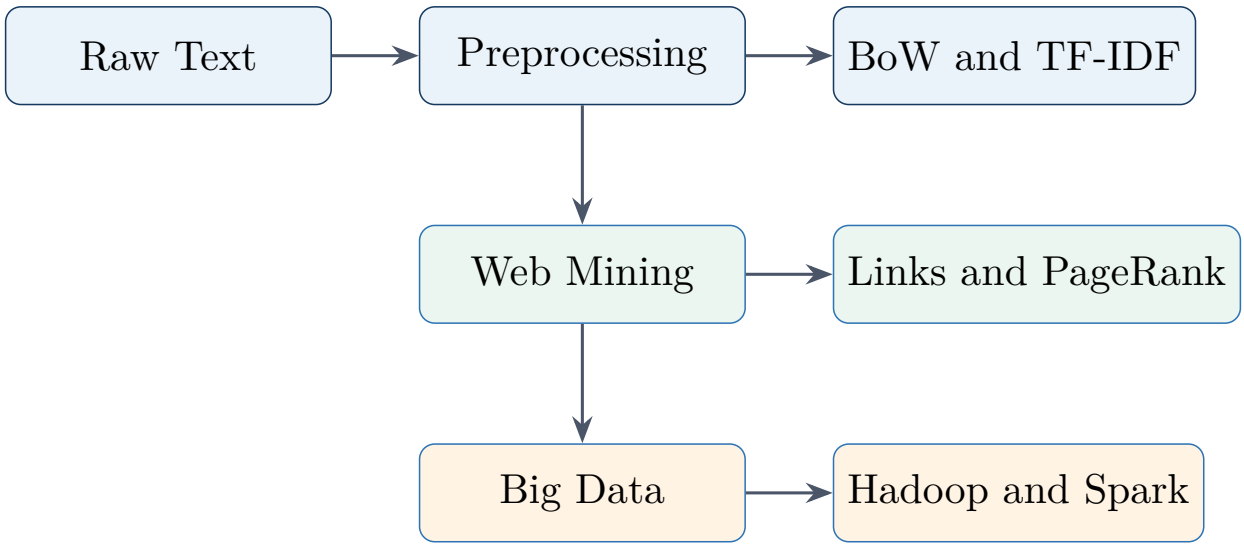
Text Mining, Web Mining, and Introduction to Big Data Analytics

Mathematical Notes

Text Preprocessing, N-Grams, Bag-of-Words, TF-IDF, Text Classification, Web Content, Web Structure, PageRank, Web Usage, Hadoop, HDFS, YARN, MapReduce, and Spark

Central idea. Text mining converts unstructured language into numerical features, web mining discovers patterns from web content, links, and behavior, and big-data platforms distribute storage and computation when one machine is no longer sufficient.

Concept Map



I. Text Mining Foundations

1. What Is Text Mining?

Text mining is the process of extracting useful information, patterns, relationships, and knowledge from unstructured or semi-structured textual data.

Common sources include emails, messages, social-media posts, news articles, customer reviews, research papers, medical reports, and web pages. Common applications include sentiment analysis, spam detection, document classification, information retrieval, topic discovery, clustering, and recommendation.

Main challenge. Raw text cannot be used directly by most machine-learning algorithms. It must be cleaned and transformed into a numerical representation.

2. Text Mining Pipeline

The standard pipeline is

$$\boxed{\text{Raw Text} \to \text{Preprocessing} \to \text{Feature Extraction} \to \text{Mining/ML} \to \text{Knowledge}.}$$

If T denotes raw text, $P(\cdot)$ denotes preprocessing, $\phi(\cdot)$ denotes feature extraction, and $f(\cdot)$ denotes a learned model, then

$$x = \phi(P(T)), \qquad \hat{y} = f(x).$$

Worked example: complete pipeline.

Raw review:

$$T = \text{“The product is extremely good and easy to use!”}$$

Stage	Output
Lowercasing	the product is extremely good and easy to use
Tokenization	[the, product, is, extremely, good, and, easy, to, use]
Stop-word removal	[product, extremely, good, easy, use]
Feature extraction	numerical vector x
Classification	$P(\text{Positive} \mid x) = 0.94$

With threshold $\tau = 0.50$,

$$0.94 > 0.50 \quad \Longrightarrow \quad \boxed{\hat{y} = \text{Positive.}}$$

II. Text Preprocessing

1. Lowercasing, Tokenization, Punctuation, and Stop Words

Let the original sentence be

$$T_0 = \text{“Data Mining is VERY useful! It helps us discover useful patterns.”}$$

Lowercasing reduces separate vocabulary entries such as `Data` and `data`. Tokenization splits text into units, punctuation removal deletes selected symbols, and stop-word removal removes frequent function words when they are not useful for the task.

Worked example: preprocessing transformation.

Operation	Result
Original	Data Mining is VERY useful! It helps us discover useful patterns.
Lowercase	data mining is very useful! it helps us discover useful patterns.
Tokenize	[data, mining, is, very, useful, it, helps, us, discover, useful, patterns]
Remove punctuation	punctuation symbols removed
Remove stop words	[data, mining, useful, helps, discover, useful, patterns]
Final normalized text	data mining useful helps discover useful patterns

The original token count is 11. After removing four stop words,

$$n_{\text{final}} = 11 - 4 = 7.$$

The reduction percentage is

$$\frac{11-7}{11} \times 100\% = 36.36\%.$$

Task dependence. Removing the word "not" from "not good" would reverse the meaning. Stop-word lists must therefore match the application.

2. Stemming and Lemmatization

Stemming applies rule-based removal of prefixes or suffixes. It is fast but may produce a nonword. **Lemmatization** uses vocabulary and linguistic knowledge to return a valid base form.

Worked example: vocabulary reduction.

Original word	Stem	Lemma
playing	play	play
played	play	play
studies	studi	study
cars	car	car
running	run	run
better	better	good

The original vocabulary contains six distinct surface forms. Stemming maps them to five distinct forms,

$$V_{\text{stem}} = \{\text{play, studi, car, run, better}\}, \qquad |V_{\text{stem}}| = 5.$$

Lemmatization also produces five distinct base forms,

$$V_{\text{lemma}} = \{\text{play, study, car, run, good}\}, \qquad |V_{\text{lemma}}| = 5.$$

The vocabulary reduction is

$$\frac{6-5}{6} \times 100\% = 16.67\%.$$

The lemma `good` is linguistically meaningful, while a stemmer may fail to connect `better` with `good`.

3. N-Grams

An n-gram is a contiguous sequence of n tokens. If a document contains m tokens, the number of contiguous n-grams is

$$\boxed{N_n = m - n + 1}, \qquad m \geq n.$$

Worked example: unigram, bigram, and trigram construction.

Consider

$$T = \text{“data mining improves decisions”}.$$

The token sequence contains $m = 4$ words.

Type	Count	N-grams
Unigram	$4-1+1=4$	[data], [mining], [improves], [decisions]
Bigram	$4-2+1=3$	[data mining], [mining improves], [improves decisions]
Trigram	$4-3+1=2$	[data mining improves], [mining improves decisions]

Thus,

$$\boxed{N_1 = 4, \qquad N_2 = 3, \qquad N_3 = 2.}$$

N-grams preserve limited local word order, unlike a basic Bag-of-Words representation.

III. Numerical Text Representation

1. Bag-of-Words

Let the vocabulary be

$$V = \{t_1, t_2, \ldots, t_m\}.$$

A document D_i is represented by the count vector

$$\boxed{x_i = (c_{i1}, c_{i2}, \ldots, c_{im})},$$

where c_{ij} is the number of occurrences of term t_j in document D_i.

Worked example: document-term matrix.

Consider

$$D_1 = \text{“data mining useful”},$$

$$D_2 = \text{“data science useful”},$$

$$D_3 = \text{“mining discovers patterns”}.$$

The vocabulary is

$$V = \{\text{data, mining, science, useful, discovers, patterns}\}.$$

The complete Bag-of-Words matrix is

Document	data	mining	science	useful	discovers	patterns
D_1	1	1	0	1	0	0
D_2	1	0	1	1	0	0
D_3	0	1	0	0	1	1

Therefore,

$$x_1 = (1, 1, 0, 1, 0, 0),$$

$$x_2 = (1, 0, 1, 1, 0, 0),$$

$$x_3 = (0, 1, 0, 0, 1, 1).$$

The vocabulary size is $|V| = 6$, so the matrix dimension is

$$3 \times 6 = 18 \text{ entries.}$$

The number of nonzero entries is 9, giving sparsity

$$1 - \frac{9}{18} = 0.50 = 50\%.$$

2. Term Frequency

Normalized term frequency is

$$\boxed{\mathrm{TF}(t, d) = \frac{f(t, d)}{\sum_{w \in d} f(w, d)}},$$

where $f(t, d)$ is the number of occurrences of term t in document d.

Worked example: normalized term frequency.

Let

$$D_1 = \text{“data mining mining”}.$$

Term	Frequency	Normalized TF
data	1	$1/3 = 0.333$
mining	2	$2/3 = 0.667$

Therefore,

$$\boxed{\mathrm{TF}(\text{mining}, D_1) = \frac{2}{3} \approx 0.667.}$$

3. Inverse Document Frequency

For a collection of N documents,

$$\boxed{\mathrm{IDF}(t) = \log\left(\frac{N}{\mathrm{df}(t)}\right)},$$

where $\mathrm{df}(t)$ is the number of documents containing t.

Worked example: IDF calculation.

Consider

$$D_1 = \text{“data mining mining”}, \quad D_2 = \text{“data science”}, \quad D_3 = \text{“web mining”}.$$

Here $N = 3$.

Term	$\mathrm{df}(t)$	$\mathrm{IDF}(t)$
data	2	$\log(3/2) = 0.405$
mining	2	$\log(3/2) = 0.405$
science	1	$\log(3) = 1.099$
web	1	$\log(3) = 1.099$

The rarer terms `science` and `web` receive higher IDF values.

4. TF-IDF

TF-IDF combines local frequency with collection-level rarity:

$$\boxed{\mathrm{TFIDF}(t, d) = \mathrm{TF}(t, d) \times \mathrm{IDF}(t)}.$$

Worked example: complete TF-IDF matrix.

Using the same three documents and the vocabulary

$$V = \{\text{data, mining, science, web}\},$$

the normalized TF values are

Document	data	mining	science	web
D_1	1/3	2/3	0	0
D_2	1/2	0	1/2	0
D_3	0	1/2	0	1/2

Multiplying TF by IDF gives

Document	data	mining	science	web
D_1	0.135	0.270	0	0
D_2	0.203	0	0.549	0
D_3	0	0.203	0	0.549

For the term `mining` in D_1,

$$\mathrm{TF}(\text{mining}, D_1) = \frac{2}{3},$$

$$\mathrm{IDF}(\text{mining}) = \log\left(\frac{3}{2}\right) = 0.405,$$

$$\mathrm{TFIDF}(\text{mining}, D_1) = \frac{2}{3}(0.405) = 0.270.$$

$$\boxed{\mathrm{TFIDF}(\text{mining}, D_1) = 0.270.}$$

5. Cosine Similarity

For document vectors x and y, cosine similarity is

$$\boxed{\cos(x, y) = \frac{x^\top y}{\|x\|_2 \, \|y\|_2}}.$$

A value near 1 indicates similar directions; a value near 0 indicates little shared weighted vocabulary.

Worked example: similarity of two TF-IDF vectors.

From the previous example,

$$x_{D_1} = (0.135, 0.270, 0, 0),$$

$$x_{D_2} = (0.203, 0, 0.549, 0).$$

The dot product is

$$x_{D_1}^\top x_{D_2} = 0.135(0.203) = 0.0274.$$

The norms are

$$\|x_{D_1}\|_2 = \sqrt{0.135^2 + 0.270^2} \approx 0.302,$$

$$\|x_{D_2}\|_2 = \sqrt{0.203^2 + 0.549^2} \approx 0.585.$$

Therefore,

$$\cos(D_1, D_2) = \frac{0.0274}{0.302(0.585)} \approx 0.155.$$

$$\boxed{\cos(D_1, D_2) \approx 0.155.}$$

The documents share the term `data`, but their remaining weighted terms differ.

IV. Text Mining Tasks

Task	Purpose	Example
Text classification	Assign predefined categories	Spam versus non-spam
Sentiment analysis	Identify opinion or emotion	Positive versus negative review
Text clustering	Group similar documents	News grouping
Topic modeling	Discover hidden themes	Sports, politics, technology
Information extraction	Extract entities and relations	Person, company, location
Information retrieval	Rank relevant documents	Search-engine results

Worked Example: Multinomial Naive Bayes Classification

The multinomial Naive Bayes decision rule is

$$\boxed{\hat{c} = \arg\max_{c} P(c) \prod_{t \in d} P(t \mid c)}.$$

With Laplace smoothing,

$$P(t \mid c) = \frac{N_{tc} + 1}{N_c + |V|}.$$

Suppose the training documents are

Document	Text	Class
D_1	good useful product	Positive
D_2	good easy	Positive
D_3	bad difficult	Negative
D_4	bad useless product	Negative

The vocabulary has $|V| = 7$ terms. Each class contains five word occurrences, and

$$P(\text{Positive}) = P(\text{Negative}) = \frac{1}{2}.$$

For the test document

$$d = \text{“good easy”},$$

the positive-class probabilities are

$$P(\text{good} \mid \text{Positive}) = \frac{2+1}{5+7} = \frac{3}{12},$$

$$P(\text{easy} \mid \text{Positive}) = \frac{1+1}{5+7} = \frac{2}{12}.$$

The negative-class probabilities are

$$P(\text{good} \mid \text{Negative}) = \frac{0+1}{12} = \frac{1}{12},$$

$$P(\text{easy} \mid \text{Negative}) = \frac{0+1}{12} = \frac{1}{12}.$$

The unnormalized class scores are

$$S_+ = \frac{1}{2}\left(\frac{3}{12}\right)\left(\frac{2}{12}\right) = 0.02083,$$

$$S_- = \frac{1}{2}\left(\frac{1}{12}\right)\left(\frac{1}{12}\right) = 0.00347.$$

Since $S_+ > S_-$,

$$\boxed{\hat{c} = \text{Positive.}}$$

V. Web Mining

Web mining applies data-mining techniques to discover patterns and knowledge from web data. It has three main branches:

- **Web content mining:** analyzes page contents;
- **Web structure mining:** analyzes hyperlinks;
- **Web usage mining:** analyzes user interactions and logs.

1. Web Content Mining

Web content mining converts text, HTML, tables, images, multimedia, product information, and reviews into structured data.

Worked example: structured product extraction.

Suppose an e-commerce page contains the following records.

Product ID	Name	Price (Tk.)	Brand	Rating	Reviews
P_1	Wireless Mouse	1200	Alpha	4.5	240
P_2	Mechanical Keyboard	3500	Beta	4.7	180
P_3	USB Hub	900	Alpha	4.1	95

The average rating is

$$\bar{r} = \frac{4.5 + 4.7 + 4.1}{3} = 4.433.$$

The review-weighted rating is

$$r_w = \frac{4.5(240) + 4.7(180) + 4.1(95)}{240 + 180 + 95} = \frac{2315.5}{515} \approx 4.496.$$

$$\boxed{\bar{r} = 4.433, \qquad r_w \approx 4.496.}$$

The weighted result gives more influence to products supported by more reviews.

2. Web Structure Mining

Web pages and hyperlinks form a directed graph

$$\boxed{G = (V, E)},$$

where V is the set of pages and E is the set of directed hyperlinks.

Worked example: adjacency matrix and degrees.

Suppose the links are

$$A \to B, \quad A \to C, \quad B \to C, \quad C \to A, \quad D \to C.$$

The adjacency matrix, using source pages as rows, is

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Page	In-degree	Out-degree
A	1	2
B	1	1
C	3	1
D	0	1

Page C has the largest in-degree:

$$\boxed{\deg^-(C) = 3.}$$

However, PageRank also considers the importance and out-degree of linking pages.

3. PageRank

For page A,

$$\boxed{PR(A) = \frac{1-d}{N} + d \sum_{i \in \mathrm{In}(A)} \frac{PR(i)}{L(i)}},$$

where d is the damping factor, N is the number of pages, and $L(i)$ is the number of outgoing links from page i.

Worked example: two PageRank iterations.

Use the previous graph, $d = 0.85$, $N = 4$, and initialize

$$PR^{(0)}(A) = PR^{(0)}(B) = PR^{(0)}(C) = PR^{(0)}(D) = 0.25.$$

The teleportation term is

$$\frac{1-d}{N} = \frac{0.15}{4} = 0.0375.$$

First iteration:

$$PR^{(1)}(A) = 0.0375 + 0.85\left(\frac{0.25}{1}\right) = 0.2500,$$

$$PR^{(1)}(B) = 0.0375 + 0.85\left(\frac{0.25}{2}\right) = 0.14375,$$

$$PR^{(1)}(C) = 0.0375 + 0.85\left(\frac{0.25}{2} + \frac{0.25}{1} + \frac{0.25}{1}\right) = 0.56875,$$

$$PR^{(1)}(D) = 0.0375.$$

Second iteration:

$$PR^{(2)}(A) = 0.0375 + 0.85(0.56875) = 0.52094,$$

$$PR^{(2)}(B) = 0.0375 + 0.85\left(\frac{0.2500}{2}\right) = 0.14375,$$

$$PR^{(2)}(C) = 0.0375 + 0.85\left(\frac{0.2500}{2} + 0.14375 + 0.0375\right) = 0.29781,$$

$$PR^{(2)}(D) = 0.0375.$$

Page	$PR^{(0)}$	$PR^{(1)}$	$PR^{(2)}$
A	0.2500	0.2500	0.52094
B	0.2500	0.14375	0.14375
C	0.2500	0.56875	0.29781
D	0.2500	0.03750	0.03750

At iteration two,

$$\boxed{PR^{(2)}(A) = 0.52094}$$

is the largest value. Further iterations continue until the ranks converge.

4. Web Usage Mining

Web usage mining analyzes logs containing user ID, time, URL, device, referrer, and action. Important preprocessing steps include cleaning, user identification, sessionization, and path construction.

Worked example: navigation support and confidence.

Suppose four sessions contain the following navigation paths.

Session	Navigation sequence
S_1	$A \to B \to C$
S_2	$A \to B$
S_3	$A \to C$
S_4	$A \to B \to C$

The sequential pattern $A \to B$ appears in three of four sessions:

$$\text{support}(A \to B) = \frac{3}{4} = 0.75.$$

Because page A appears in all four sessions,

$$\text{confidence}(A \to B) = \frac{\text{support}(A \to B)}{\text{support}(A)} = \frac{3/4}{4/4} = 0.75.$$

Similarly,

$$\text{support}(B \to C) = \frac{2}{4} = 0.50,$$

$$\text{confidence}(B \to C) = \frac{2}{3} = 0.667.$$

Thus $A \to B$ is the stronger navigation rule under both support and confidence.

VI. Introduction to Big Data Analytics

Big data refers to data whose size, generation rate, or complexity makes conventional single-machine processing inefficient.

1. The 5Vs

Property	Meaning	Example
Volume	Amount of data	Video archives
Velocity	Rate of generation	IoT streams
Variety	Multiple formats	Text, images, tables
Veracity	Quality and reliability	Noisy sensor readings
Value	Useful insight	Fraud prevention

Worked example: data velocity and volume.

Suppose an IoT platform receives data at 500 MB/s.

Data generated per hour is

$$500 \times 3600 = 1{,}800{,}000 \text{ MB} = 1.8 \text{ TB}.$$

Data generated per day is

$$1.8 \times 24 = 43.2 \text{ TB}.$$

For a 30-day month,

$$43.2 \times 30 = 1296 \text{ TB} = 1.296 \text{ PB}.$$

Period	Generated data
1 second	500 MB
1 hour	1.8 TB
1 day	43.2 TB
30 days	1.296 PB

This example demonstrates both high **velocity** and large **volume**.

VII. Hadoop Architecture

Apache Hadoop provides distributed storage through HDFS, cluster-resource management through YARN, and batch processing through MapReduce.

1. HDFS Blocks and Replication

If a file of size F is divided into blocks of size B, the required number of blocks is

$$\boxed{N_B = \left\lceil \frac{F}{B} \right\rceil}.$$

With replication factor r, the physical amount of stored data is approximately

$$\boxed{S_{\text{physical}} = rF}.$$

Worked example: HDFS storage.

Let

$$F = 750 \text{ MB}, \qquad B = 128 \text{ MB}, \qquad r = 3.$$

The number of blocks is

$$N_B = \left\lceil \frac{750}{128} \right\rceil = \lceil 5.859 \rceil = 6.$$

The first five blocks contain

$$5(128) = 640 \text{ MB}.$$

The final block contains

$$750 - 640 = 110 \text{ MB}.$$

The total number of block replicas is

$$6 \times 3 = 18.$$

The approximate physical storage is

$$3(750) = 2250 \text{ MB} = 2.25 \text{ GB}.$$

6 logical blocks, 18 replicas, 2.25 GB stored.

2. YARN Resource Allocation

YARN manages cluster resources and schedules application containers.

Worked example: maximum concurrent containers.

Suppose a cluster has eight worker nodes. Each node has 16 CPU cores and 64 GB memory. One core and 4 GB are reserved on each node.

Available CPU cores are

$$C = 8(16 - 1) = 120.$$

Available memory is

$$M = 8(64 - 4) = 480 \text{ GB}.$$

Each application container requires four cores and 8 GB memory. The CPU limit is

$$N_{\text{CPU}} = \left\lfloor \frac{120}{4} \right\rfloor = 30.$$

The memory limit is

$$N_{\text{memory}} = \left\lfloor \frac{480}{8} \right\rfloor = 60.$$

Therefore,

$$N_{\text{containers}} = \min(30, 60) = 30.$$

30 containers can run concurrently.

CPU is the limiting resource.

3. Data Locality

If the same job processes F GB on n workers, ideal balanced local input per worker is

$$\boxed{F_{\text{worker}} = \frac{F}{n}}.$$

Worked example: local versus network processing.

For a 1.2 TB data set on six workers,

$$F_{\text{worker}} = \frac{1.2}{6} = 0.2 \text{ TB} = 200 \text{ GB}.$$

If every worker reads its local block, network transfer for input can approach zero. If all data must instead move through the network, as much as 1.2 TB may cross the network. This illustrates Hadoop's principle of moving computation close to data.

VIII. MapReduce and Apache Spark

1. MapReduce Word Count

MapReduce processes key-value pairs through map, shuffle/sort, and reduce stages.

$$\boxed{\text{Input} \rightarrow \text{Map} \rightarrow \text{Shuffle/Sort} \rightarrow \text{Reduce} \rightarrow \text{Output}.}$$

Worked example: distributed word count.

Input records:

Record	Text
R_1	data mining data
R_2	web data mining
R_3	big data analytics

Map output:

$$R_1 \rightarrow (\text{data}, 1), (\text{mining}, 1), (\text{data}, 1),$$

$$R_2 \rightarrow (\text{web}, 1), (\text{data}, 1), (\text{mining}, 1),$$

$$R_3 \rightarrow (\text{big}, 1), (\text{data}, 1), (\text{analytics}, 1).$$

After shuffle and sort:

Key	Grouped values	Reduced output
analytics	[1]	(analytics, 1)
big	[1]	(big, 1)
data	[1, 1, 1, 1]	(data, 4)
mining	[1, 1]	(mining, 2)
web	[1]	(web, 1)

Therefore,

$$\boxed{\text{data} : 4, \quad \text{mining} : 2, \quad \text{web} : 1, \quad \text{big} : 1, \quad \text{analytics} : 1.}$$

2. Apache Spark Transformations and Actions

Spark builds a computation from transformations such as `map` and `filter`, then evaluates it when an action such as `reduce`, `count`, or `collect` is requested.

Worked example: transformation pipeline.

Input data:

$$X = [2, 3, 4, 5, 6].$$

Square every value:

$$X_1 = \text{map}(x \mapsto x^2) = [4, 9, 16, 25, 36].$$

Keep values greater than 10:

$$X_2 = \text{filter}(x > 10) = [16, 25, 36].$$

Reduce by summation:

$$S = 16 + 25 + 36 = 77.$$

$$\boxed{\text{reduce}(X_2, +) = 77.}$$

3. Hadoop MapReduce versus Spark

Feature	Hadoop MapReduce	Apache Spark
Processing	Primarily disk-based	Supports in-memory reuse
Iteration	Repeated disk I/O	Efficient iterative computation
Batch processing	Strong	Strong
Streaming	Not the primary model	Structured Streaming
Machine learning	External ecosystem tools	MLlib
Typical use	Large batch jobs	Interactive, iterative, and streaming jobs

Worked example: simplified iterative I/O comparison.

Suppose an iterative algorithm processes a 200 GB data set for five iterations. Assume that MapReduce reads and writes the full data once per iteration.

MapReduce disk I/O is

$$I_{\text{MR}} = 5(200 + 200) = 2000 \text{ GB}.$$

Assume Spark reads the data once, reuses it in memory, and writes the final result once:

$$I_{\text{Spark}} = 200 + 200 = 400 \text{ GB}.$$

The simplified I/O reduction is

$$\frac{2000 - 400}{2000} \times 100\% = 80\%.$$

Spark reduces disk I/O by 80% under these assumptions.

This example explains why Spark is often more suitable for iterative machine-learning workloads.

IX. Method Selection and Practice

1. Selection Guide

Problem	Strong starting method
Convert cleaned documents to count vectors	Bag-of-Words
Weight frequent but collection-rare terms	TF-IDF
Compare document-vector directions	Cosine similarity
Classify labeled text	Naive Bayes, logistic regression, SVM, or neural network
Extract attributes from web pages	Web content mining
Rank linked web pages	PageRank
Discover navigation behavior	Web usage mining
Store very large replicated files	HDFS
Manage cluster resources	YARN
Run reliable large batch jobs	MapReduce
Run iterative or interactive analytics	Spark

2. Practice Problems

1. A five-word sentence is converted into bigrams and trigrams. Calculate the number of each.
2. In a collection of 10 documents, the term `mining` appears in four documents. Calculate its IDF using the natural logarithm.
3. A term occurs three times in an eight-word document. If its IDF is 1.2, calculate TF-IDF.
4. For vectors $x = (1, 2, 0)$ and $y = (2, 1, 0)$, calculate cosine similarity.
5. A file is 900 MB and the HDFS block size is 128 MB. Find the number of blocks.
6. With replication factor 3, calculate the physical storage required for the 900 MB file.
7. A web pattern appears in 30 of 50 sessions. Calculate support.
8. A cluster receives 2 TB per day. Calculate its 30-day data volume.

Answer check.

1. Bigrams: $5 - 2 + 1 = 4$; trigrams: $5 - 3 + 1 = 3$.

2. IDF $= \log(10/4) = \log(2.5) \approx 0.916$.
3. TF $= 3/8 = 0.375$; TF-IDF $= 0.375(1.2) = 0.45$.
4. $x^\top y = 4$, $\|x\| = \|y\| = \sqrt{5}$; cosine $= 4/5 = 0.8$.
5. $\lceil 900/128 \rceil = \lceil 7.03125 \rceil = 8$ blocks.
6. $3(900) = 2700$ MB $= 2.7$ GB.
7. Support $= 30/50 = 0.60$.
8. $2(30) = 60$ TB.

X. Key Takeaways

1. Text preprocessing converts raw language into consistent tokens and numerical features.
2. Bag-of-Words records counts, while TF-IDF reduces the influence of common collection-wide terms.
3. Cosine similarity compares document directions rather than raw magnitudes.
4. Web mining studies page content, hyperlink structure, and user behavior.
5. PageRank distributes importance through incoming links and outgoing-link normalization.
6. Big-data systems address volume, velocity, variety, veracity, and value through distributed storage and computation.
7. HDFS stores replicated blocks, YARN allocates resources, MapReduce provides disk-oriented batch processing, and Spark supports reusable in-memory computation.

Prepared By
Md. Atikuzzaman
Lecturer, Department of Computer Science and Engineering
Green University of Bangladesh
Email: atik@cse.green.edu.bd