DISCO

Engineering Challenges in Legal Technology

Anthony Cassandra, Ph.D.

Director of Engineering

February 1, 2019

Outline

- Who am I?
- Where am I?
- What do I do?
- What are my problems?
- What are your questions?

Who am I?

Education

University of Buffalo	*Nothing*	
Suffolk County Community College	A.S.	
Stony Brook University	B.S.	
Brown University	M.Sc.	
Brown University	Ph.D.	

Work Experience

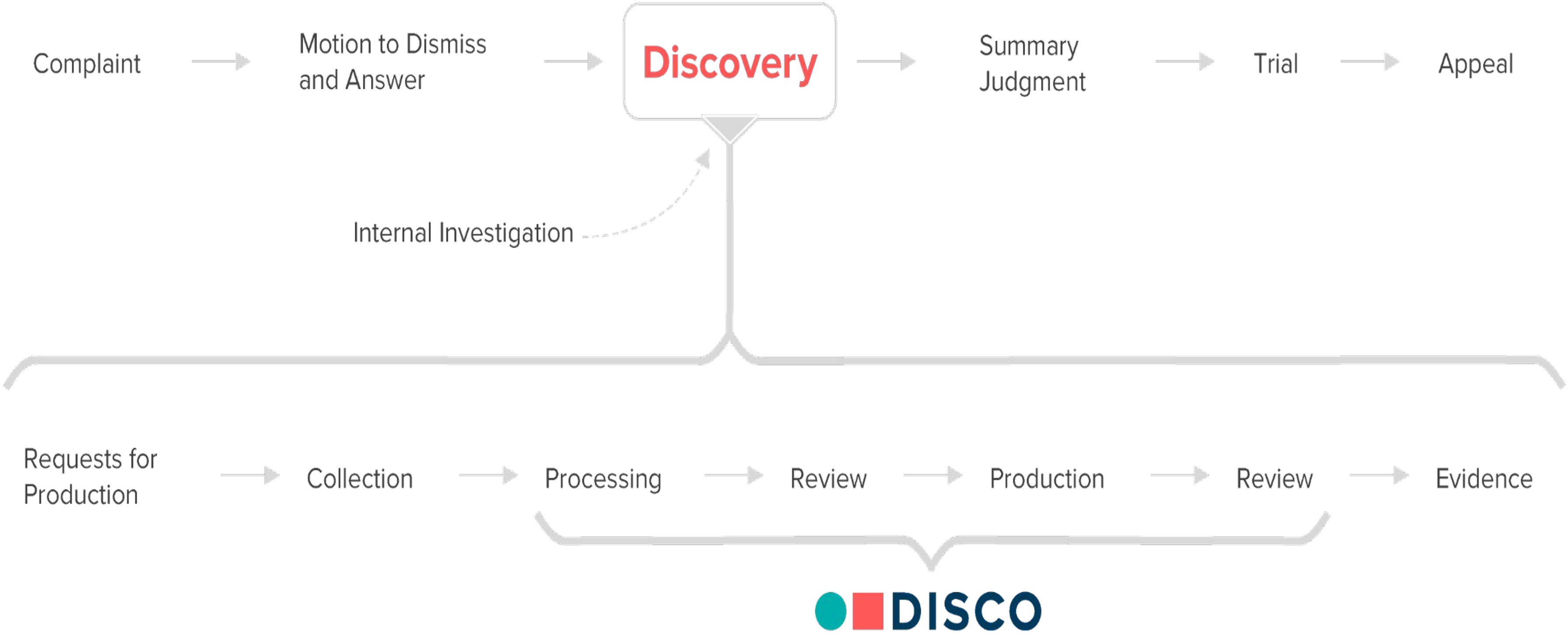
machinist, inspector	factory	7+ yrs.	LR
researcher	industrial lab	3 yrs.	MCC
founder, engineer, director	failed startup	2 yrs.	enetica
adjunct professor	large university	1 yr.	
engineer, researcher	industrial lab	1 yr.	Telcordia
assistant professor	small university	3 yrs.	SEU
researcher	academic lab	1 yr.	
founder, engineer, CTO	successful startup	8 yrs.	pronto
engineer, architect, manager	medium-sized company	3 yrs.	bazaarvoice
engineer, architect, director	successful startup	2 yrs	DISCO

Where am I?

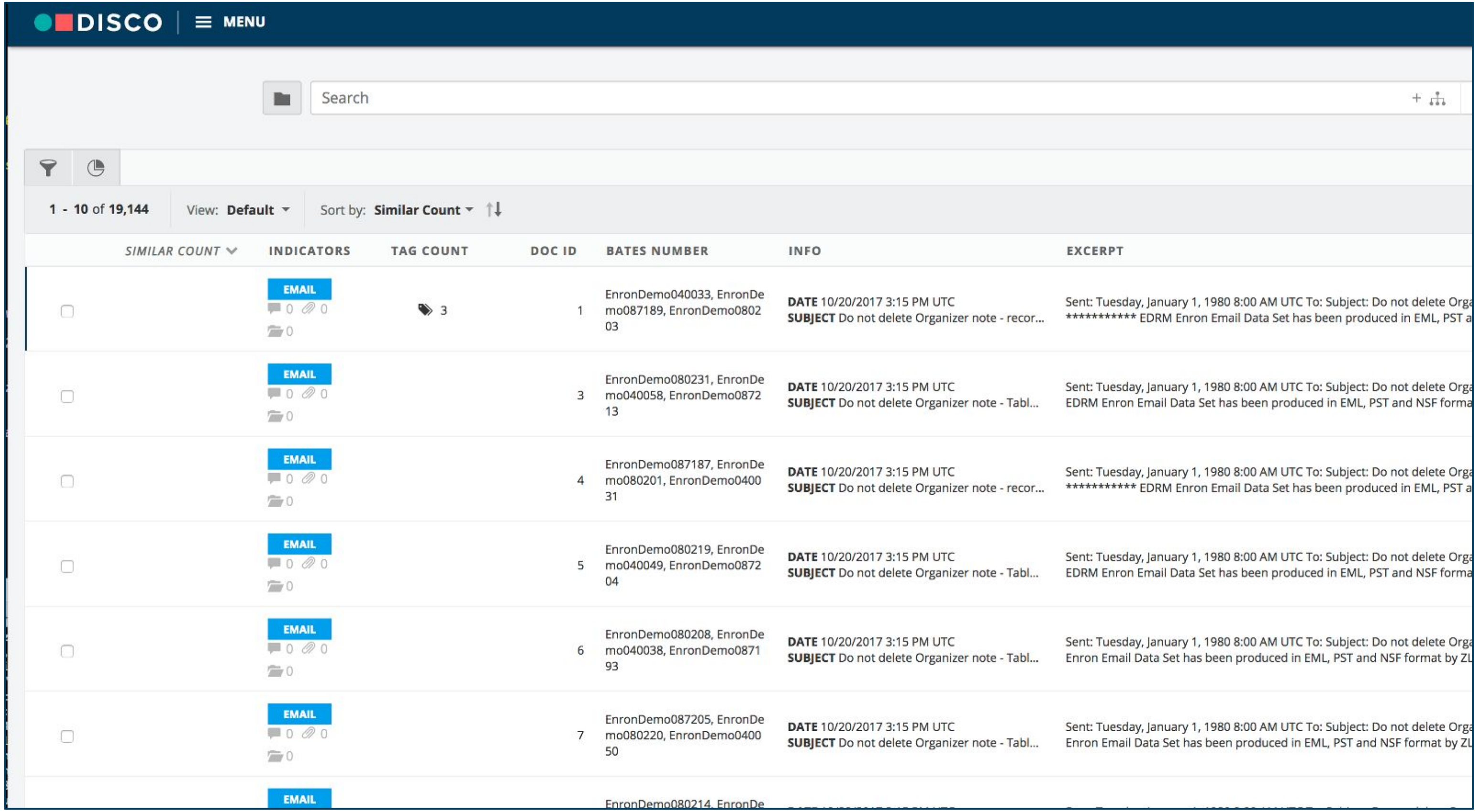
Current Company

- CS DISCO, Inc.
- Legal Technology
- e-Discovery Product
- Cloud-based
- Recent funding round.
    - Expanding to support more of the legal processes.

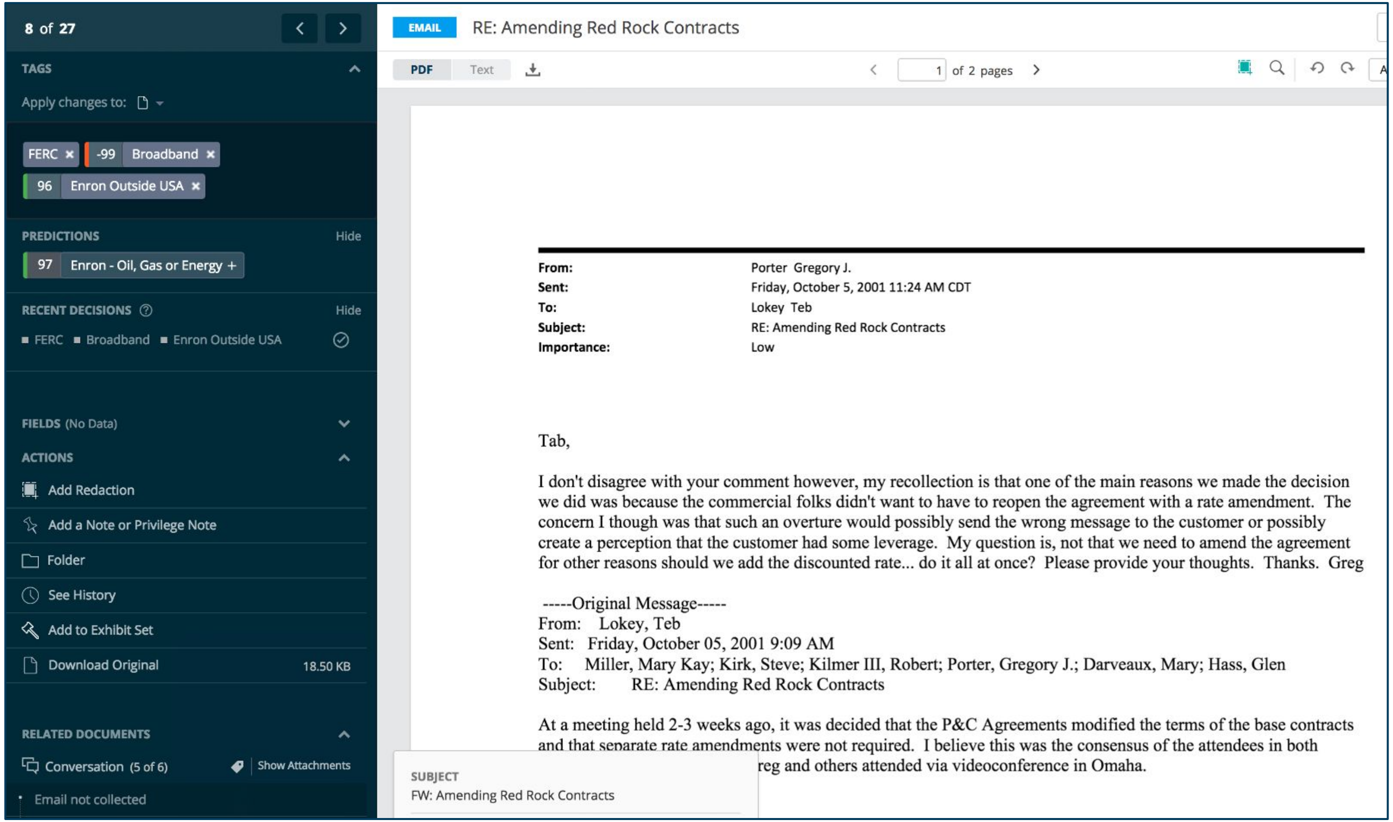
The Legal Discovery Process



Searching During e-Discovery



Document Reviewing and Tagging



What do I do?

Platform Team Name: Atlas

- **AWS Aurora**: main system of record (SoR) for documents:
  - > 1 billion documents (and growing); and
  - > 125 attributes (+ text) per document (and growing).
- **Elasticsearch**: search is the “heart” of our product:
  - > 140 data nodes (across 4 clusters); and
  - > 150 TB storage across all indices (and growing).
- **Services / APIs**: for insert, update, lookup and search, etc.

Sample of Our Technologies

- Python / C# / Bash
- AWS: Aurora/RDS, S3, ECS, ECR, SQS, Lambda, EC2
- Elasticsearch, Redis, Consul, Celery, Flask, Git
- Datadog, Kibana, Packetbeat, Filebeat, Logstash, Logspout
- Docker, Jenkins, Terraform, Code Pipeline, OpsWorks

Everyone on the team has to be familiar with all of these.

Unique Domain Problems

- Previous retail product search experience:
  - tens of millions of users;
  - simple queries; with
  - low data durability and recall requirements.
- Current legal search experience:
  - thousands of users;
  - highly sophisticated and very complex queries; with
  - very high data durability and recall requirements.

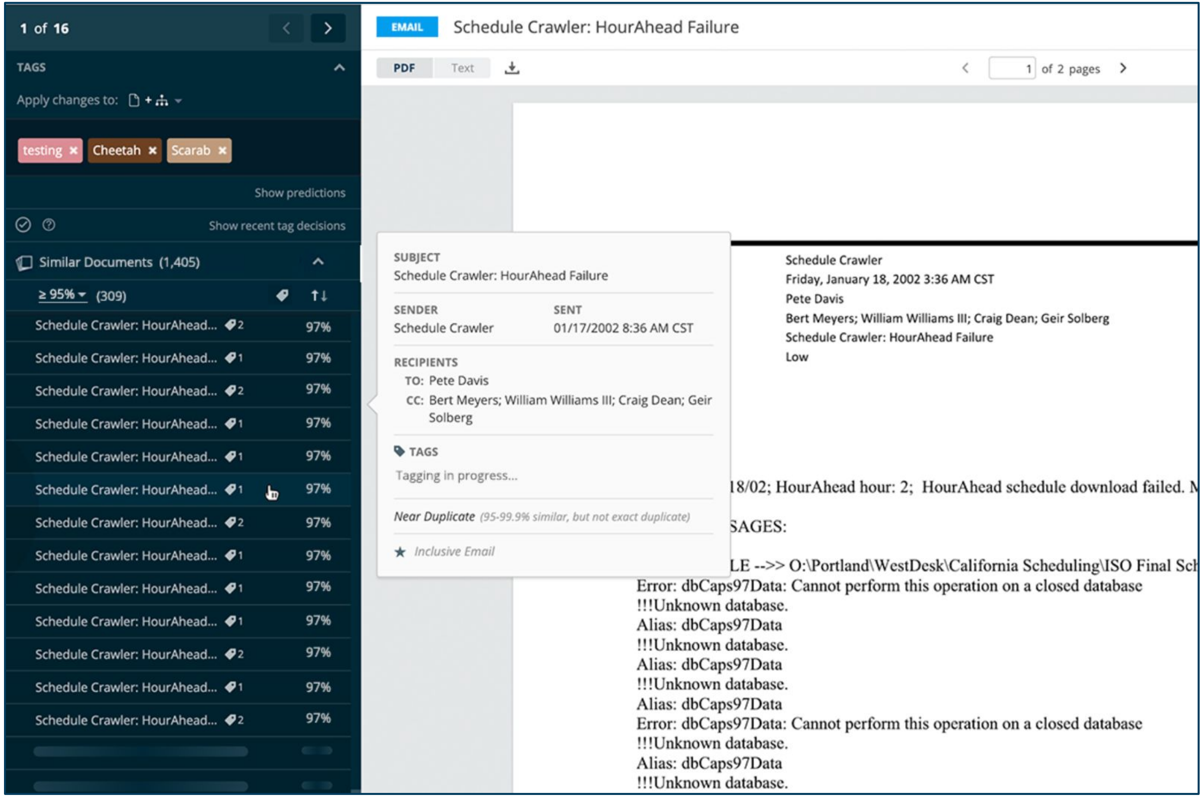
Team's Technical Challenges

- Underlying data model has reached its limits.
- Redesigning infrastructure to scale further.
- Continue to support new features of the review product.
- Supporting new products.
- Scaling team, organization and operations.

These are typical for a startup that has survived.

What are my problems?

Similar Documents Feature



The Similar Documents Project

- Original quick-n-dirty, start-up version:
  - poor quality; and
  - does not scale.
- Replace with a better version:
  - shingling;
  - min-hash; and
  - locally sensitive hashing (LSH).

Base Requirements

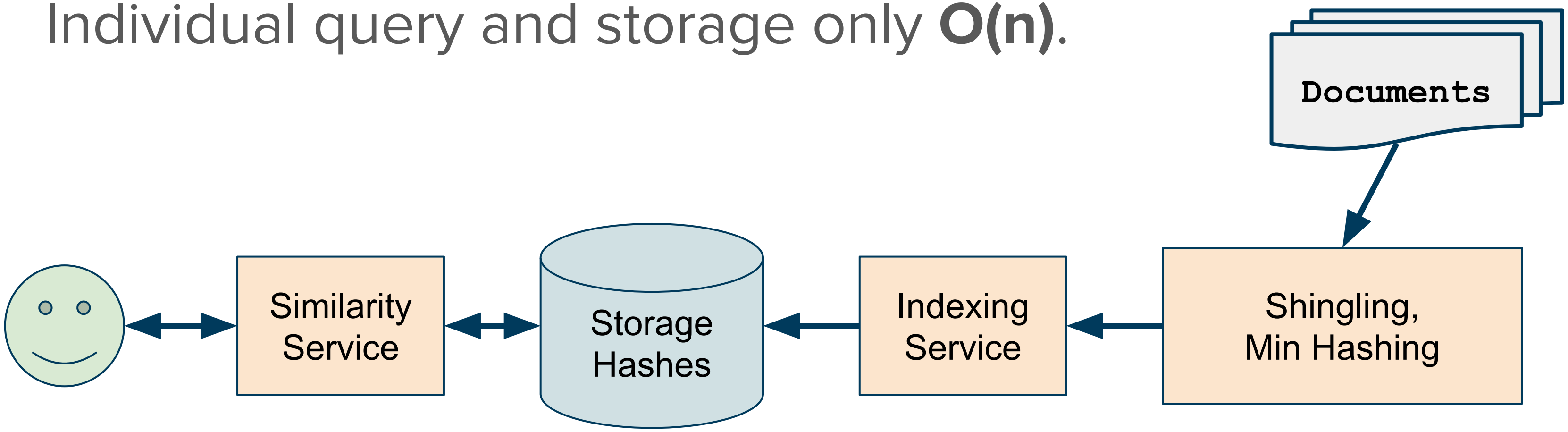
- Pagination and sorting requirements.
- Strong *(< 100ms)* service level agreements (SLAs).

Design Challenge

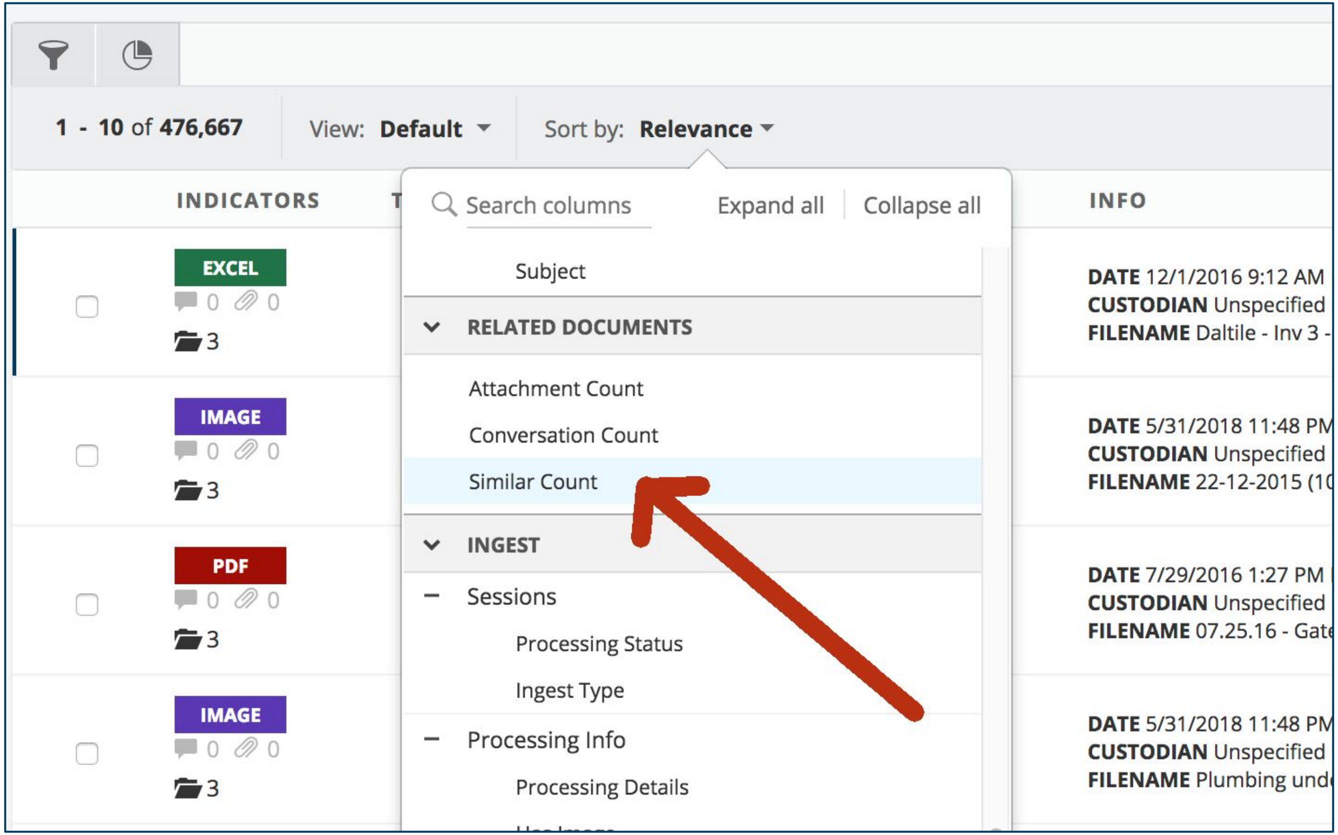
- Leaves us with a problem requiring $\mathbf{O(n^2)}$ comparisons.
- We can have **n** = tens of millions of documents.
- Our CS training tells us to avoid anything $\mathbf{O(n^2)}$.

The “Right” Solution

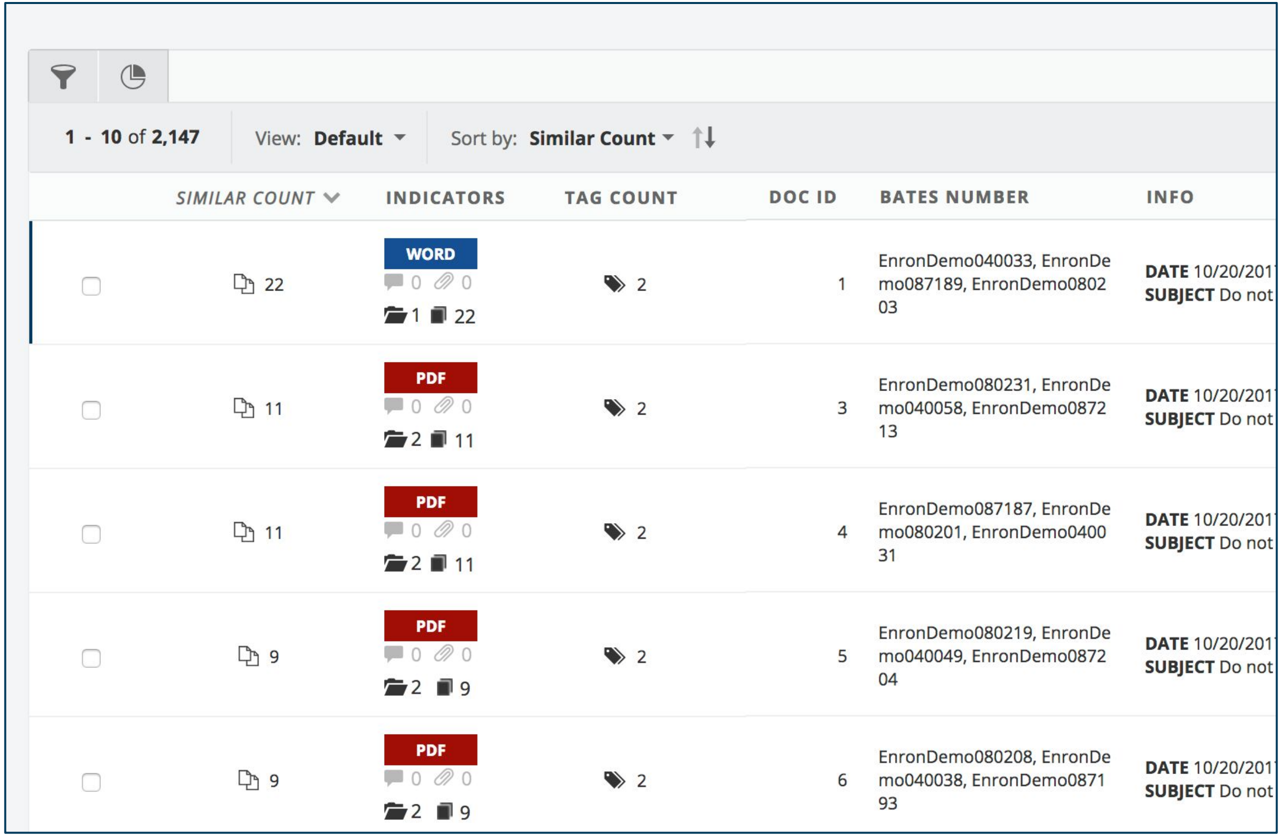
- Pre-compute and cleverly index min hashes.
- Do comparisons dynamically at query time.
- Individual query and storage only **O(n)**.



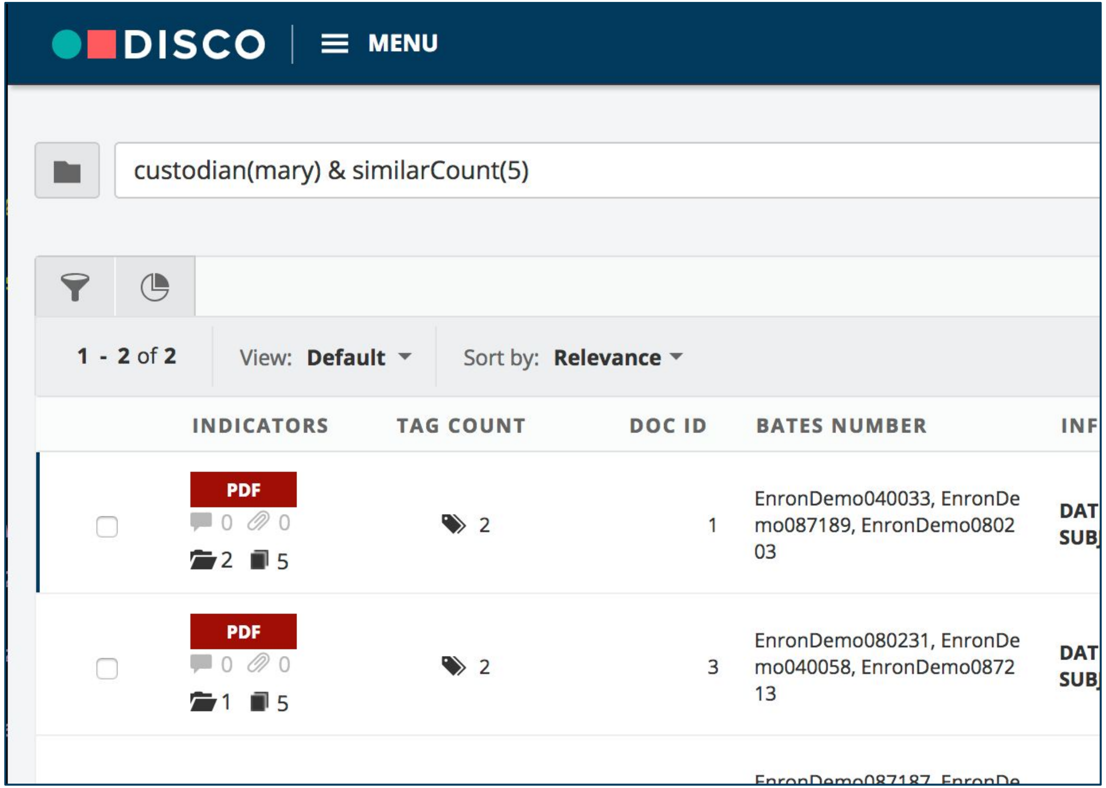
Problem I - Search Sorting Requirements



Problem I - Search Sorting Requirements



Problem II - Search Syntax Requirements



Bottom Line

- We *must* pre-compute similar documents (and counts).
- It is the only way we can:
  - efficiently sort by counts; and
  - know the counts while searching.
- Needs **$O(n^2)$** computation.
- Needs **$O(n^2)$** storage.

Worst Case Analysis

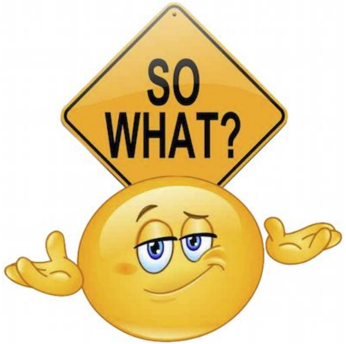
- Documents in a legal case:
  - 3×10^7
  - = 30,000,000 (30 million)
- Possible similar document relations:
  - $(3 \times 10^7) * (3 \times 10^7)$
  - = 900,000,000,000,000 (900 trillion)

Strategy

- Raise the Alarm!

- Talk to the Product Manager!

Business Reaction



- Worst case analysis does not impress anyone.
- Business is fuzzy: it is about risks and trade-offs.
- Needs to be a practical problem, not a theoretical one.

Practically Speaking …

- Average case is much, much lower.
- Average case is within in realm of possible.
- But still need limits on a per-document basis still needed.
- Worst case for design is still hundreds of millions.
- Back to the Product Manager for more “negotiations”.

This is What Success Looks Like

- Requirement Changed: 10K maximum per document.
- New worst case:
    - $(3 \times 10^7) * (1 \times 10^4) = 300{,}000{,}000{,}000$ (300 billion)
- Ain’t that better?
- Average case estimates:
    - 10’s of millions for large legal cases.
- What if we are wrong?

What Next?

- How to generate all the similar docs relations?
  - Answer: Parallelize and distribute (by different team).
- But final “counts” require global knowledge.
  - Must collate after parallel computation is done.
- How to do all this “fast” enough?
  - The customer is waiting.
- How to store all this data?
  - There’s going to be a lot of it.

Data Storage Choices

- AWS Aurora?
- AWS Dynamo?
- Apache Cassandra?
- Other key-value stores? (e.g., Mongo, Redis)
- Other columnar stores? (e.g., Vertica)

Dangers in Choosing Technologies

- Engineers' Biases:
  - Newest
  - Coolest
  - Most interesting
  - Most familiar
  - Theoretical best
  - Resume building

Practical Considerations

- Do we want to introduce a new platform dependency?
- Do we want to introduce a learning curve for the platform?
- Do we have the operational expertise for the platform?
- Amount of maintenance is needed by the platform?
- Time to market considerations?
- Tends to be a buzz-kill for younger engineers.
- All these led to benchmark Aurora vs. Dynamo.

Performance and Cost Testing

- Both Aurora and Dynamo should be able to do the job.
- AWS has different cost models and different performance:
  - Annoying amount of time spent dealing with pricing.
  - CPU, read/write request, GB stored, data transfer, etc.
- Need to tweak and benchmark them to meet SLAs.
  - Hard to know the right price point without testing.
- Only then can we determine the costs and choose.
  - Spoiler Alert: Aurora Won

Can we Afford It?

- Cost of Goods Sold (COGS): important business metric.
- “Cost” in this context is our AWS bill.
- We charge customers by their data size (GB / month).
- Our financial models depend on estimates of costs per GB.
- Material change in COGS is of great interest the business.

Now, we do the math …

- The storage cost can grow considerably, but:
  - the more similar docs storage we need,
  - the more documents there are, and
  - the more we would be charging.
- This adds much less than 1% to costs per GB.
- i.e., Storage truly is cheap.

Now We Build It

- Distributed System Design
- Interface Definitions
- Logging
- Monitoring
- Alerts
- Metrics / Dashboards
- High Availability (HA)
- Disaster Recovery (DR)
- Deployment plan
- Release plan
- Test plan
- Documentation
- Component diagrams
- Sequence diagrams
- Architecture Review
- Configuration
- Project task breakdown
- Time estimates
- Cost estimates
- Cross-team coordination
- Scheduling

Solution

- An expensive, inelegant and inefficient system, but it
- enables features that will be a competitive advantage and
- will not materially cut into our profit margins.

A Conclusion

- There is no “right” way to build a piece of software.
- Business context dictates what is “right”.
- The business context will change as the company grows.

“Today’s bad hack was yesterday’s good decision.”

What are your questions?

Thank You

www.csdisco.com