Duplicate Detection

David Kauchak
cs458
Fall 2012

adapted from:
http://www.stanford.edu/class/cs276/handouts/lecture14-Crawling.ppt

Administrative

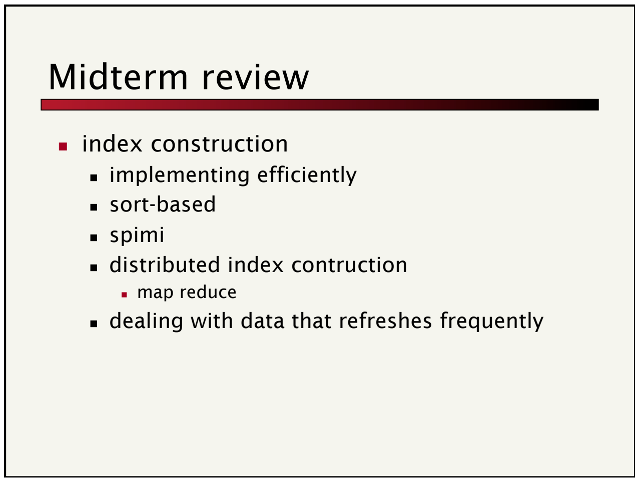
- Assignment 3
- Midterm
  - posted online by tomorrow morning
  - take it by Tue. at midnight
  - 1.5 hours
  - Don't talk to anyone about it until after Tuesday

Midterm review: general notes

- We've covered a lot of material
  - Anything from lecture, readings, homeworks and assignments is fair game
  - Today's material NOT on the midterm
- T/F, short answer, short work-through problems
- Some questions like homework, but also "conceptual" questions
- Won't need calculator

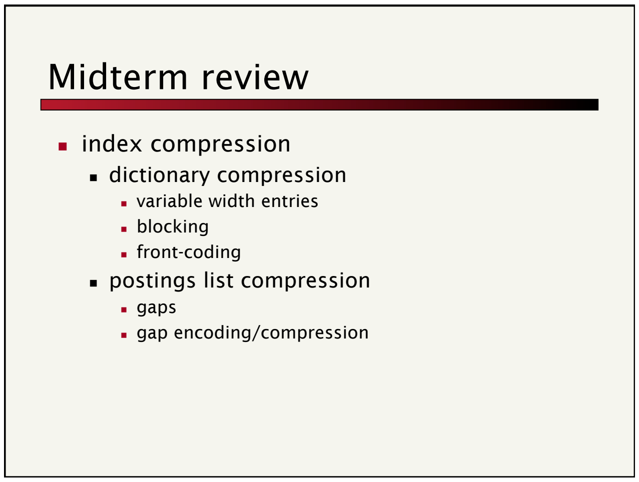
Midterm review

- indexes
  - representation
  - skip pointers
  - why we need an index
- boolean index
  - merge operation
  - query optimization
  - phrase queries (query proximity)



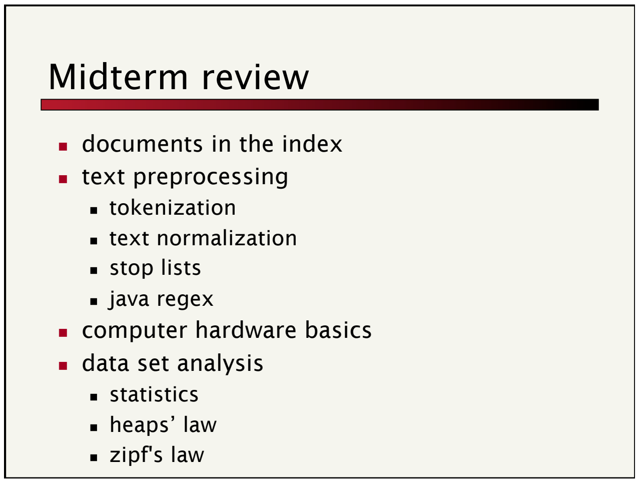
Midterm review

- index construction
  - implementing efficiently
  - sort-based
  - spimi
  - distributed index contruction
    - map reduce
  - dealing with data that refreshes frequently



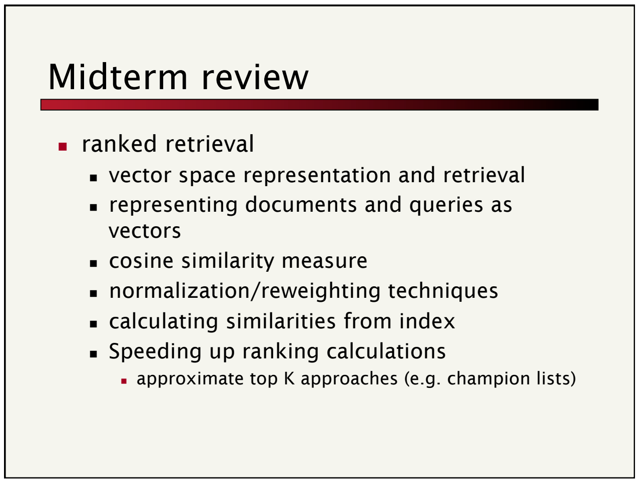
Midterm review

- index compression
  - dictionary compression
    - variable width entries
    - blocking
    - front-coding
  - postings list compression
    - gaps
    - gap encoding/compression



Midterm review

- documents in the index
- text preprocessing
  - tokenization
  - text normalization
  - stop lists
  - java regex
- computer hardware basics
- data set analysis
  - statistics
  - heaps' law
  - zipf's law



Midterm review

- ranked retrieval
  - vector space representation and retrieval
  - representing documents and queries as vectors
  - cosine similarity measure
  - normalization/reweighting techniques
  - calculating similarities from index
  - Speeding up ranking calculations
    - approximate top K approaches (e.g. champion lists)

Midterm review

- Evaluation
  - precision
  - recall
  - F1
  - 11-point precision
  - MAP
  - Kappa statistic
  - A/B testing

Midterm review

- snippet/summary generation

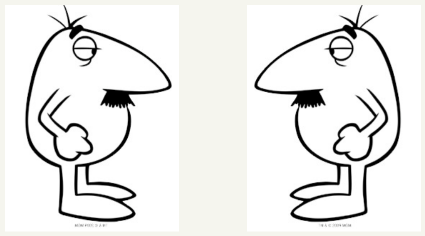
- spelling correction
  - edit distance
  - word n-grams
  - jaccard coefficient

- relevance feedback

Midterm review

- web
  - basic web search engine
  - spam
  - estimating the size of the web (or the size of a search engine's index)

Duplicate detection



http://rlv.zcache.com/cartoon_man_with_balled_fist_postcard-p239288482636625726trdg_400.jpg

Duplicate documents

The web is full of duplicated content

- Redundancy/mirroring
- Copied content

Do we care?

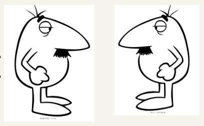
How can we detect duplicates?

Many approaches…. Today:

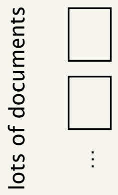
Hashing

- Hash each document
- Compares hashes
- For those that are equal, check if the content is equal

Key challenge: efficiency

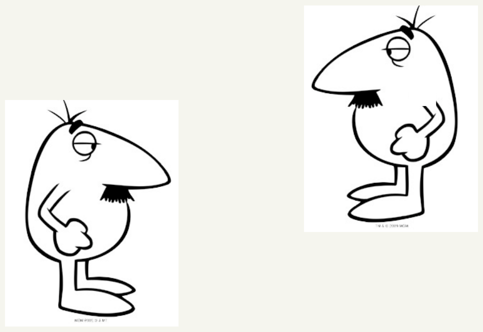
dup()

Are two documents duplicates?
(has to be fast!)



Can't do afford to do complete pairwise comparisons.

Duplicate?



Near duplicate documents

Many, many cases of near duplicates

- E.g., last modified date the only difference between two copies of a page

A good hashing function specifically tries not to have collisions

Ideas?

- Locality sensitive hashing – (http://www.mit.edu/~andoni/LSH/)
- Similarity – main challenge is efficiency!

Computing Similarity

We could use edit distance, but way too slow

What did we do for spelling correction?

compare word n-grams (shingles) overlap

- ***a rose is a rose is a rose*** →
  - a_rose_is_a
  - rose_is_a_rose
  - is_a_rose_is
  - a_rose_is_a

Use Jaccard Coefficient to measure the similarity between documents (A and B)/(A or B)

N-gram intersection

Computing exact set intersection of n-grams between all pairs of documents is expensive/intractable

How did we solve the efficiency problem for spelling correction?

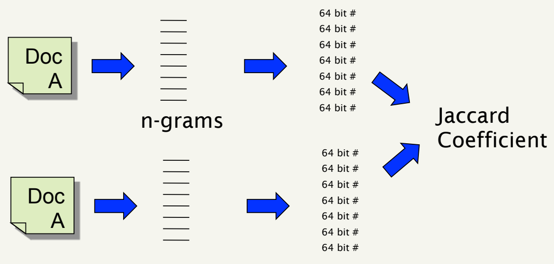
- Indexed words by character n-grams
- AND query of the character n-grams in our query word

Will this work for documents?

Number of word n-grams for a document is too large!

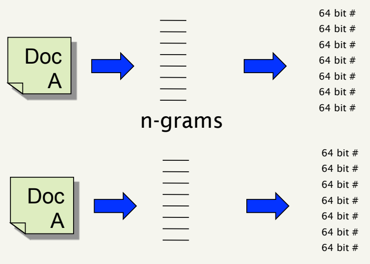
Efficient calculation of JC

Use a hash function that maps an n-gram to a 64 bit number

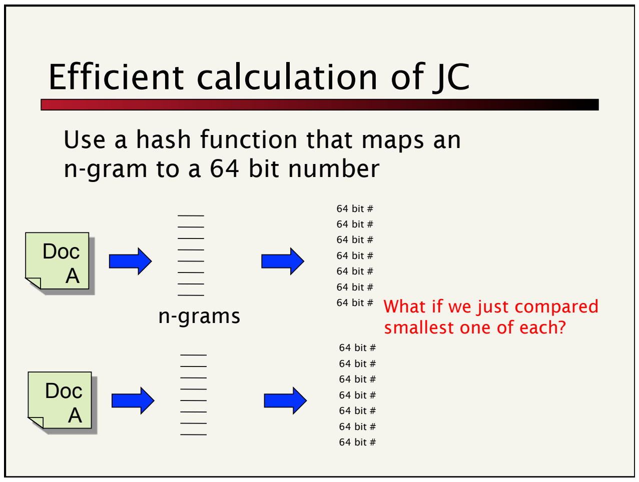


Efficient calculation of JC

Use a hash function that maps an n-gram to a 64 bit number



I've reduced the storage. but how can I tell if they're similar?

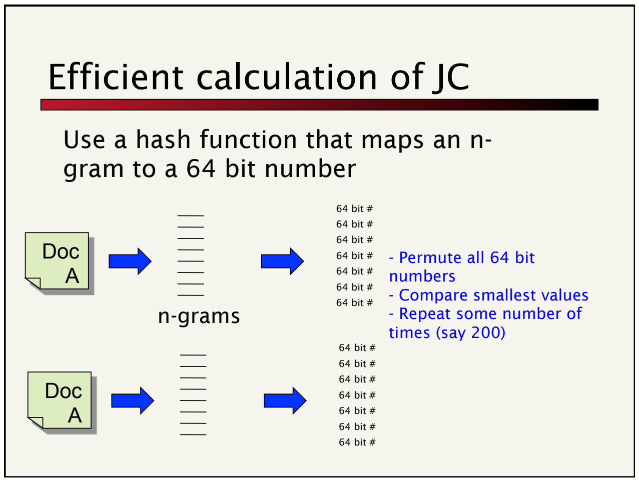


Efficient calculation of JC

Use a hash function that maps an n-gram to a 64 bit number

Doc A
n-grams
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
Doc A
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #

What if we just compared smallest one of each?

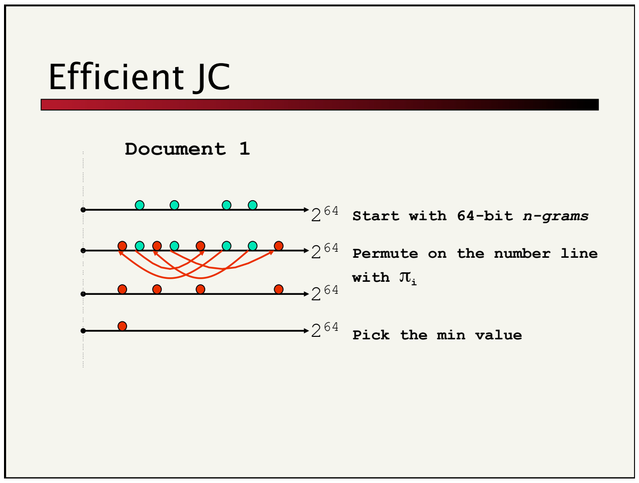


Efficient calculation of JC

Use a hash function that maps an n-gram to a 64 bit number

Doc A
n-grams
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
Doc A
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #
64 bit #

- Permute all 64 bit numbers
- Compare smallest values
- Repeat some number of times (say 200)

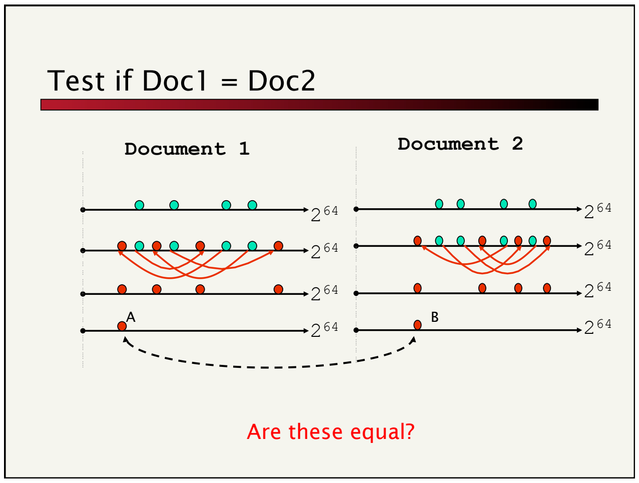


Efficient JC

Start with 64-bit *n-grams*

Permute on the number line with π_i

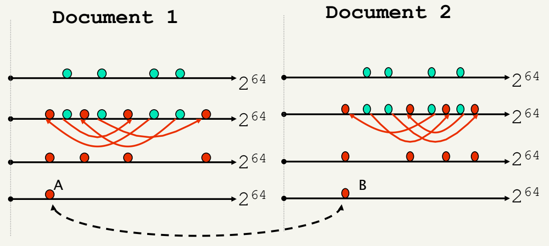
Pick the min value



Test if Doc1 = Doc2

Are these equal?

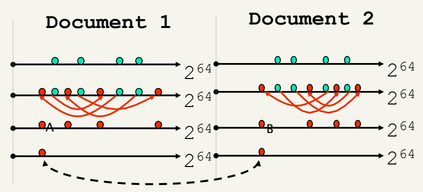
Test if Doc1 = Doc2



The minimum values after the permutations will be equal with probability =

`Size_of_intersection / Size_of_union`

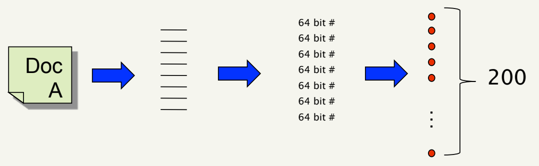
Repeat



- Repeat this, say 200 times, with different permutations
- Measure the number of times they're equal
- This is a reasonable estimate for the JC

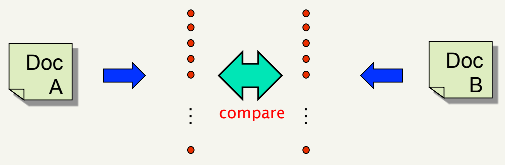
Putting it all together

For each document, precompute the 200 permuted, smallest numbers



Putting it all together

These 200 64-bit numbers then represent the document and can be used to calculate JC between any two docs



JC = proportion that are equal

All signature pairs

Now we have an extremely efficient method for *estimating* a Jaccard coefficient for a single pair of documents.

But we still have to estimate N^2 coefficients where N is the number of web pages.

- **Still slow**

Need to reduce the set of options

- locality sensitive hashing (LSH)
- sorting (Henzinger 2006)

Article discussion

Is it more relevant to know what your close friends are searching for, or the majority of people worldwide?

How do companies balance out trying to provide the best search results possible while trying to make the most money possible? Are those goals aligned?

How would you know if a search engine is "impartial" i.e. that they provide you the best results regardless of how it affects them as a company?

http://www.youtube.com/watch?v=KNWuOJXP-R4