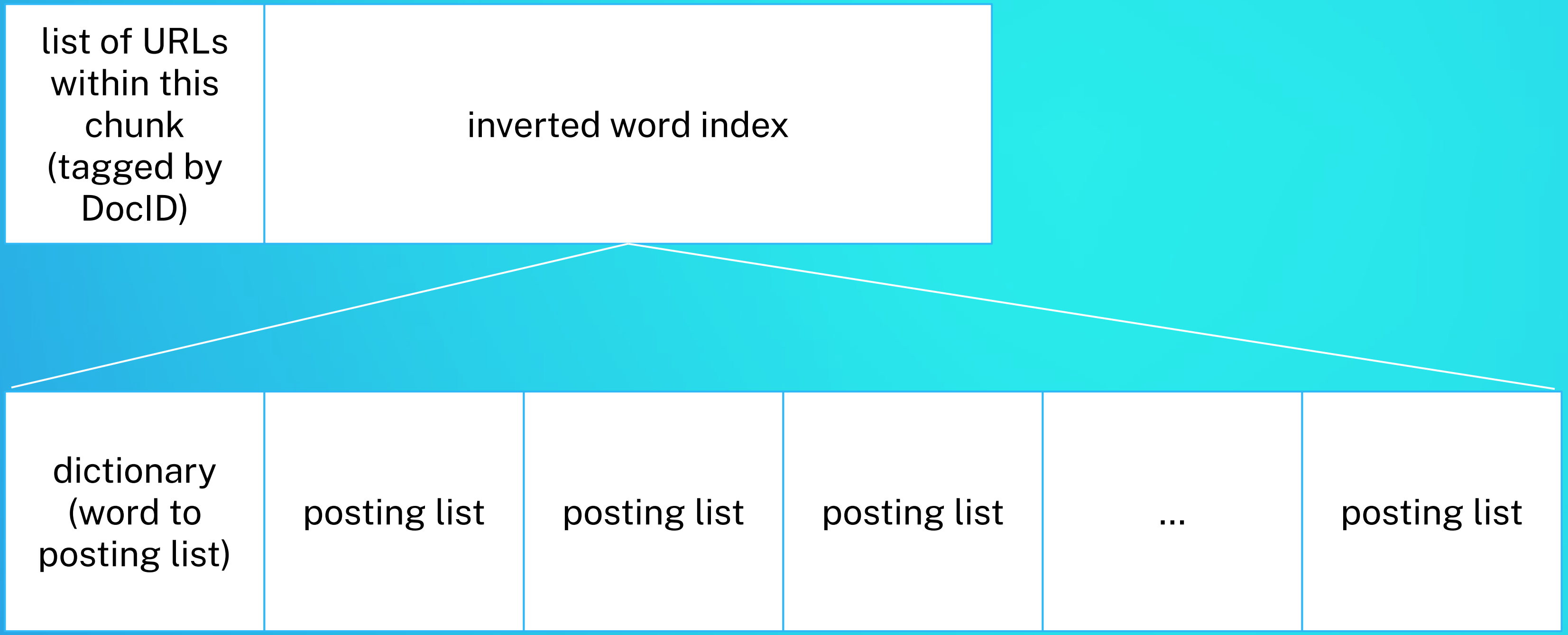
number of outgoing links

PageRank value

CheiRank value

Index

Index Chunk



list of URLs within this chunk (tagged by DocID)

inverted word index

dictionary (word to posting list)

posting list

posting list

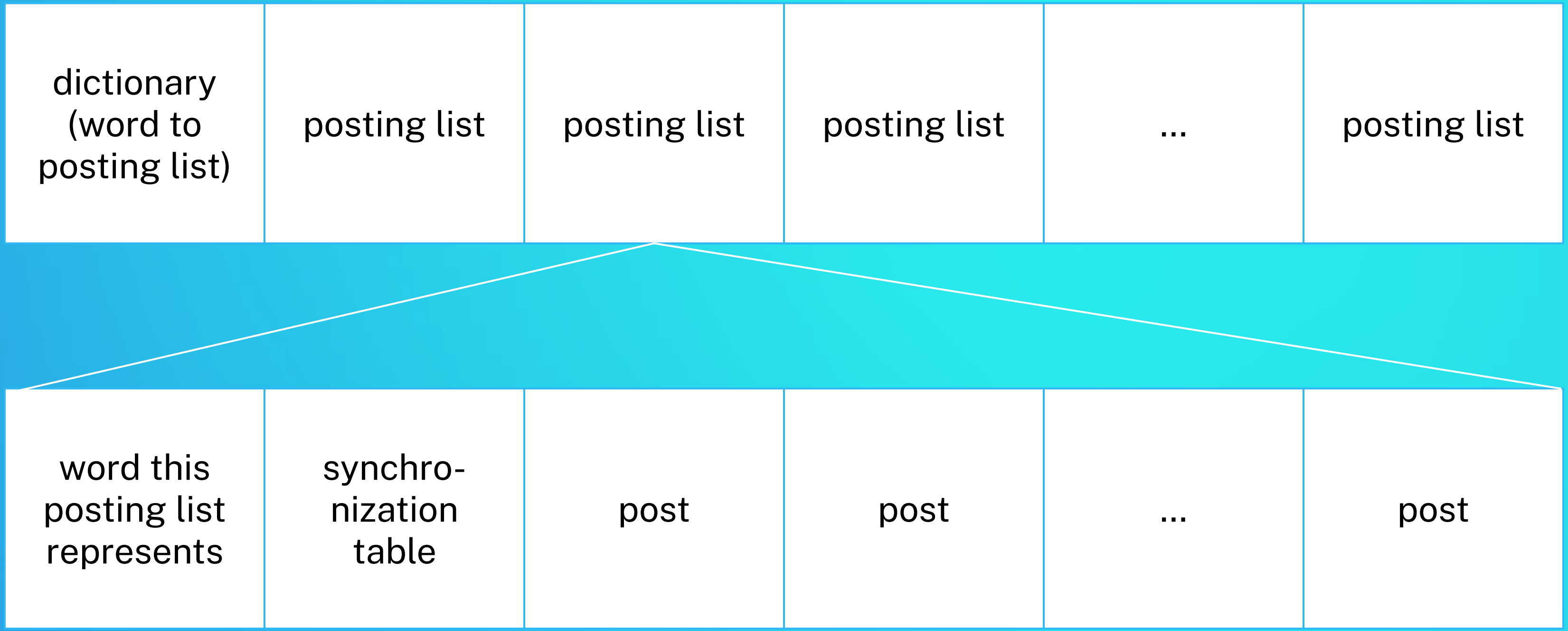
posting list

...

posting list

Index

Posting List



dictionary (word to posting list)

posting list

posting list

posting list

...

posting list

word this posting list represents

synchronization table

post

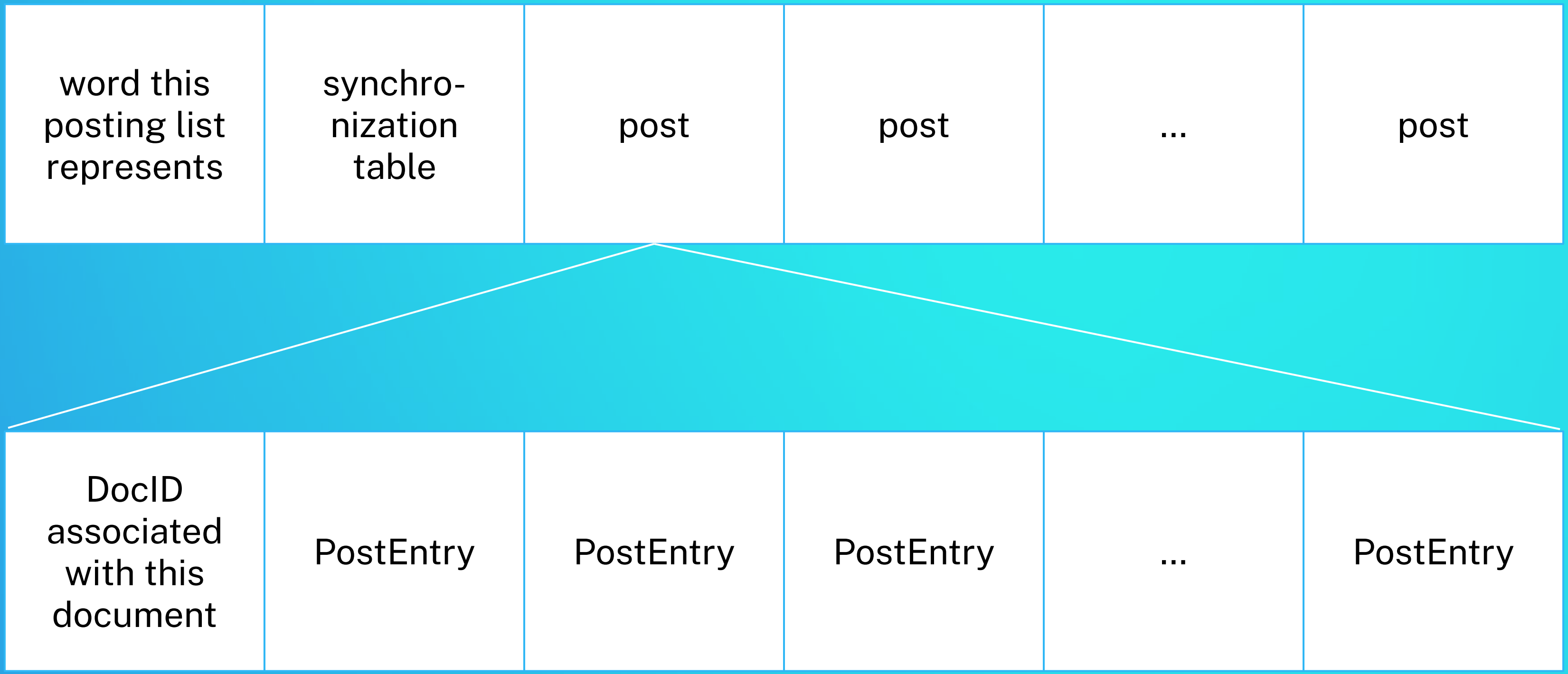
post

...

post

Index

Post

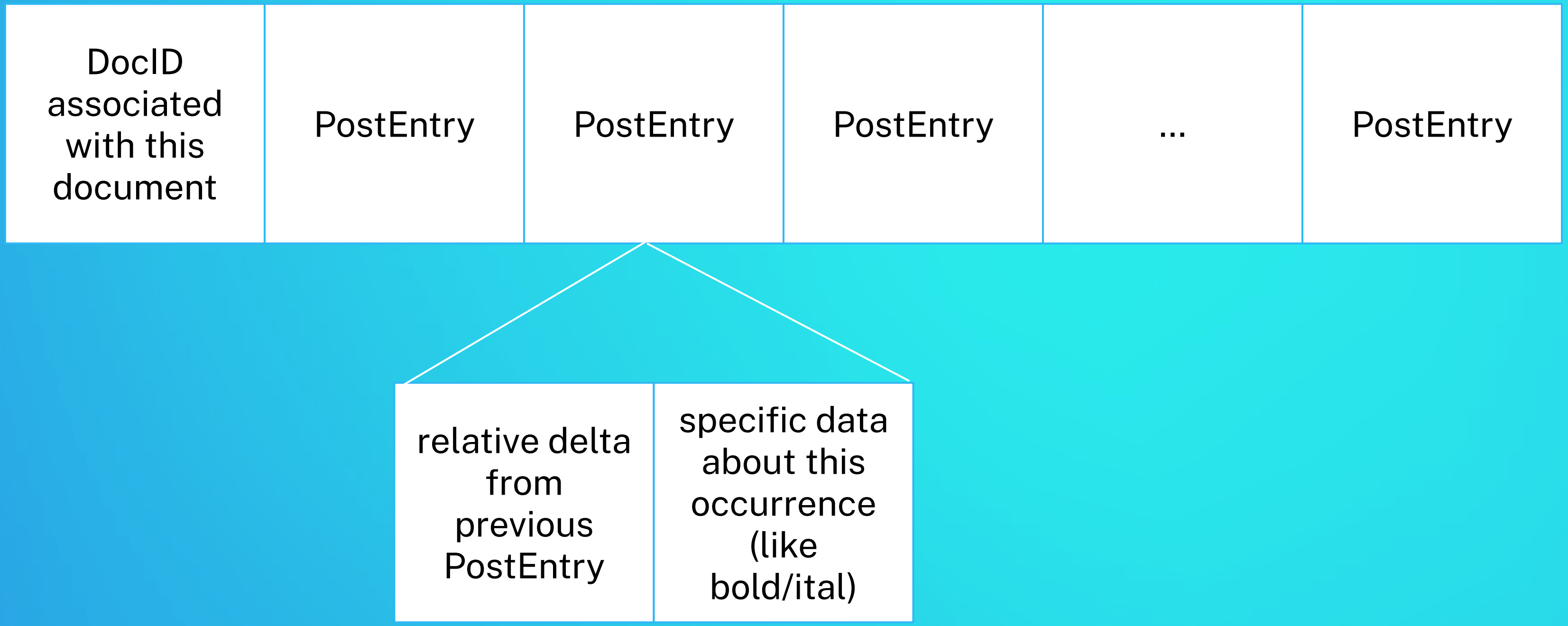


word this posting list represents	synchronization table	post	post	...	post

DocID associated with this document	PostEntry	PostEntry	PostEntry	...	PostEntry

Index

PostEntry



DocID associated with this document

PostEntry

PostEntry

PostEntry

...

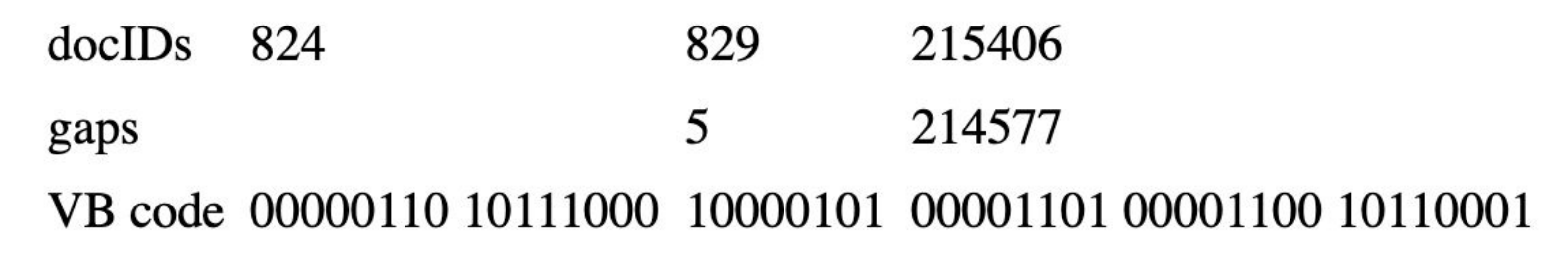
PostEntry

relative delta from previous PostEntry

specific data about this occurrence (like bold/ital)

Compression and Synchronization

- postings lists are compressed with variable byte encoding
- instead of storing absolute locations, relative offsets from the previous entry are stored
- these relative offsets are compressed with vByte encoding (not UTF-8)



docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

- for the ISR functions that Seek() to a particular location, or need to “jump ahead” at a particular spot, instead of moving forward the ISRs one by one an inch at a time…
- we (will) use synchronization points to help scrub the ISRs forward much faster

Query Compiler

- Parses and tokenizes input query
- Uses Top Down Recursive Descent (TDRD) to construct an ISR tree structure
- Supports the following BNF Grammar:
    - **OR**: <Constraint> ::= <BaseConstraint> { <OrOp> <BaseConstraint> }
    - **AND/NOT**: <BaseConstraint> ::= <SimpleConstraint> { [<AndOp>] <SimpleConstraint> } | <SimpleConstraint> <NotOp> <SimpleConstraint>
    - **PHRASE:** <Phrase> ::= '"' { <SearchWord> } '"'
    - **NESTED:** <NestedConstraint> ::= '(' <Constraint> ')'
    - <SimpleConstraint> ::= <Phrase> | <NestedConstraint> | <SearchWord>
- WordISR retrieves posting lists from the index
- Other ISRs (Not/Container, And, Or, Phrase) operate on children ISRs

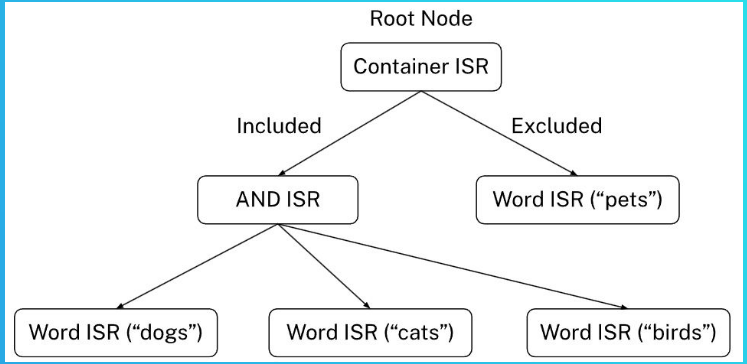
Query Compiler

Example Query: (Dogs Cats Birds) NOT Pets

1. Parse and Tokenize:

   LPAREN, Word(“dogs”), Word(“cats”), Word(“birds”), RPAREN, NOT, Word(“pets”)

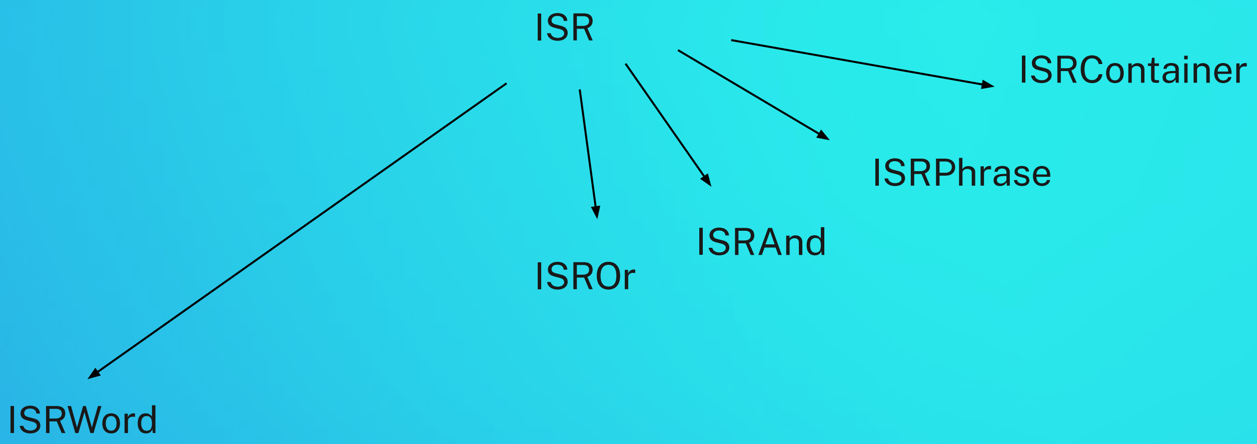
2. Construct expression tree using BNF language
3. Evaluate expression tree and construct ISR tree structure



4. Search using the Root Node of the ISR tree

Index Stream Readers

- provides an interface to read through the index and find occurrences of a particular term
- Next(), NextDocument(), Seek()

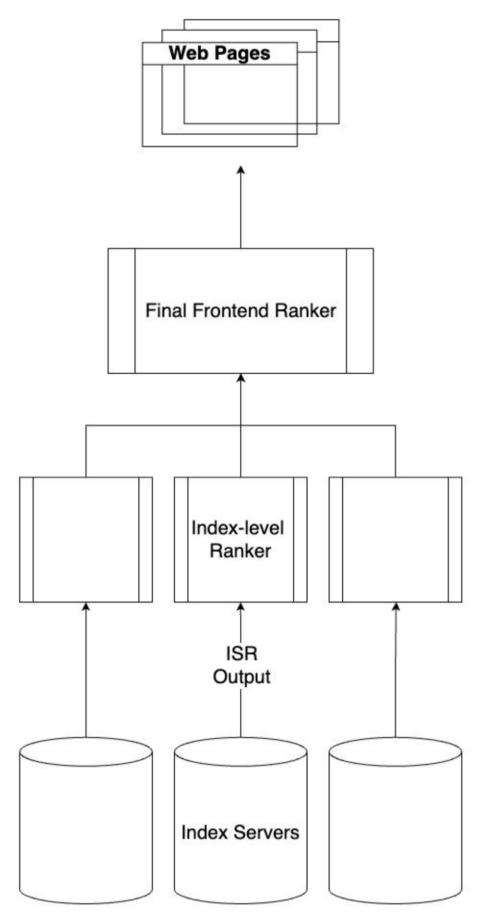


Ranker

Architecture

- ML-based ranker using XGBoost
  - Data (Pages) described through heuristic metadata scores
  - Ranker selects outputs (pages) with the top 10 ranks

- 2-stage ranking system
  1. Index-level ranker (Heuristic score)
  2. Final Frontend ranker (Predicted score)



Ranker

Heuristics

Pages heuristically “scored” based on page metadata

Static Ranking:

- Document Length
- Title Length
- Page Rank
- CheiRank
- Number of Images
- Number of Outgoing Links
- Domain

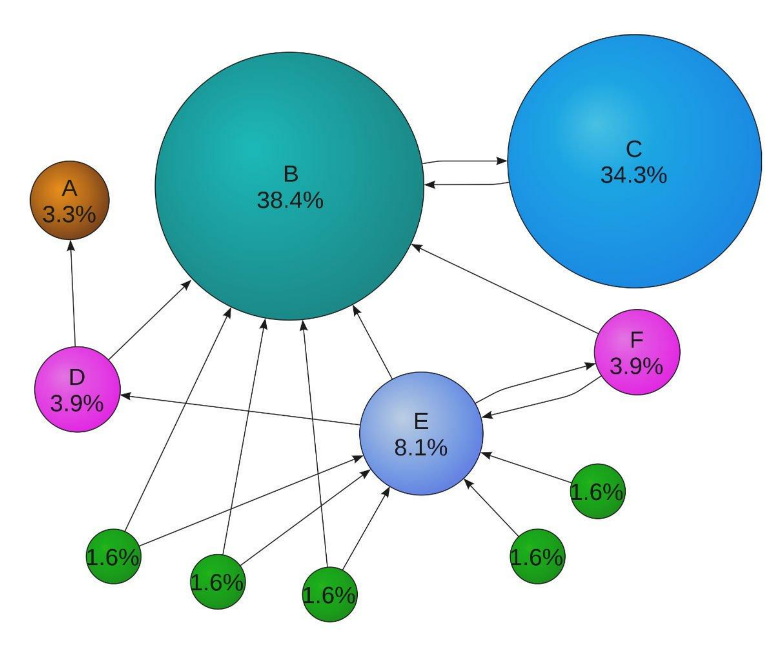
Dynamic Ranking:

- Weighted sum score based on query match locations
  - Title matches have higher weight compared to body matches

Ranker

PageRank, CheiRank, and Louvain

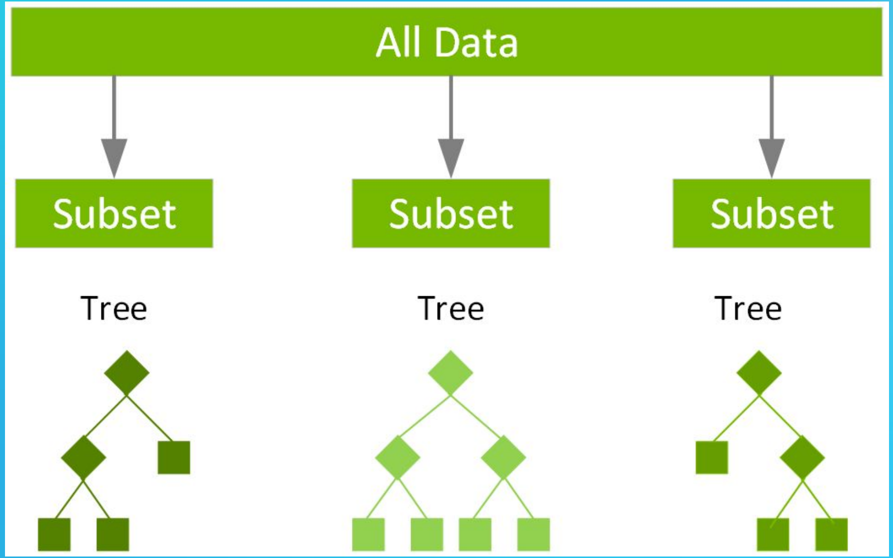
- Network-based
- Consider link structure and between-site relationships.
- PageRank highlights popular nodes, while CheiRank highlights communicative nodes.
- Louvain detects communities by greedily maximizing modularity.
- Communities likely share topics and have similar goodness.



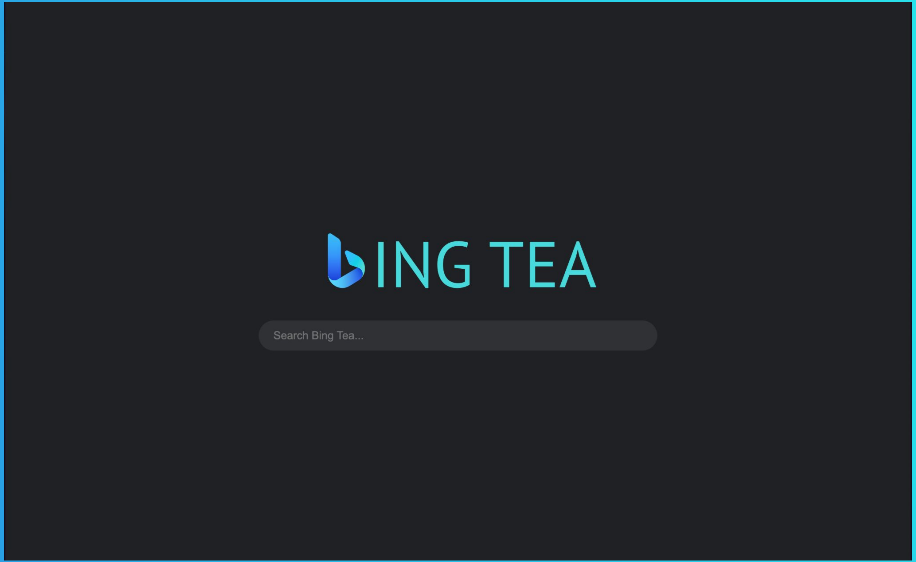
Ranker

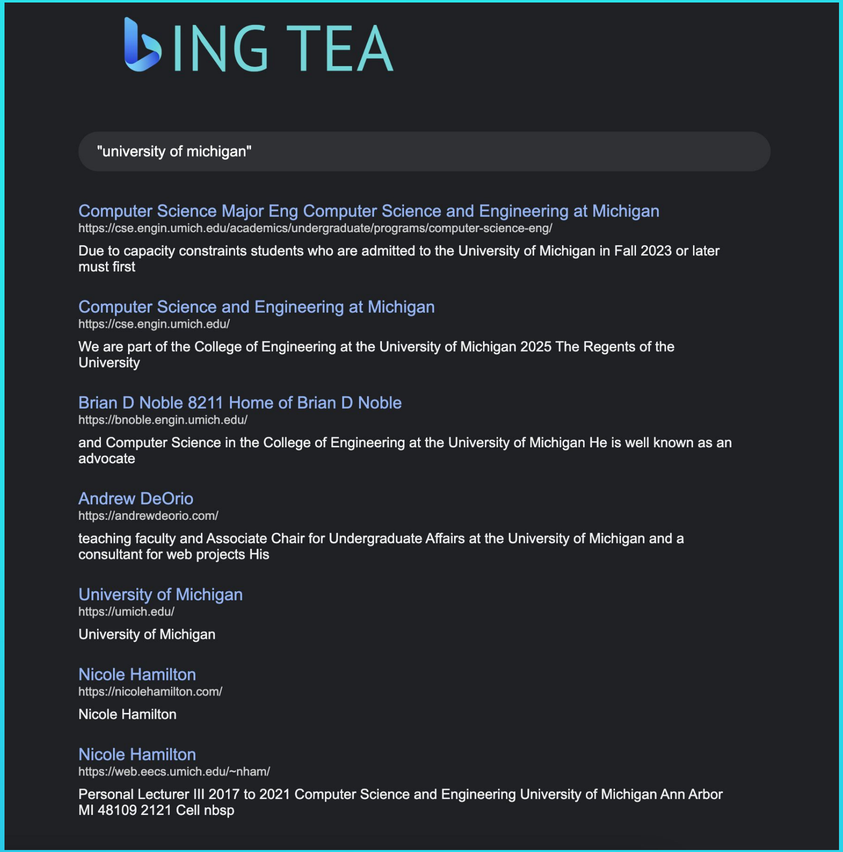
XGBoost

- Instead of a linear model, we use a more advanced model to extract more nuance.
- eXtreme Gradient Boosting (XGBoost) is a gradient boosted decision tree model.
- It trains quickly and accurately, and is commonly used in data science competitions.
- Often needs less data and faster to train than neural networks, while performing better.



Frontend





"university of michigan"

Computer Science Major Eng Computer Science and Engineering at Michigan
https://cse.engin.umich.edu/academics/undergraduate/programs/computer-science-eng/
Due to capacity constraints students who are admitted to the University of Michigan in Fall 2023 or later must first

Computer Science and Engineering at Michigan
https://cse.engin.umich.edu/
We are part of the College of Engineering at the University of Michigan 2025 The Regents of the University

Brian D Noble 8211 Home of Brian D Noble
https://bnoble.engin.umich.edu/
and Computer Science in the College of Engineering at the University of Michigan He is well known as an advocate

Andrew DeOrio
https://andrewdeorio.com/
teaching faculty and Associate Chair for Undergraduate Affairs at the University of Michigan and a consultant for web projects His

University of Michigan
https://umich.edu/
University of Michigan

Nicole Hamilton
https://nicolehamilton.com/
Nicole Hamilton

Nicole Hamilton
https://web.eecs.umich.edu/~nham/
Personal Lecturer III 2017 to 2021 Computer Science and Engineering University of Michigan Ann Arbor MI 48109 2121 Cell nbsp