Rolling the Dice on Big Data

Ilse Ipsen
Department of Mathematics

The Economist, 27 February 2010

Science, 11 February 2011

McKinsey Global Institute, May 2011

McKinsey Global Institute

May 2011

Big data: The next frontier for innovation, competition, and productivity


Big data—a growing torrent
$600 to buy a disk drive that can store all of the world's music
5 billion mobile phones in use in 2010
30 billion pieces of content shared on Facebook every month
40% projected growth in global data generated per year vs. 5% growth in global IT spending
235 terabytes data collected by the US Library of Congress in April 2011
15 out of 17 sectors in the United States have more data stored per company than the US Library of Congress


Big data—a growing torrent
$600 to buy a disk drive that can store all of the world's music
5 billion mobile phones in use in 2010
30 billion pieces of content shared on Facebook every month
40% projected growth in global data generated per year vs. 5% growth in global IT spending
235 terabytes data collected by the US Library of Congress in April 2011
15 out of 17 sectors in the United States have more data stored per company than the US Library of Congress

What is "Big" ?

Measuring Units

- 1 byte $\sim$ 1 character
- 10 bytes $\sim$ 1 word
- 100 bytes $\sim$ 1 sentence
- 1 kilobyte $=$ 1,000 bytes $\sim$ 1 page

Measuring Units

- 1 byte $\sim$ 1 character
- 10 bytes $\sim$ 1 word
- 100 bytes $\sim$ 1 sentence
- 1 kilobyte = 1,000 bytes $\sim$ 1 page

- 1 megabyte = 1,000 kilobytes
  $\sim$ complete works of Shakespeare
- 1 gigabyte = 1,000 megabytes
  $\sim$ a big shelf full of books

Measuring Units

- 1 byte $\sim$ 1 character
- 10 bytes $\sim$ 1 word
- 100 bytes $\sim$ 1 sentence
- 1 kilobyte $=$ 1,000 bytes $\sim$ 1 page

- 1 megabyte $=$ 1,000 kilobytes
  $\sim$ complete works of Shakespeare
- 1 gigabyte $=$ 1,000 megabytes
  $\sim$ a big shelf full of books

- 1 terabyte $=$ 1,000 gigabytes
  $\sim$ all books in the Library of Congress

Measuring Units

- 1 byte $\sim$ 1 character
- 10 bytes $\sim$ 1 word
- 100 bytes $\sim$ 1 sentence
- 1 kilobyte = 1,000 bytes $\sim$ 1 page
- 1 megabyte = 1,000 kilobytes
  $\sim$ complete works of Shakespeare
- 1 gigabyte = 1,000 megabytes
  $\sim$ a big shelf full of books
- 1 terabyte = 1,000 gigabytes
  $\sim$ all books in the Library of Congress
- 1 petabyte = 1,000 terabytes
  $\sim$ 20 million 4-door filing cabinets full of text

1 byte $\sim$ 1 grain of sand

1 byte $\sim$ 1 grain of sand

1 terabyte $\sim$ number of grains to fill a swimming pool

Rolling the Dice on Big Data

Data

Rolling the Dice on Big Data

Rolling the Dice on Big Data

Not quite

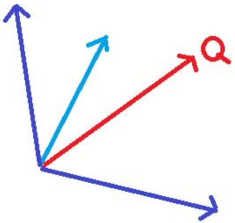
The Data in this Talk

Given:

Database: Collection of "documents" (data points)
Query: Single "document" (data point)

Want:

Documents closest to query



A tiny example to illustrate a "big data" problem

A "Tiny Data" Example

Database: Emails from known authors

Email 1: shipment of gold damaged in a fire
Email 2: delivery of silver arrived in a silver truck
Email 3: shipment of gold arrived in a truck

Query: Email from unknown author

gold silver truck

Which emails match the query best?
These emails may give clues about the author of query

Simplest approach for matching: **Word frequency**

Tabulating Emails and Query

Database (term document matrix) + Query

Terms	Email 1	Email 2	Email 3	Query
a	1	1	1	0
arrived	0	1	1	0
damaged	1	0	0	0
delivery	0	1	0	0
fire	1	0	0	0
gold	1	0	1	1
in	1	1	1	0
of	1	1	1	0
silver	0	2	0	1
shipment	1	0	1	0
truck	0	1	1	1

Basic Approach for Finding Matching Emails

1. Common words
   For each Email: Count number of words **common** to Email and Query

Basic Approach for Finding Matching Emails

1. Common words
   For each Email: Count number of words **common** to Email and Query

2. Length
   Count **number of words** in each Email, and in Query

Basic Approach for Finding Matching Emails

1. Common words
   For each Email: Count number of words **common** to Email and Query

2. Length
   Count **number of words** in each Email, and in Query

3. Matching score for each Email:

$$\text{Matching score} = \frac{\text{Number of common words}}{(\text{Length of Email}) * (\text{Length of Query})}$$

Emails with highest matching scores:
May give clues about authors of Query

“Count” Common Words in Query and Email 1

Terms	E_1	Q	Multiply
a	1	0	0
arrived	0	0	0
damaged	1	0	0
delivery	0	0	0
fire	1	0	0
gold	1	1	1
in	1	0	0
of	1	0	0
silver	0	1	0
shipment	1	0	0
truck	0	1	0
Sum			1

common words in Email 1 and Query: $E_1 * Q = 1$

“Count” Common Words in Query and Email 2

Terms	E_2	Q	Multiply
a	1	0	0
arrived	1	0	0
damaged	0	0	0
delivery	1	0	0
fire	0	0	0
gold	0	1	0
in	1	0	0
of	1	0	0
silver	2	1	2
shipment	0	0	0
truck	1	1	1
Sum			3

common words in Email 2 and Query: $E_2 * Q = 3$

"Count" Common Words in Query and Email 3

Terms	E_3	Q	Multiply
a	1	0	0
arrived	1	0	0
damaged	0	0	0
delivery	0	0	0
fire	0	0	0
gold	1	1	1
in	1	0	0
of	1	0	0
silver	0	1	0
shipment	1	0	0
truck	1	1	1
Sum			2

common words in Email 3 and Query: $E_3 * Q = 2$

Basic Approach for Finding Matching Emails

1. Number of words **common** to Emails and Query

$$
\begin{aligned}
E_1 * Q &= 1 \\
E_2 * Q &= 3 \\
E_3 * Q &= 2
\end{aligned}
$$

2. Length
   Count **number of words** in each email, and in query

Length of Query

Terms	Q	Square
a	0	0
arrived	0	0
damaged	0	0
delivery	0	0
fire	0	0
gold	1	1
in	0	0
of	0	0
silver	1	1
shipment	0	0
truck	1	1
$\sqrt{\textbf{Sum}}$		$\sqrt{3}$

Length of Query: $\|Q\| = \sqrt{3} \approx 1.7$

Length of Email 2

Terms	E_2	Square
a	1	1
arrived	1	1
damaged	0	0
delivery	1	1
fire	0	0
gold	0	0
in	1	1
of	1	1
silver	2	4
shipment	0	0
truck	1	1
$\sqrt{\textbf{Sum}}$		$\sqrt{10}$

Length of Email 2: $\|E_2\| = \sqrt{10} \approx 3.2$

Basic Approach for Finding Matching Emails

1. Number of words **common** to Emails and Query

$$\begin{aligned} E_1 * Q &= 1 \\ E_2 * Q &= 3 \\ E_3 * Q &= 2 \end{aligned}$$

2. **Length** of Emails and Query

$$\begin{aligned} \|Q\| &= \sqrt{3} \approx 1.7 \\ \|E_1\| &= \sqrt{7} \approx 2.6 \\ \|E_2\| &= \sqrt{10} \approx 3.2 \\ \|E_3\| &= \sqrt{7} \approx 2.6 \end{aligned}$$

Matching Score for each Email

$$\text{Matching score} = \frac{\text{Number of common words}}{(\text{Length of email}) * (\text{Length of query})}$$

Email 1

$$\frac{E_1 * Q}{\|E_1\| \, \|Q\|} = \frac{1}{\sqrt{7}\,\sqrt{3}} \approx .22$$

Email 2

$$\frac{E_2 * Q}{\|E_2\| \, \|Q\|} = \frac{3}{\sqrt{10}\,\sqrt{3}} \approx .55$$

Email 3

$$\frac{E_3 * Q}{\|E_3\| \, \|Q\|} = \frac{2}{\sqrt{7}\,\sqrt{3}} \approx .44$$

Email 2 is the best match for the query

Conclusion for “Tiny Data” Example

Database: Emails from known authors

Email 1: shipment of gold damaged in a fire
Email 2: delivery of silver arrived in a silver truck
Email 3: shipment of gold arrived in a truck

Query: Email from unknown author

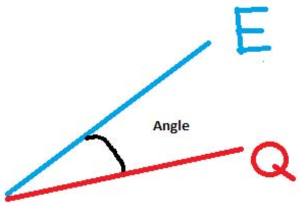
gold silver truck

Best matching email:

Email 2: delivery of silver arrived in a silver truck

The Reason for the Weird Way of Counting

Vector Space Model



Emails, Query = vectors

Matching score = cosine of **angle** between Email and Query

$$\frac{E * Q}{\|E\|\,\|Q\|} = \cos\angle(E, Q)$$

What this means “in practice”

Average number of emails per day: 294 billion
Number words in English language: at least 250,000

Matching one query with a **single** email:

250,000 operations (one for every possible word)

Matching one query with **all** emails:

250,000 * 294 billion $= 73.5 \cdot 10^{15}$ operations

What this means "in practice"

Average number of emails per day: 294 billion
Number words in English language: at least 250,000

Matching one query with a **single** email:

250,000 operations (one for every possible word)

Matching one query with **all** emails:

250,000 * 294 billion $= 73.5 \cdot 10^{15}$ operations

- Fast PC (Intel Core i7 980 XE)

  109 Gflops $= 109 * 10^9$ floating point operations per second

  Matching one query with all emails: about 8 days

What this means "in practice"

Average number of emails per day: 294 billion
Number words in English language: at least 250,000

Matching one query with a **single** email:

250,000 operations (one for every possible word)

Matching one query with **all** emails:

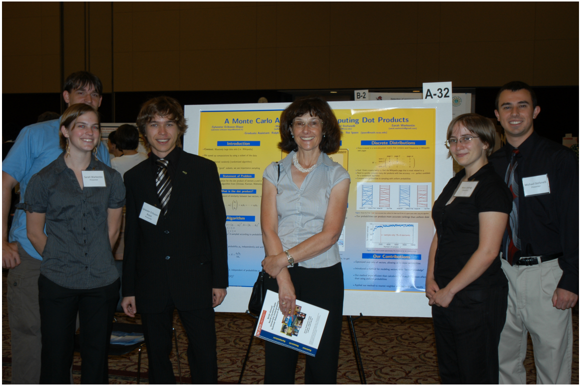
250,000 * 294 billion $= 73.5 \cdot 10^{15}$ operations

- Fast PC (Intel Core i7 980 XE)
  109 Gflops $= 109 * 10^9$ floating point operations per second
  Matching one query with all emails: about 8 days
- US supercomputer (Cray XT5, Opteron quad core 2.3GHz)
  Peak 1,381,400 Gflops
  Matching one query with all emails: about 1 minute

Can the Matching be Performed Faster?

Can the Matching be Performed Faster?

Yes!



Ralph Abbey, Sarah Warkentin, Sylvester Eriksson-Bique, Mary Solbrig, Michael Stefanelli

Rolling the Dice on Big Data

Rolling the Dice

Rolling the Dice on Big Data

Rolling the Dice

on which words to use for the matching

Randomized Query Matching Algorithm

Idea

Do not use every word in query and emails

Monte Carlo Sampling: Use only selected words

{Downsize to smaller database with fewer words}

Randomized Query Matching Algorithm

Idea

Do not use every word in query and emails
Monte Carlo Sampling: Use only selected words
{Downsize to smaller database with fewer words}

Justification

- Don't need exact matching scores
  Identify only emails with highest matching scores
- Database available for offline computation
  Derive "statistics" based on word frequencies
- Perform query matching online
  Use "statistics" to select words used for matching

Suggestions for Downsizing the Database

- Statistics

  n: number of words in database
  Q_j: frequency of word j in query
  W_j: frequency of word j in database

- Suggestion for selecting word j

  Probability of sampling word j

$$p_j = \frac{W_j\, Q_j}{W_1\, Q_1 + \cdots + W_n\, Q_n}$$

Frequently occurring words more likely to be sampled

Rolling the Dice = Downsizing the Database

User input

s: number of samples

$\{$*number of words in downsized database*$\}$

Monte Carlo Sampling $\{$Roll the dice s times$\}$

For $t = 1, \ldots, s$

Sample *index* j_t *from* $\{1, \ldots, n\}$ **with probability** p_{j_t} *independently and with replacement*

Downsized database contains only s words:

word j_1, word j_2, ..., word j_s

Matching with Downsized Database

Downsized database: word j_1, word j_2, ..., word j_s

Word frequency in Query: $\hat{Q} = \begin{pmatrix} Q_{j_1} & Q_{j_2} & \dots & Q_{j_s} \end{pmatrix}$

For each Email E:

Matching with Downsized Database

Downsized database: word j_1, word j_2, ..., word j_s
Word frequency in Query: $\hat{Q} = \begin{pmatrix} Q_{j_1} & Q_{j_2} & \dots & Q_{j_s} \end{pmatrix}$

For each Email E:

- Word frequency $\hat{E} = \begin{pmatrix} F_{j_1} & F_{j_2} & \dots & F_{j_s} \end{pmatrix}$

Matching with Downsized Database

Downsized database: word j_1, word j_2, ..., word j_s
Word frequency in Query: $\hat{Q} = \begin{pmatrix} Q_{j_1} & Q_{j_2} & \ldots & Q_{j_s} \end{pmatrix}$

For each Email E:

- Word frequency $\hat{E} = \begin{pmatrix} F_{j_1} & F_{j_2} & \ldots & F_{j_s} \end{pmatrix}$
- Approximate number of words **common** to Email and Query

$$C = \frac{1}{s} \left(\frac{F_{j_1}\, Q_{j_1}}{p_{j_1}} + \frac{F_{j_2}\, Q_{j_2}}{p_{j_2}} + \cdots + \frac{F_{j_s}\, Q_{j_s}}{p_{j_s}} \right)$$

$\{s,\ p_{j_1},\ p_{j_2},\ \ldots,\ p_{j_s}$ compensate for fewer words$\}$

Matching with Downsized Database

Downsized database: word j_1, word j_2, . . ., word j_s

Word frequency in Query: $\hat{Q} = \begin{pmatrix} Q_{j_1} & Q_{j_2} & \dots & Q_{j_s} \end{pmatrix}$

For each Email E:

- Word frequency $\hat{E} = \begin{pmatrix} F_{j_1} & F_{j_2} & \dots & F_{j_s} \end{pmatrix}$
- Approximate number of words **common** to Email and Query

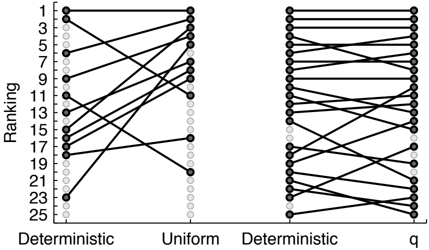
$$C = \frac{1}{s}\left(\frac{F_{j_1}\,Q_{j_1}}{p_{j_1}} + \frac{F_{j_2}\,Q_{j_2}}{p_{j_2}} + \cdots + \frac{F_{j_s}\,Q_{j_s}}{p_{j_s}}\right)$$

$\{s,\ p_{j_1},\ p_{j_2},\ \dots,\ p_{j_s}$ compensate for fewer words$\}$

- Approximate **matching score** of Email: $\frac{C}{\|\hat{E}\|\ \|\hat{Q}\|}$

Reuters-215787 Collection: Transcribed Subset

201 documents and 5601 words
Number of sampled words $s = 56 \approx 1$ percent

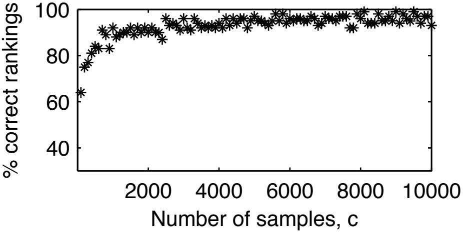


Bucket of computed 25 best matches contains
Correct 10 best matches in 99% of all cases

Wikipedia Dataset

200 documents and 198,853 words

Average percent of correct 10 best matches
as function of sample size



Sampling 1% of the words gives correct 9 best matches.
More sampling does not help a lot.

Summary

Big data

Matching queries against document database

Rolling the dice

Randomized downsizing of database vocabulary
Frequently occurring words more likely to be kept

But ...

Summary

Big data

Matching queries against document database

Rolling the dice

Randomized downsizing of database vocabulary
Frequently occurring words more likely to be kept

But ...

Why not use a predictable (deterministic) algorithm?
Why use a randomized algorithm?

Summary

Big data

Matching queries against document database

Rolling the dice

Randomized downsizing of database vocabulary
Frequently occurring words more likely to be kept

But ...

Why not use a predictable (deterministic) algorithm?
Why use a randomized algorithm?

Advantages of randomized algorithm

- Easy to analyze
- Fast, and simple to implement
- As good in practice as deterministic algorithm
  (for this type of application)

The Bigger Picture

Many different methods for fast query matching

Algorithm in this talk:

Randomized matrix vector multiplication

Other randomized matrix algorithms:

Matrix multiplication
Subset selection
Least squares problems (regression)
Low rank approximation (PCA)

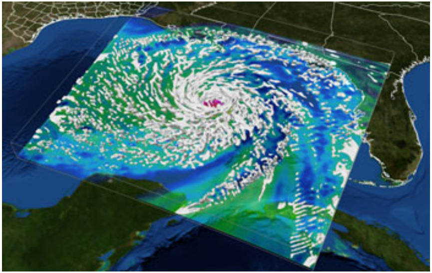
Applications for randomized algorithms:
Social network analysis, population genetics, circuit testing, ...

National Science Foundation, 29 March 2012

Press Release 12-060

NSF Leads Federal Efforts In Big Data

At White House event, NSF Director announces new Big Data solicitation, $10 million Expeditions in Computing award, and awards in cyberinfrastructure, geosciences, training



Hurricane Ike visualization created by Texas Advanced Computing Center (TACC) supercomputer Ranger.
Credit and Larger Version