

Text Similarity and Clustering

Text Summarization and Topic Models

1121AITA07

MBA, IM, NTPU (M5265) (Fall 2023)

Tue 2, 3, 4 (9:10-12:00) (B3F17)



<https://meet.google.com/miy-fbif-max>



Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

<https://web.ntpu.edu.tw/~myday>

2023-11-08; 2023-11-15



Syllabus

Week	Date	Subject/Topics
1	2023/09/13	Introduction to Artificial Intelligence for Text Analytics
2	2023/09/20	Foundations of Text Analytics: Natural Language Processing (NLP)
3	2023/09/27	Python for Natural Language Processing
4	2023/10/04	Natural Language Processing with Transformers
5	2023/10/11	Case Study on Artificial Intelligence for Text Analytics I
6	2023/10/18	Text Classification and Sentiment Analysis

Syllabus

Week	Date	Subject/Topics
7	2023/10/25	Multilingual Named Entity Recognition (NER)
8	2023/11/01	Midterm Project Report
9	2023/11/08	Text Similarity and Clustering
10	2023/11/15	Text Summarization and Topic Models
11	2023/11/22	Text Generation with Large Language Models (LLMs)
12	2023/11/29	Case Study on Artificial Intelligence for Text Analytics II

Syllabus

Week Date Subject/Topics

13 2023/12/06 Question Answering and Dialogue Systems

14 2023/12/13 Deep Learning, Generative AI, Transfer Learning, Zero-Shot, and Few-Shot Learning for Text Analytics

15 2023/12/20 Final Project Report I

16 2023/12/27 Final Project Report II

17 2024/01/03 Self-learning

18 2024/01/10 Self-learning

Text Similarity
Text Clustering
Text Summarization
Topic Models

Outline

- **Text Similarity**

- Analyzing and quantifying the likeness between text documents.

- **Text Clustering**

- Grouping similar text documents using various algorithms.

- **Text Summarization**

- Condensing text data into a shorter, coherent form.

- **Topic Models**

- Identifying underlying themes or topics within text collections.

Text Similarity and Clustering

Text Similarity and Clustering

**Text Dataset
(Unsupervised)**

Text Pre-Processing

**Feature Extraction
(Vectorization) (TF-IDF)(Embedding)**

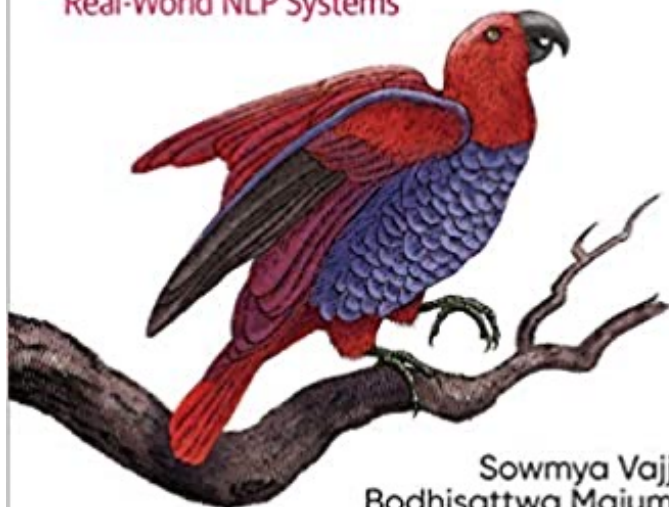
Text Similarity

Text Clustering

O'REILLY®

Practical Natural Language Processing

A Comprehensive Guide to Building Real-World NLP Systems



Sowmya Vajjala,
Bodhisattwa Majumder,
Anuj Gupta & Harshit Surana

FOUNDATIONS

Covered in
Chapters 1 to 3



ML for NLP



NLP Pipelines



Data
Gathering



Multilingual
NLP



Text
Representation

CORE TASKS

Covered in
Chapters 3 to 7



Text
Classification



Information
Extraction



Conversational
Agents



Information
Retrieval



Question
Answering

GENERAL APPLICATIONS

Covered in
Chapters 4 to 7



Spam
Classification



Calendar Event
Extractor



Personal
Assistants



Search
Engines

JEOPARDY!

Jeopardy!

INDUSTRY SPECIFIC

Covered in
Chapters 8 to 10



Social Media
Analysis



Retail Data
Extraction



Health Records
Analysis



Financial
Analysis



Legal Entity
Extraction

AI PROJECT PLAYBOOK

Covered in
Chapters 2 & 11



Project
Processes



Best
Practices



Model
Iterations



MLOps



AI Teams
& Hiring

Text Similarity and Clustering

- How do we measure **similarity** between terms and documents?
- How can we use **distance** measures to find the most **relevant documents**?
- How can we build a **recommender system** from **text similarity**?
- How do we **group similar documents (document clustering)**?

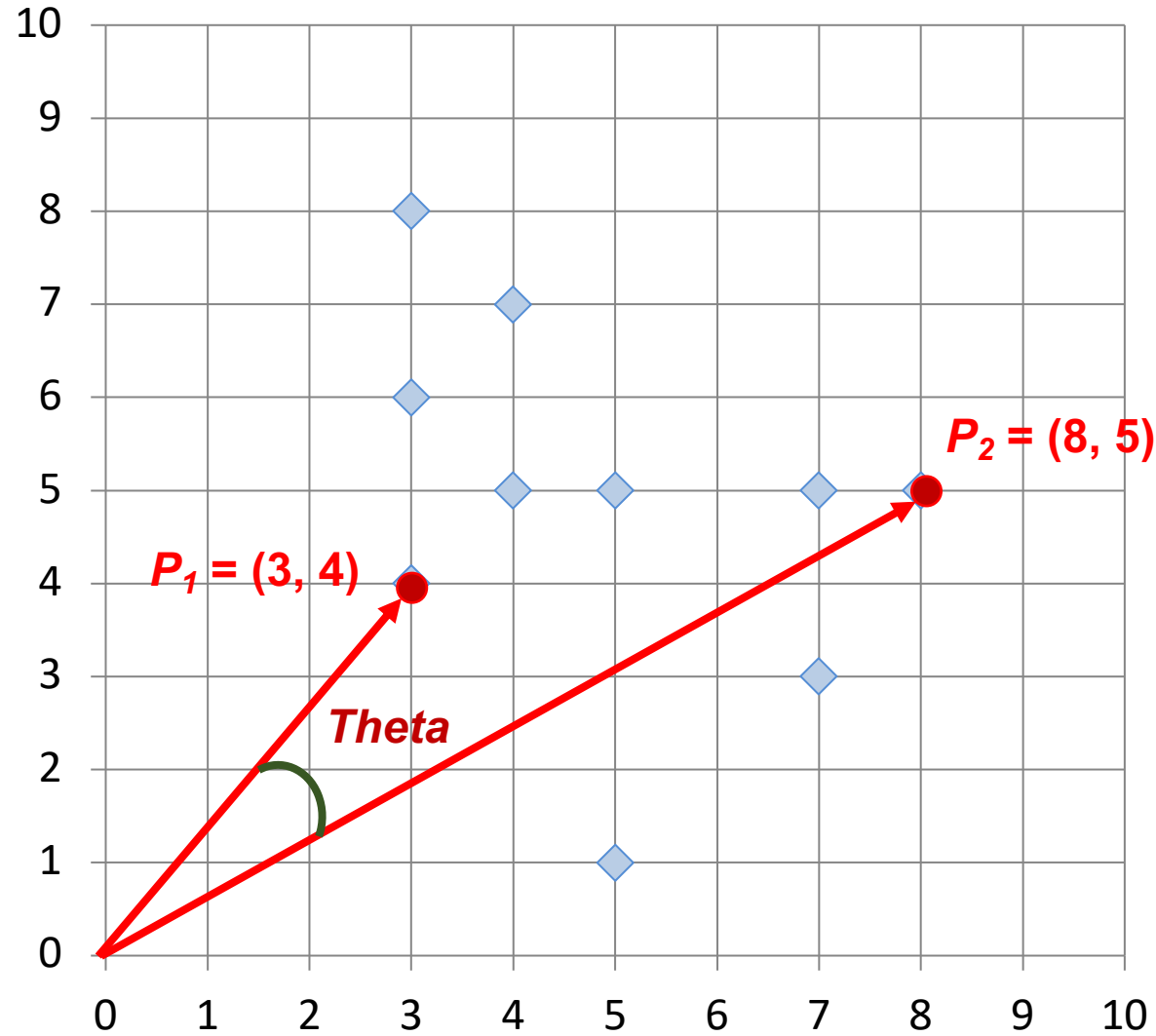
Text Similarity and Clustering

- **Information Retrieval (IR)**
- **Feature Engineering**
- **Similarity Measures**
- **Unsupervised Machine Learning Algorithms**

Text Similarity

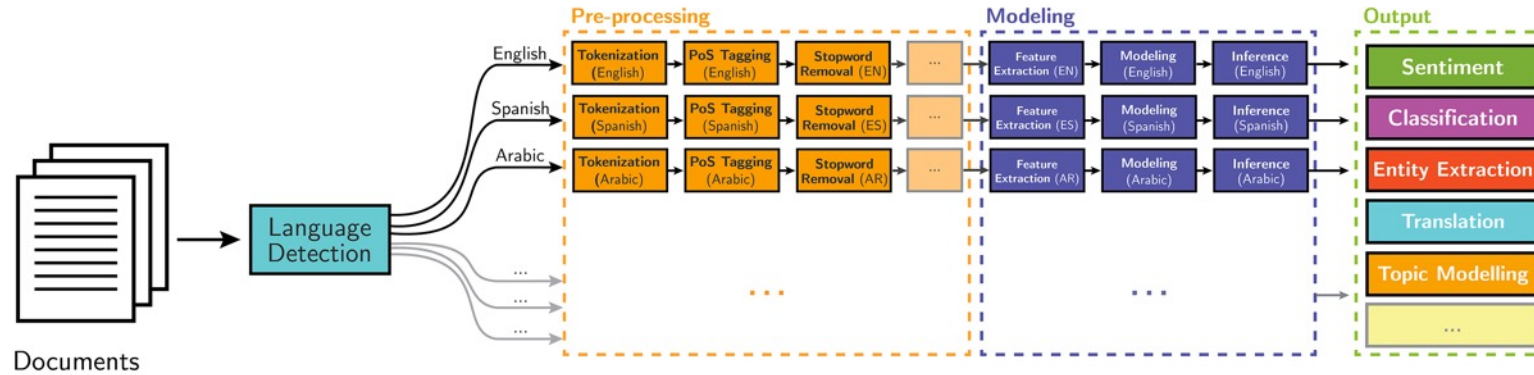
- **Lexical similarity**
 - **Syntax, structure, and content of the documents**
- **Semantic similarity**
 - **Semantics, meaning, and context of the documents**

Cosine Similarity

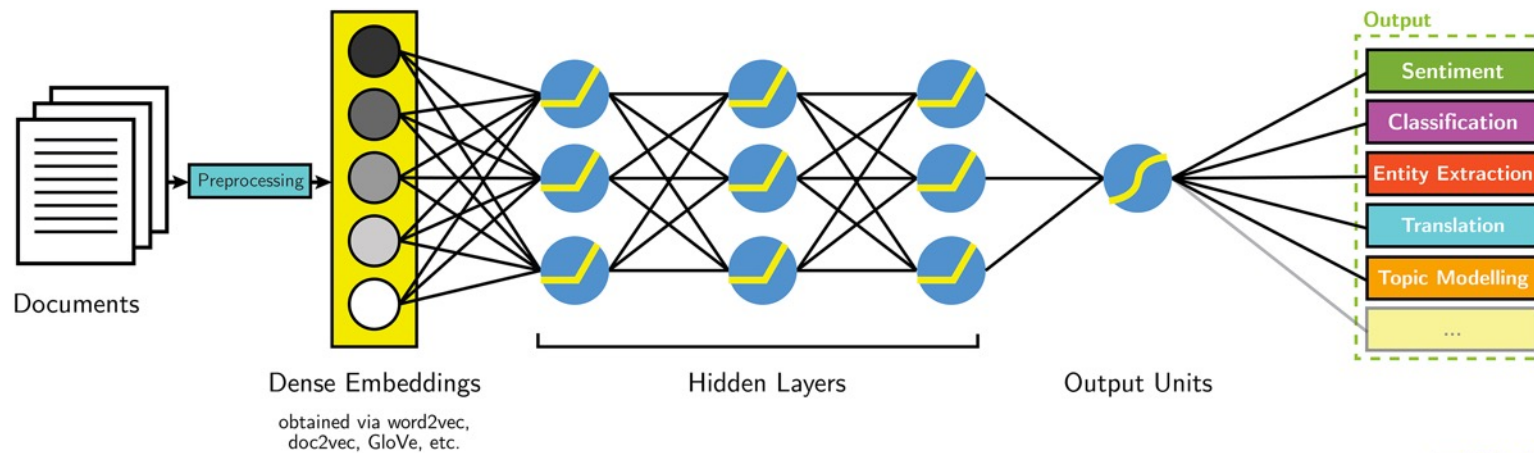


NLP

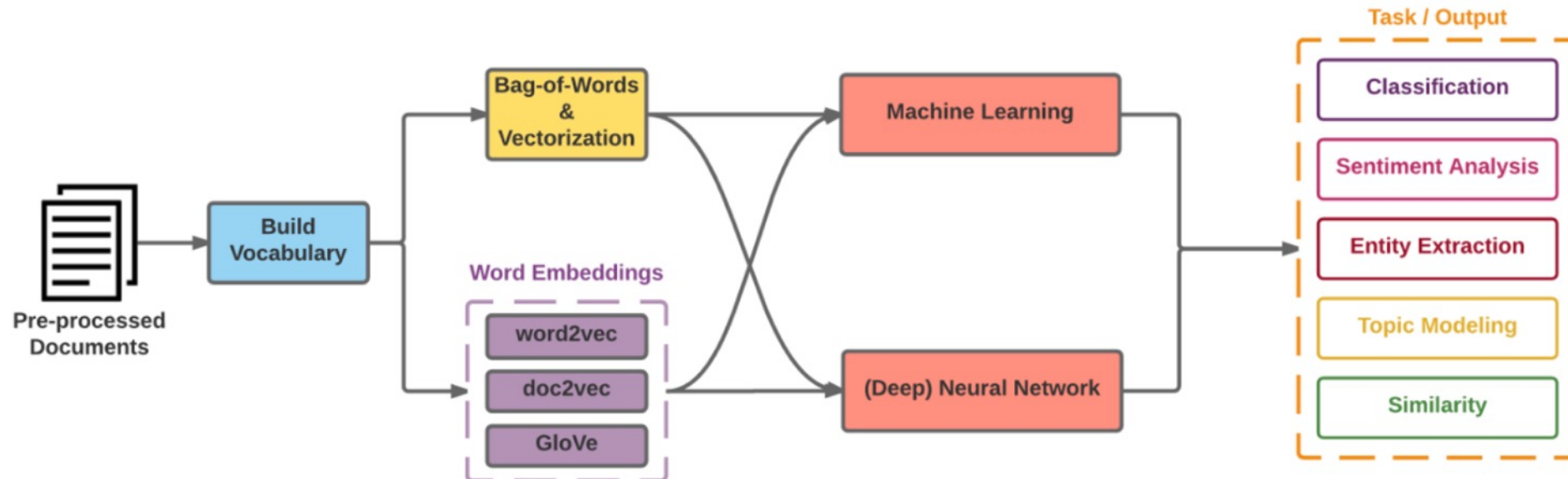
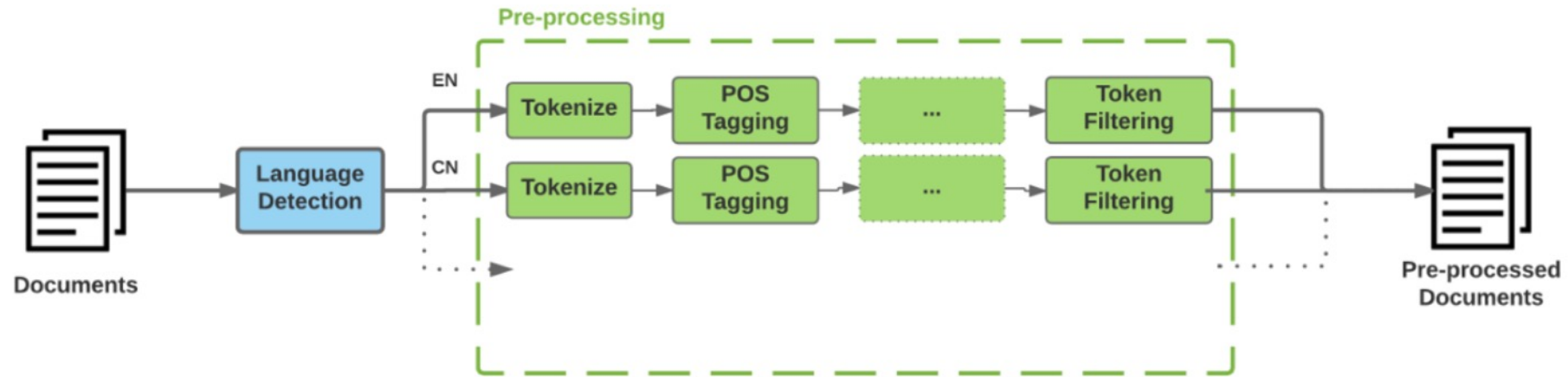
Classical NLP



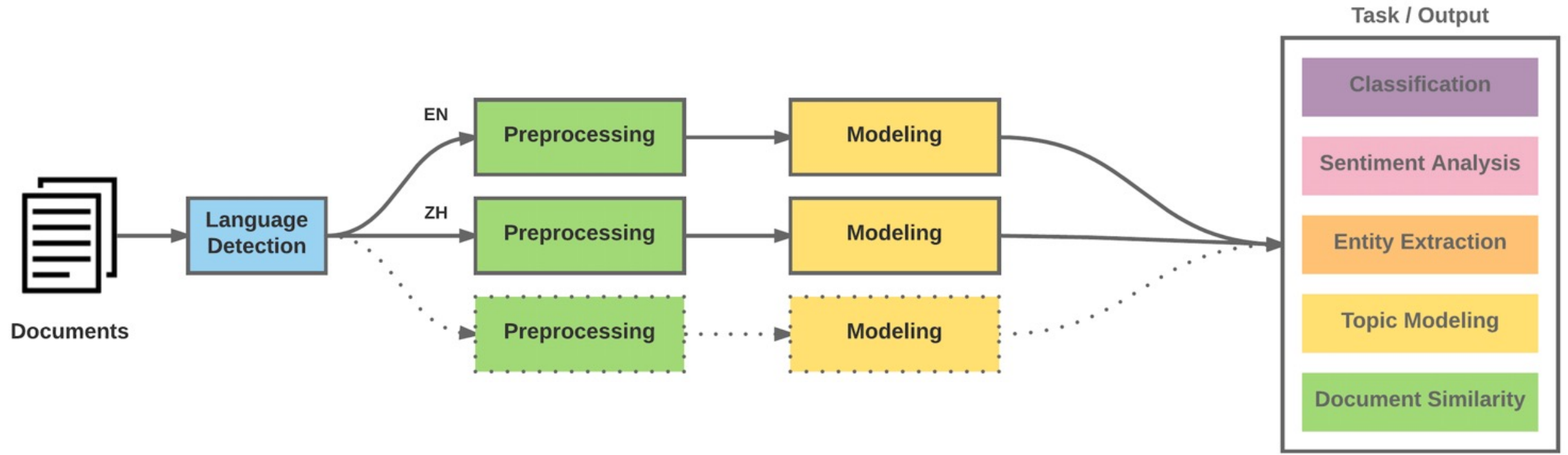
Deep Learning-based NLP



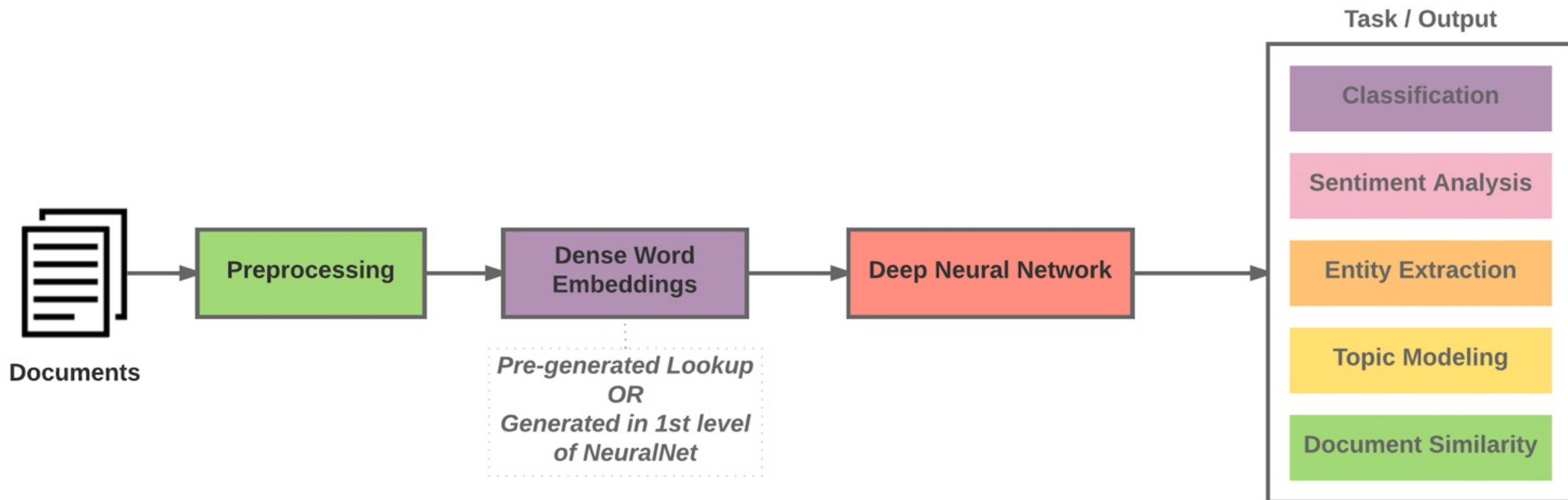
Modern NLP Pipeline



Modern NLP Pipeline

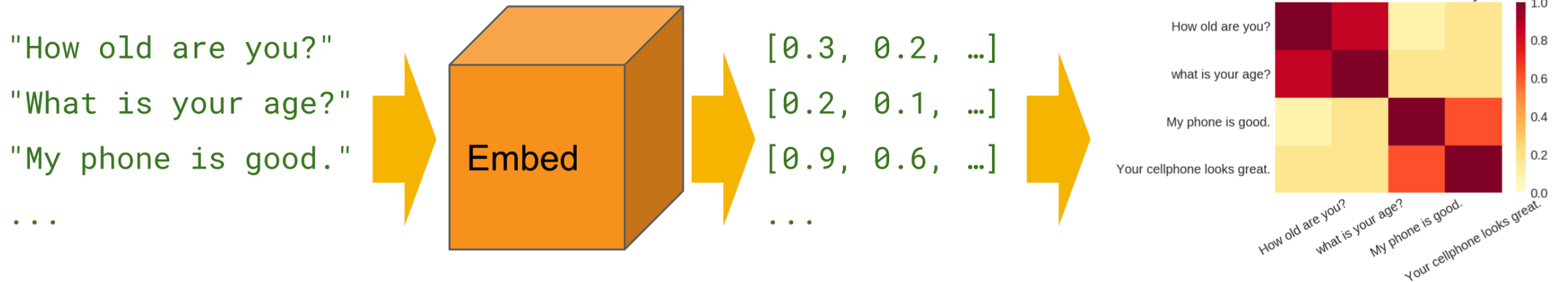


Deep Learning NLP



Text Similarity

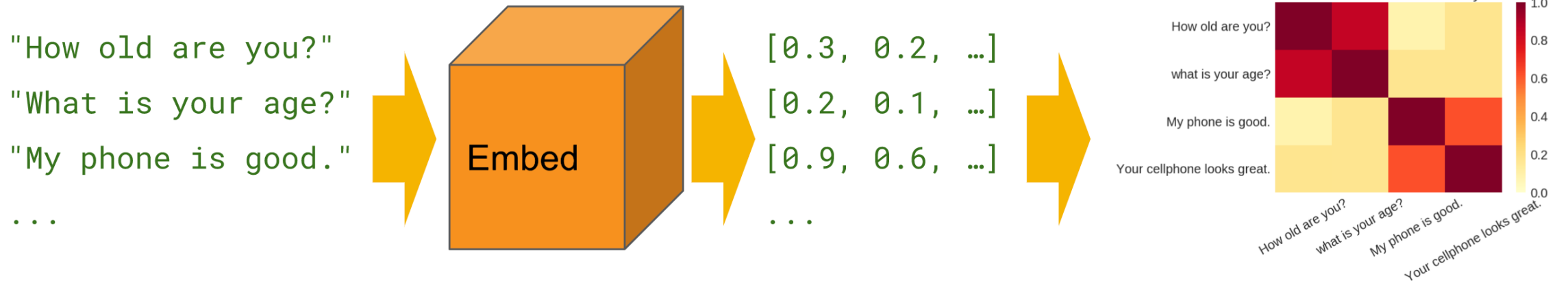
Semantic Similarity



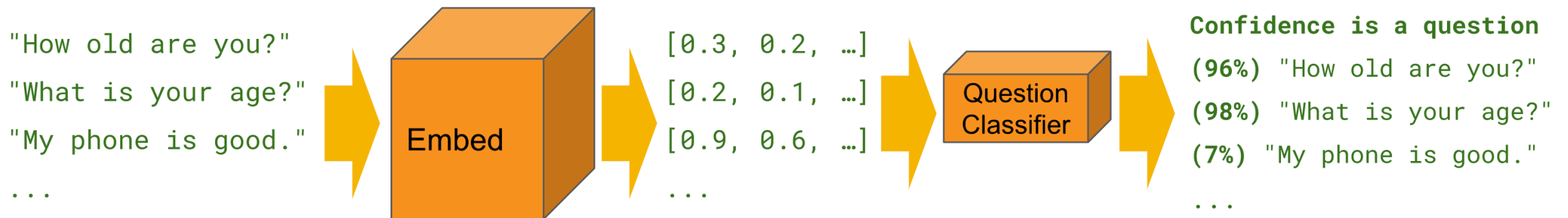
Text Similarity

Text Classification

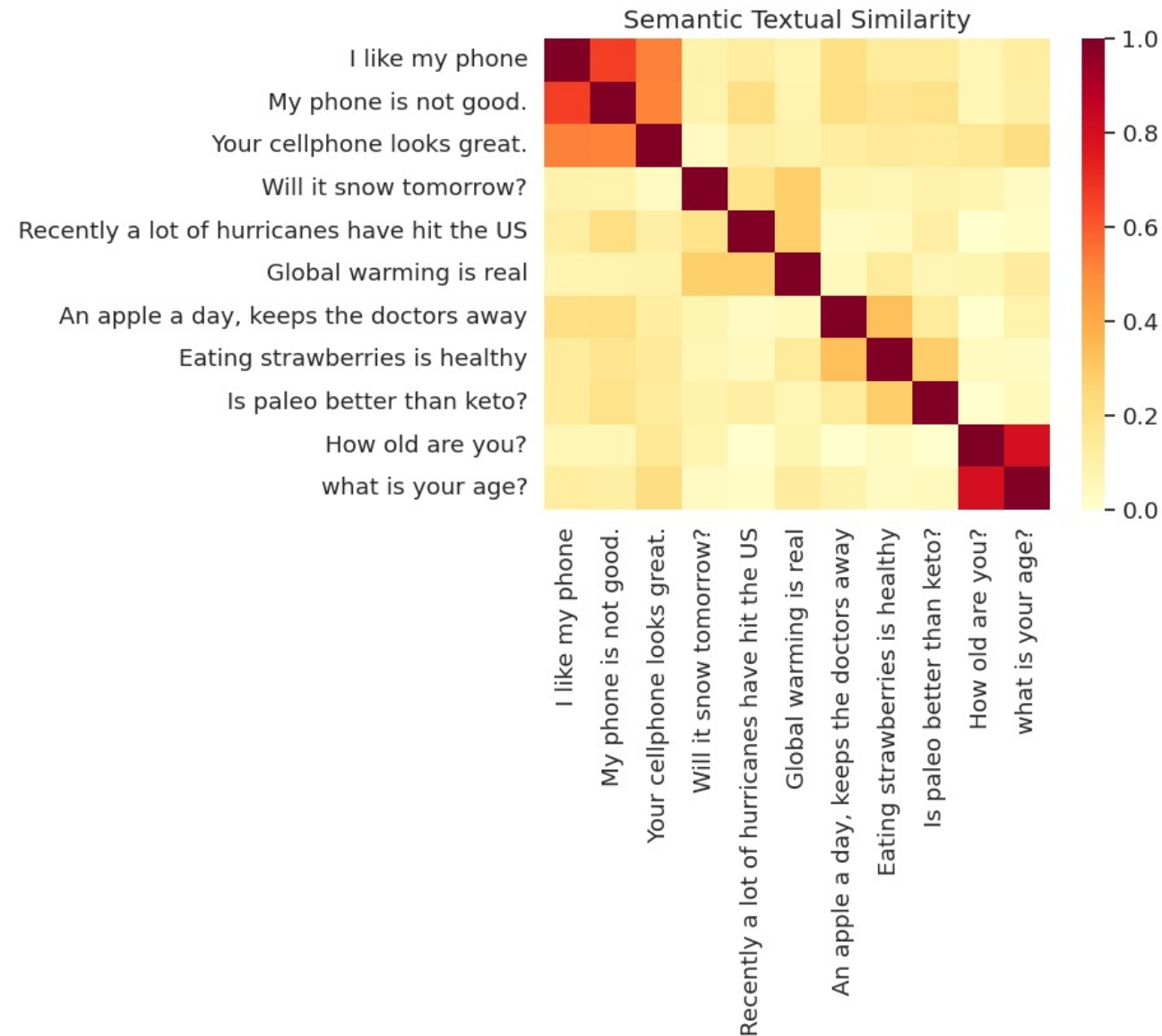
Semantic Similarity



Classification



Semantic Textual Similarity



Natural Language Processing (NLP) and Text Mining

Raw text

Sentence Segmentation

Tokenization

Part-of-Speech (POS)

Stop word removal

Stemming / **Lemmatization**

Dependency Parser

String Metrics & Matching

word's stem

am → am

having → hav

word's lemma

am → be

having → have

Data Mining Tasks & Methods

Data Mining Tasks & Methods	Data Mining Algorithms	Learning Type
Prediction		
Classification	Decision Trees, Neural Networks, Support Vector Machines, kNN, Naïve Bayes, GA	Supervised
Regression	Linear/Nonlinear Regression, ANN, Regression Trees, SVM, kNN, GA	Supervised
Time series	Autoregressive Methods, Averaging Methods, Exponential Smoothing, ARIMA	Supervised
Association		
Market-basket	Apriori, OneR, ZeroR, Eclat, GA	Unsupervised
Link analysis	Expectation Maximization, Apriori Algorithm, Graph-Based Matching	Unsupervised
Sequence analysis	Apriori Algorithm, FP-Growth, Graph-Based Matching	Unsupervised
Segmentation		
Clustering	k-means, Expectation Maximization (EM)	Unsupervised
Outlier analysis	k-means, Expectation Maximization (EM)	Unsupervised

Example of Cluster Analysis

Point	P	P(x,y)
p01	a	(3, 4)
p02	b	(3, 6)
p03	c	(3, 8)
p04	d	(4, 5)
p05	e	(4, 7)
p06	f	(5, 1)
p07	g	(5, 5)
p08	h	(7, 3)
p09	i	(7, 5)
p10	j	(8, 5)

K-Means Clustering

Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

m1 (3.67, 5.83)

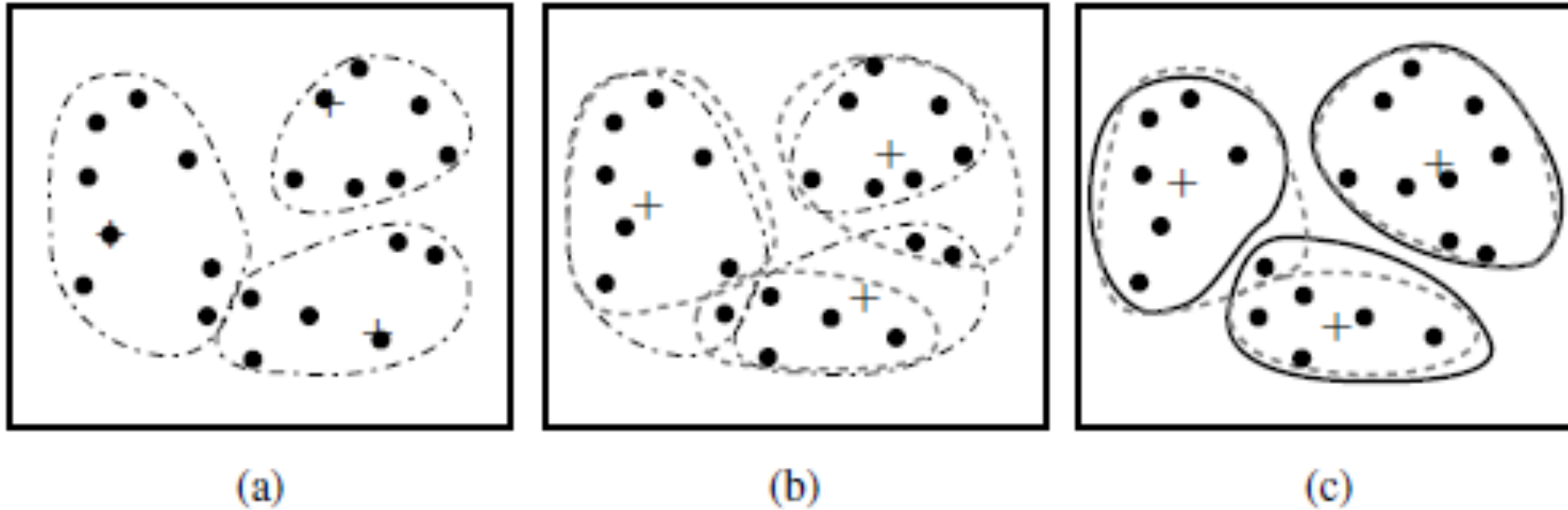
m2 (6.75, 3.50)

Cluster Analysis

Cluster Analysis

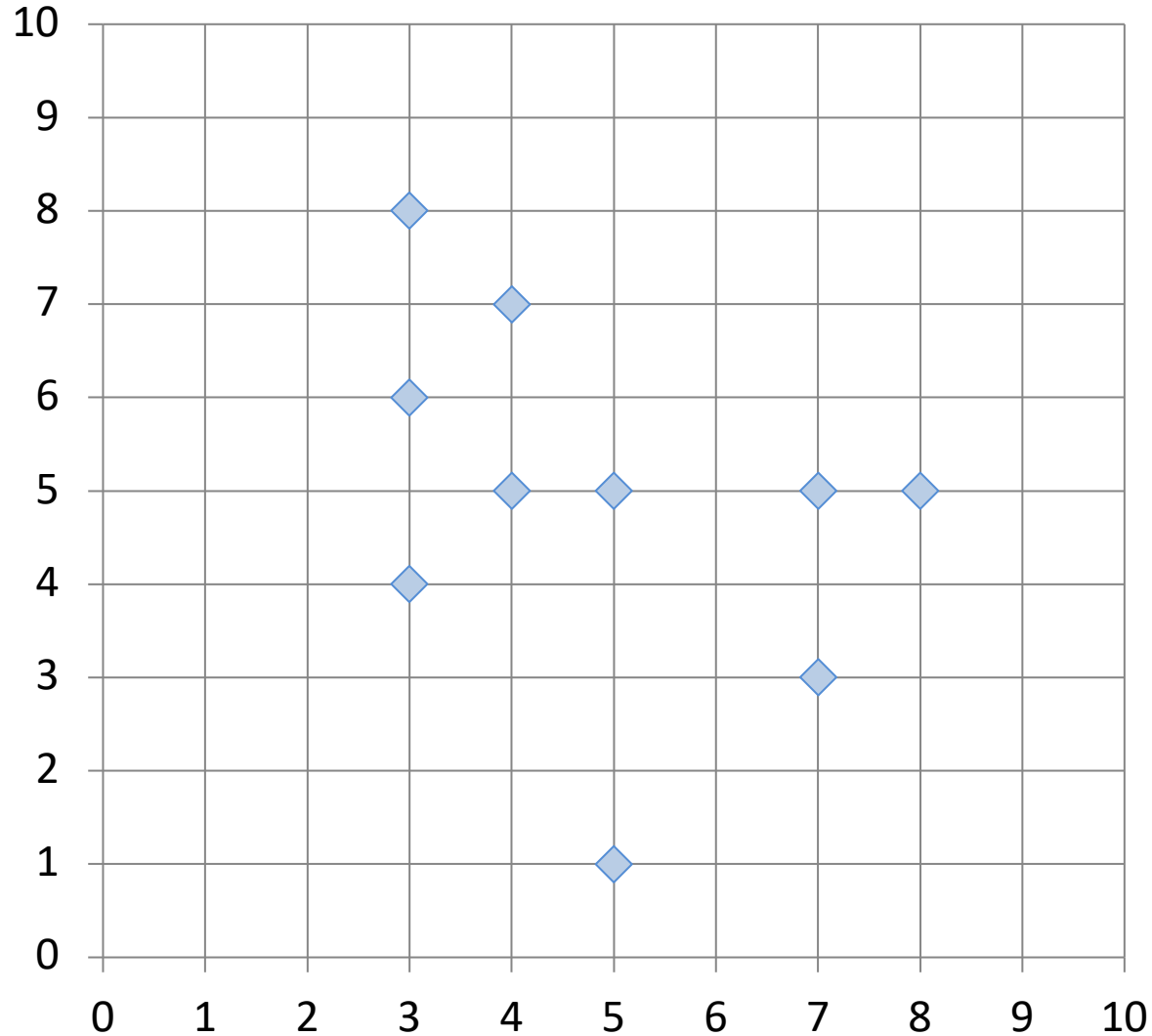
- Used for automatic identification of **natural groupings** of things
- Part of the machine-learning family
- Employ **unsupervised learning**
- Learns the clusters of things from past data, then assigns new instances
- There is not an output variable
- Also known as **segmentation**

Cluster Analysis



Clustering of a set of objects based on the *k-means method*.
(The mean of each cluster is marked by a “+”.)

Example of Cluster Analysis



Point	P	P(x,y)
p01	a	(3, 4)
p02	b	(3, 6)
p03	c	(3, 8)
p04	d	(4, 5)
p05	e	(4, 7)
p06	f	(5, 1)
p07	g	(5, 5)
p08	h	(7, 3)
p09	i	(7, 5)
p10	j	(8, 5)

Cluster Analysis for Data Mining

- **How many clusters?**
 - There is not a “truly optimal” way to calculate it
 - **Heuristics are often used**
 1. Look at the sparseness of clusters
 2. **Number of clusters = $(n/2)^{1/2}$** (n: no of data points)
 3. Use Akaike information criterion (AIC)
 4. Use Bayesian information criterion (BIC)
- **Most cluster analysis methods involve the use of a **distance measure** to calculate the closeness between pairs of items**
 - **Euclidian versus Manhattan (rectilinear) distance**

***k*-Means Clustering Algorithm**

- ***k*** : pre-determined number of clusters
- Algorithm (**Step 0**: determine value of ***k***)

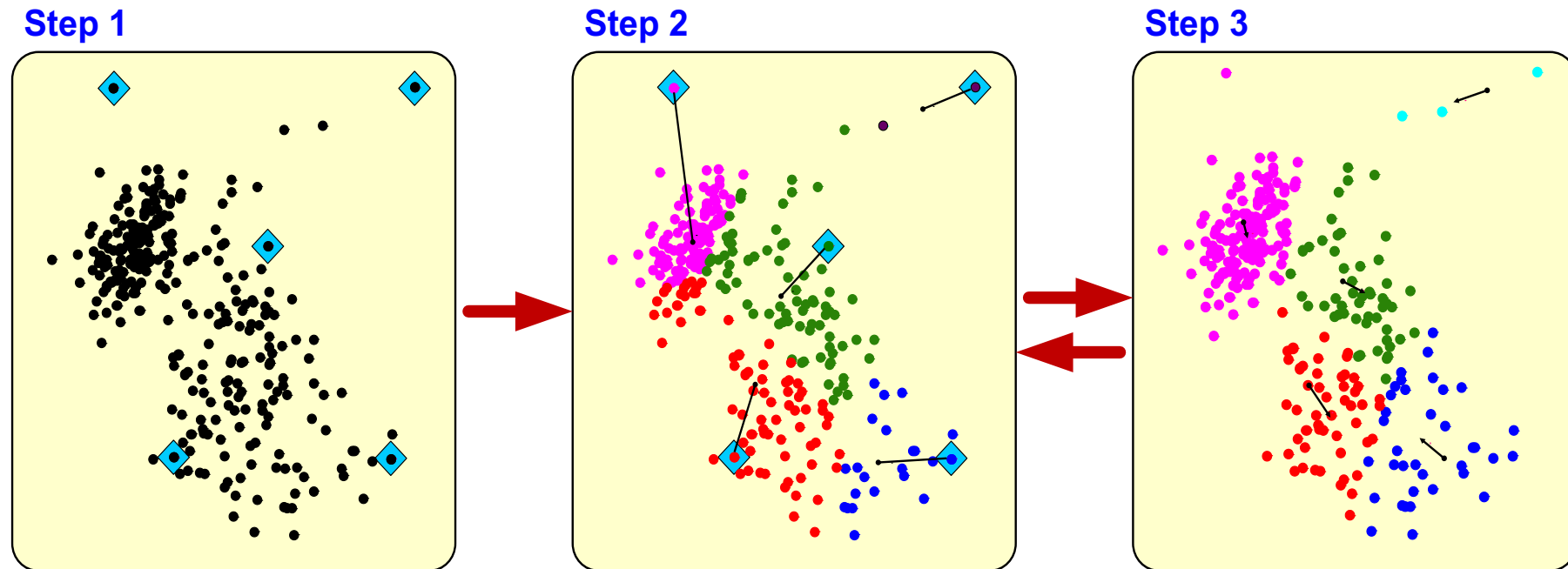
Step 1: Randomly generate ***k*** random points as initial cluster centers

Step 2: Assign each point to the nearest cluster center

Step 3: Re-compute the new cluster centers

Repetition step: Repeat steps 2 and 3 until some convergence criterion is met (usually that the assignment of points to clusters becomes stable)

Cluster Analysis for Data Mining - *k*-Means Clustering Algorithm



Similarity

Distance

Similarity and Dissimilarity Between Objects

- Distances are normally used to measure the similarity or dissimilarity between two data objects

- Some popular ones include: **Minkowski distance**:

$$d(i, j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \dots + |x_{ip} - x_{jp}|^q)}$$

where $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ are two p -dimensional data objects, and q is a positive integer

- If $q = 1$, d is **Manhattan distance**

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

Similarity and Dissimilarity Between Objects (Cont.)

- If $q = 2$, d is **Euclidean distance**:

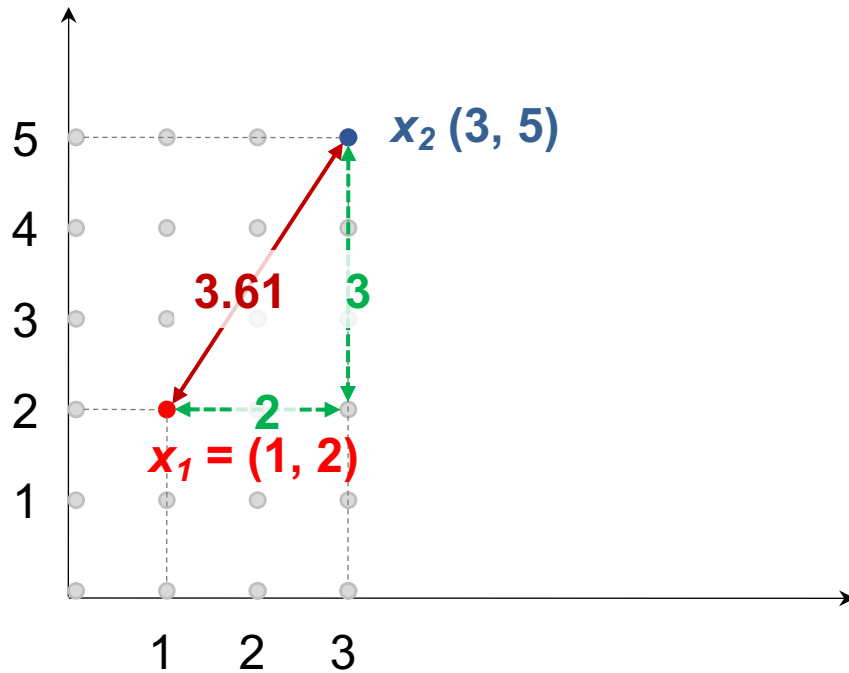
$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \dots + |x_{i_p} - x_{j_p}|^2)}$$

- **Properties**

- $d(i,j) \geq 0$
 - $d(i,i) = 0$
 - $d(i,j) = d(j,i)$
 - $d(i,j) \leq d(i,k) + d(k,j)$
- **Also, one can use weighted distance, parametric Pearson product moment correlation, or other dissimilarity measures**

Euclidean distance vs Manhattan distance

- Distance of two point $x_1 = (1, 2)$ and $x_2 (3, 5)$

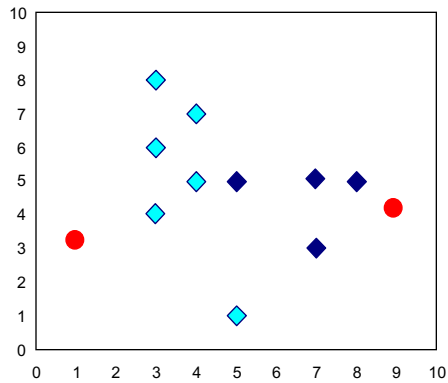


$$\begin{aligned} \text{Euclidean distance:} \\ &= ((3-1)^2 + (5-2)^2)^{1/2} \\ &= (2^2 + 3^2)^{1/2} \\ &= (4 + 9)^{1/2} \\ &= (13)^{1/2} \\ &= 3.61 \end{aligned}$$

$$\begin{aligned} \text{Manhattan distance:} \\ &= (3-1) + (5-2) \\ &= 2 + 3 \\ &= 5 \end{aligned}$$

The *K-Means* Clustering Method

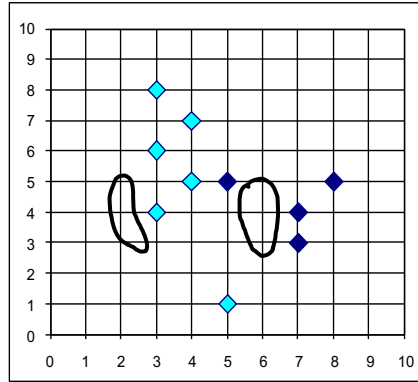
- **Example**



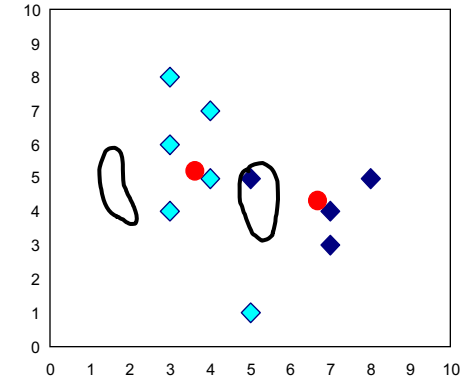
K=2

Arbitrarily choose K object as initial cluster center

Assign each object to most similar center



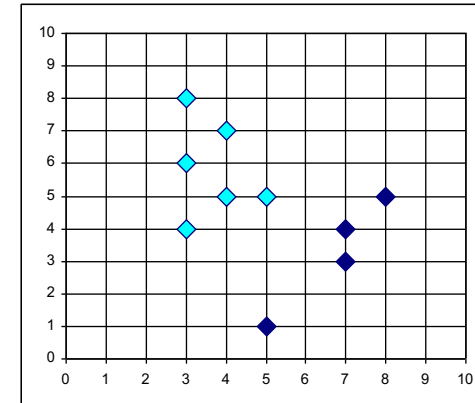
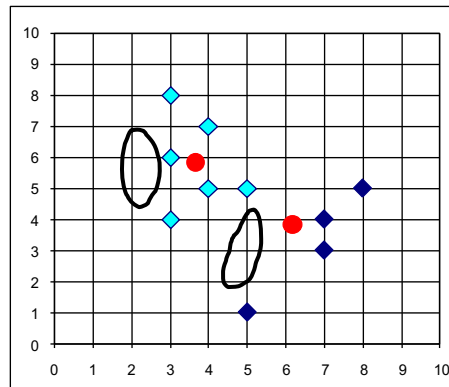
Update the cluster means



reassign

reassign

Update the cluster means



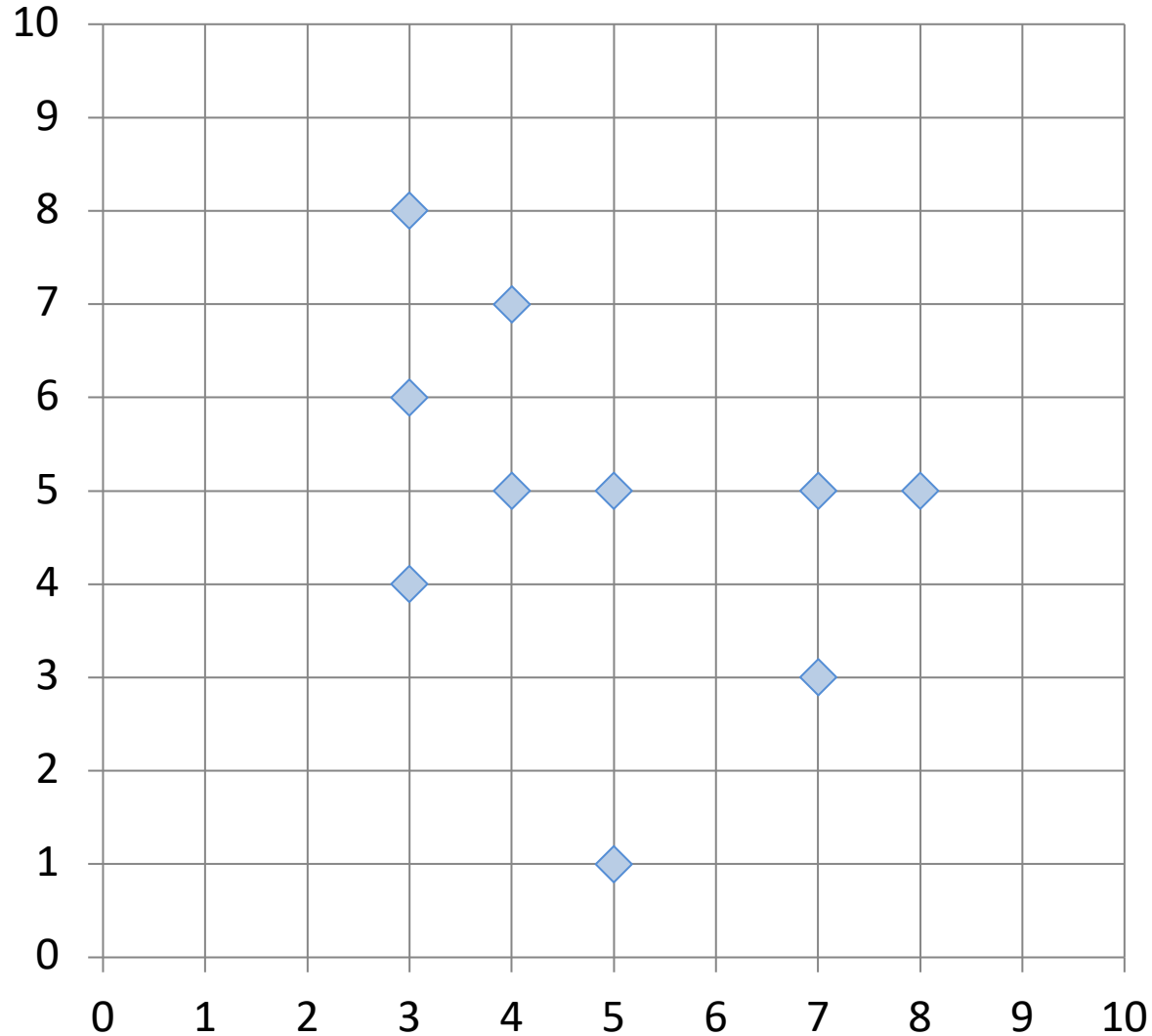
K-Means Clustering

Example of Cluster Analysis

Point	P	P(x,y)
p01	a	(3, 4)
p02	b	(3, 6)
p03	c	(3, 8)
p04	d	(4, 5)
p05	e	(4, 7)
p06	f	(5, 1)
p07	g	(5, 5)
p08	h	(7, 3)
p09	i	(7, 5)
p10	j	(8, 5)

K-Means Clustering

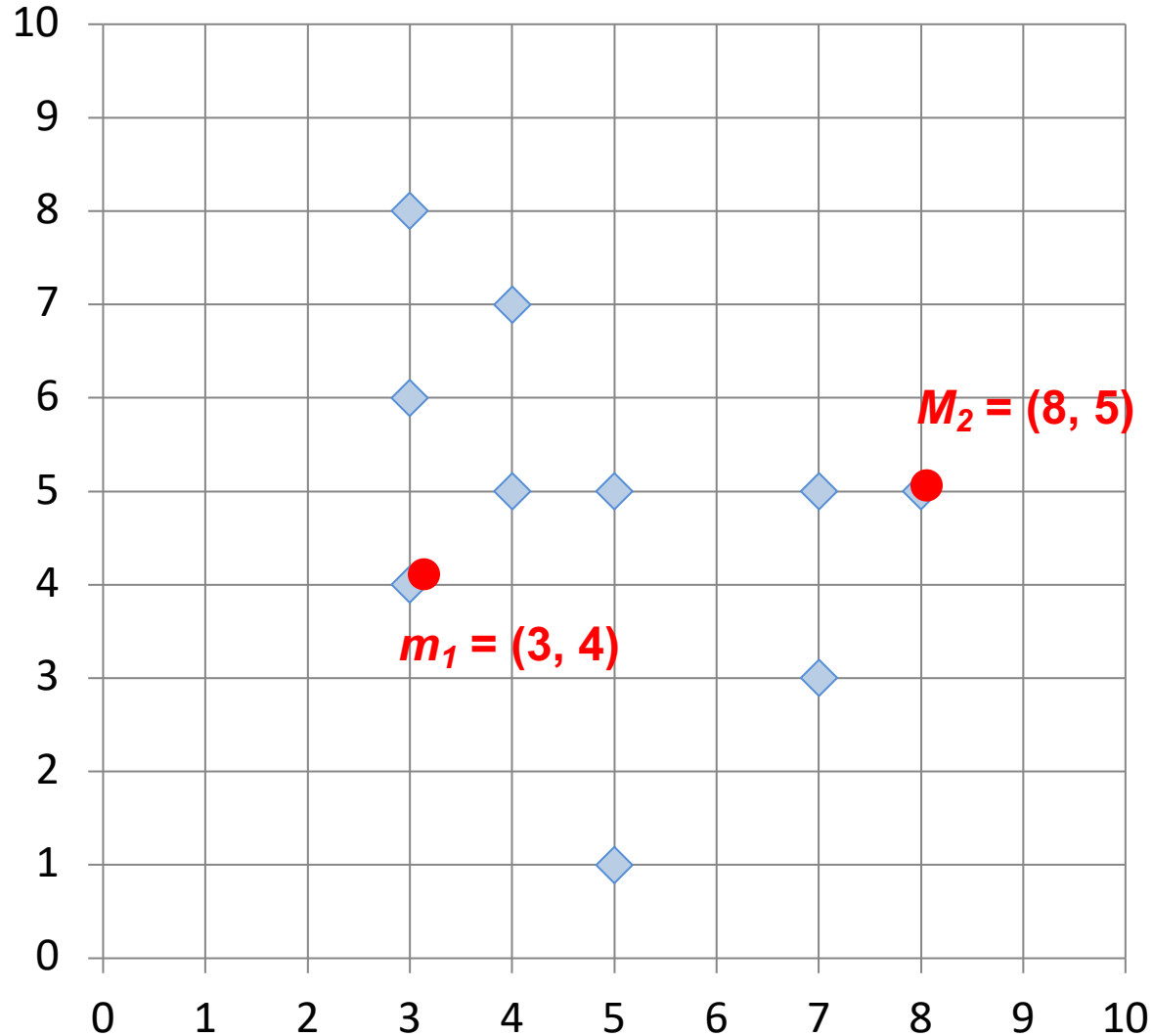
Step by Step



Point	P	P(x,y)
p01	a	(3, 4)
p02	b	(3, 6)
p03	c	(3, 8)
p04	d	(4, 5)
p05	e	(4, 7)
p06	f	(5, 1)
p07	g	(5, 5)
p08	h	(7, 3)
p09	i	(7, 5)
p10	j	(8, 5)

K-Means Clustering

Step 1: K=2, Arbitrarily choose K object as initial cluster center

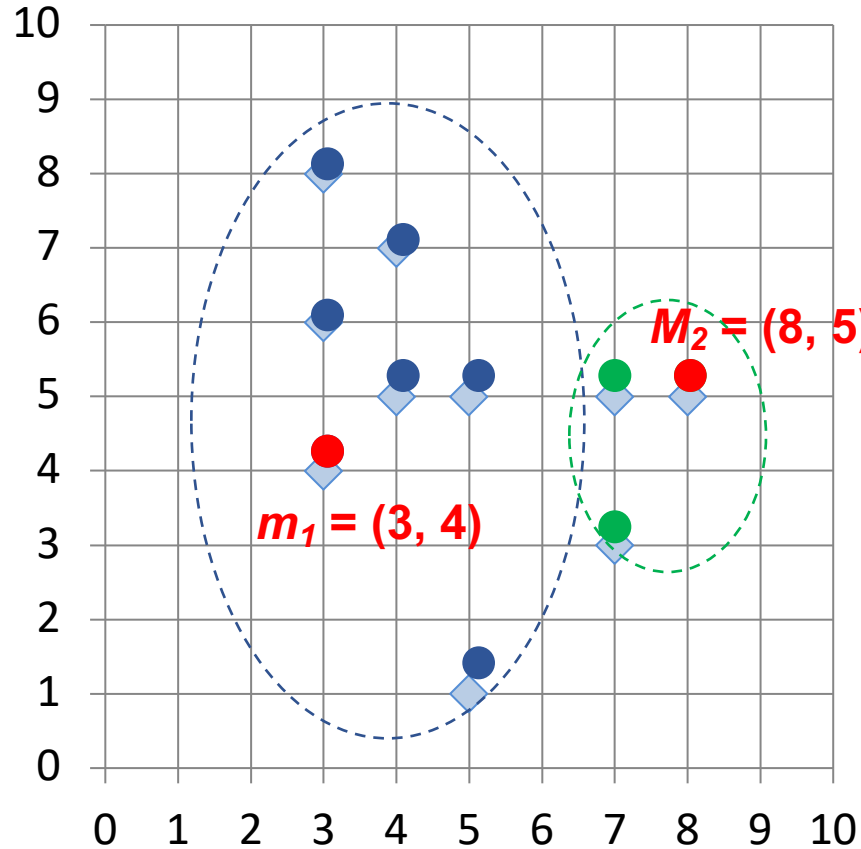


Point	P	P(x,y)
p01	a	(3, 4)
p02	b	(3, 6)
p03	c	(3, 8)
p04	d	(4, 5)
p05	e	(4, 7)
p06	f	(5, 1)
p07	g	(5, 5)
p08	h	(7, 3)
p09	i	(7, 5)
p10	j	(8, 5)

Initial m_1 (3, 4)
Initial m_2 (8, 5)

Step 2: Compute seed points as the centroids of the clusters of the current partition

Step 3: Assign each objects to most similar center



Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	0.00	5.10	Cluster1
p02	b	(3, 6)	2.00	5.10	Cluster1
p03	c	(3, 8)	4.00	5.83	Cluster1
p04	d	(4, 5)	1.41	4.00	Cluster1
p05	e	(4, 7)	3.16	4.47	Cluster1
p06	f	(5, 1)	3.61	5.00	Cluster1
p07	g	(5, 5)	2.24	3.00	Cluster1
p08	h	(7, 3)	4.12	2.24	Cluster2
p09	i	(7, 5)	4.12	1.00	Cluster2
p10	j	(8, 5)	5.10	0.00	Cluster2

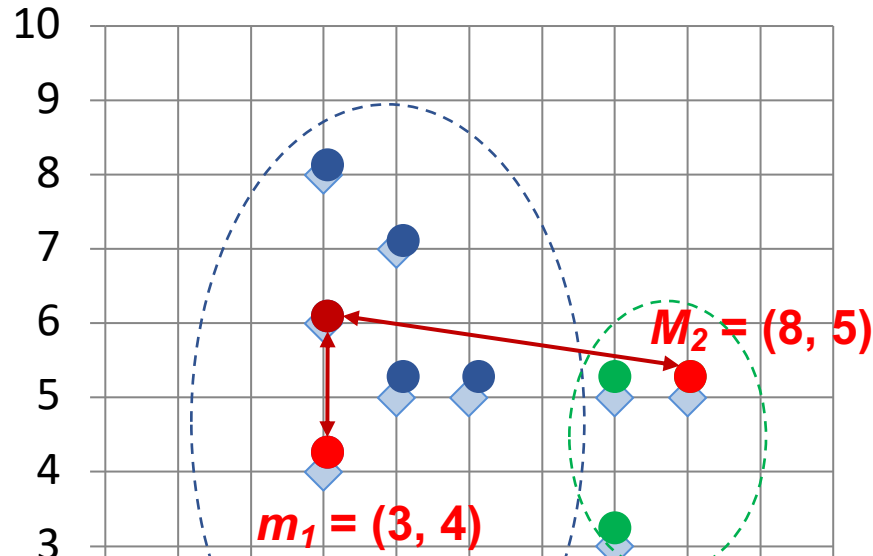
Initial m1 (3, 4)

Initial m2 (8, 5)

***K-Means* Clustering**

Step 2: Compute seed points as the centroids of the clusters of the current partition

Step 3: Assign each objects to most similar center



Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	0.00	5.10	Cluster1
p02	b	(3, 6)	2.00	5.10	Cluster1
p03	c	(3, 8)	4.00	5.83	Cluster1
p04	d	(4, 5)	1.41	4.00	Cluster1

Euclidean distance
 $b(3,6) \leftrightarrow m1(3,4)$
 $= ((3-3)^2 + (4-6)^2)^{1/2}$
 $= (0^2 + (-2)^2)^{1/2}$
 $= (0 + 4)^{1/2}$
 $= (4)^{1/2}$
 $= 2.00$

Euclidean distance
 $b(3,6) \leftrightarrow m2(8,5)$
 $= ((8-3)^2 + (5-6)^2)^{1/2}$
 $= (5^2 + (-1)^2)^{1/2}$
 $= (25 + 1)^{1/2}$
 $= (26)^{1/2}$
 $= 5.10$

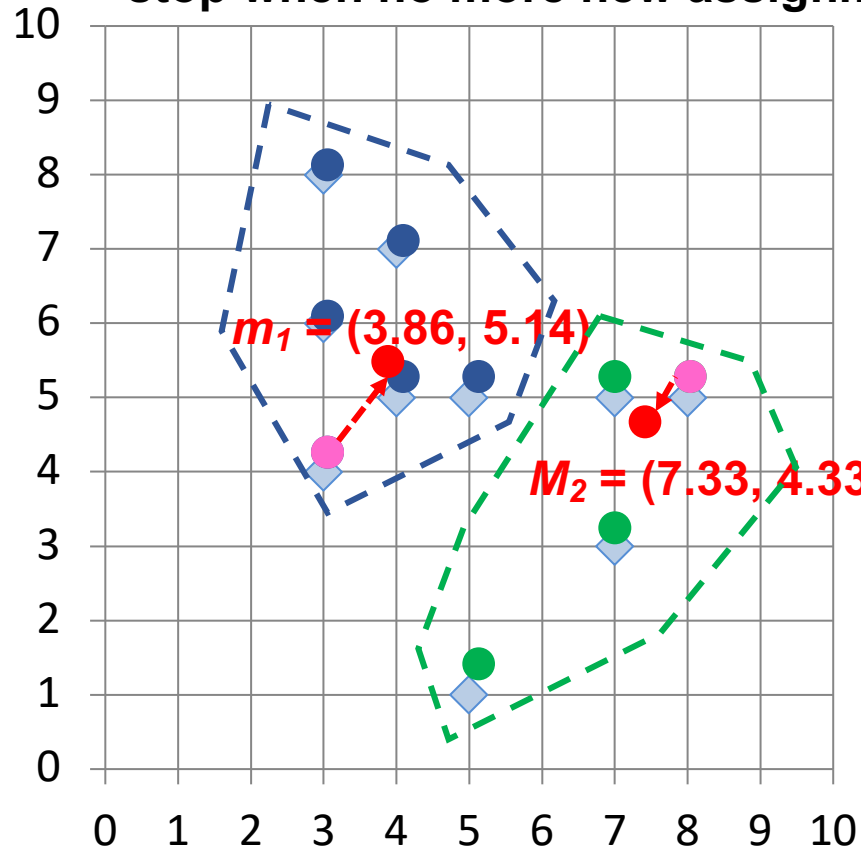
Initial m1 (3, 4)

Initial m2 (8, 5)

K-1

Cluster1
Cluster1
Cluster1
Cluster1
Cluster1
Cluster2
Cluster2
Cluster2

**Step 4: Update the cluster means,
Repeat Step 2, 3,
stop when no more new assignment**



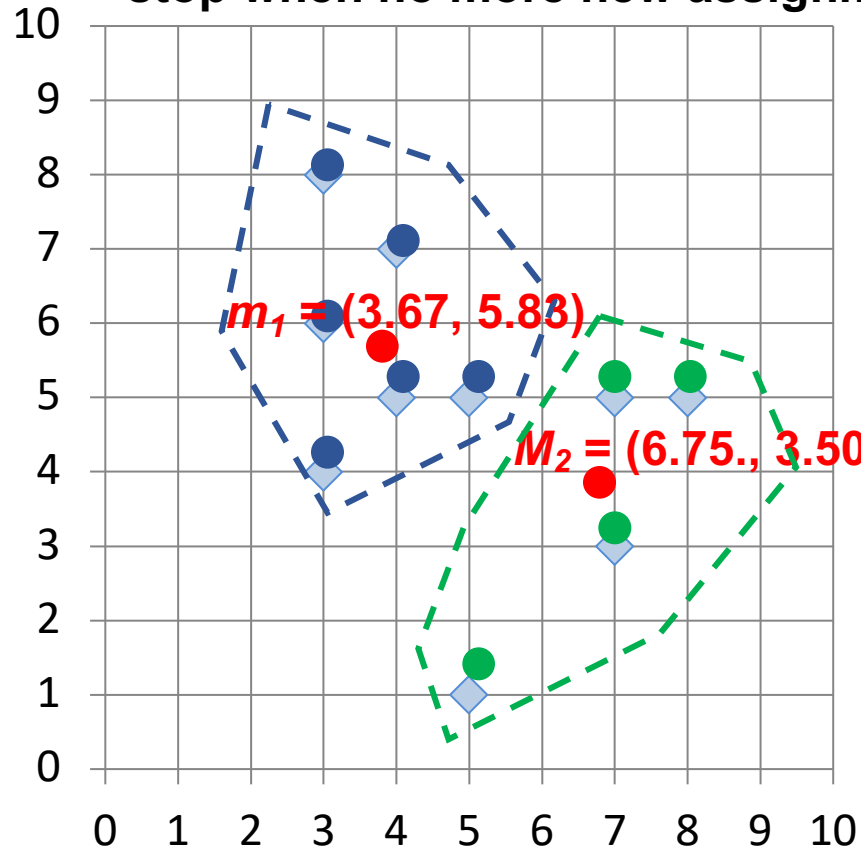
Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.43	4.34	Cluster1
p02	b	(3, 6)	1.22	4.64	Cluster1
p03	c	(3, 8)	2.99	5.68	Cluster1
p04	d	(4, 5)	0.20	3.40	Cluster1
p05	e	(4, 7)	1.87	4.27	Cluster1
p06	f	(5, 1)	4.29	4.06	Cluster2
p07	g	(5, 5)	1.15	2.42	Cluster1
p08	h	(7, 3)	3.80	1.37	Cluster2
p09	i	(7, 5)	3.14	0.75	Cluster2
p10	j	(8, 5)	4.14	0.95	Cluster2

***K-Means* Clustering**

m1 (3.86, 5.14)

m2 (7.33, 4.33)

**Step 4: Update the cluster means,
Repeat Step 2, 3,
stop when no more new assignment**



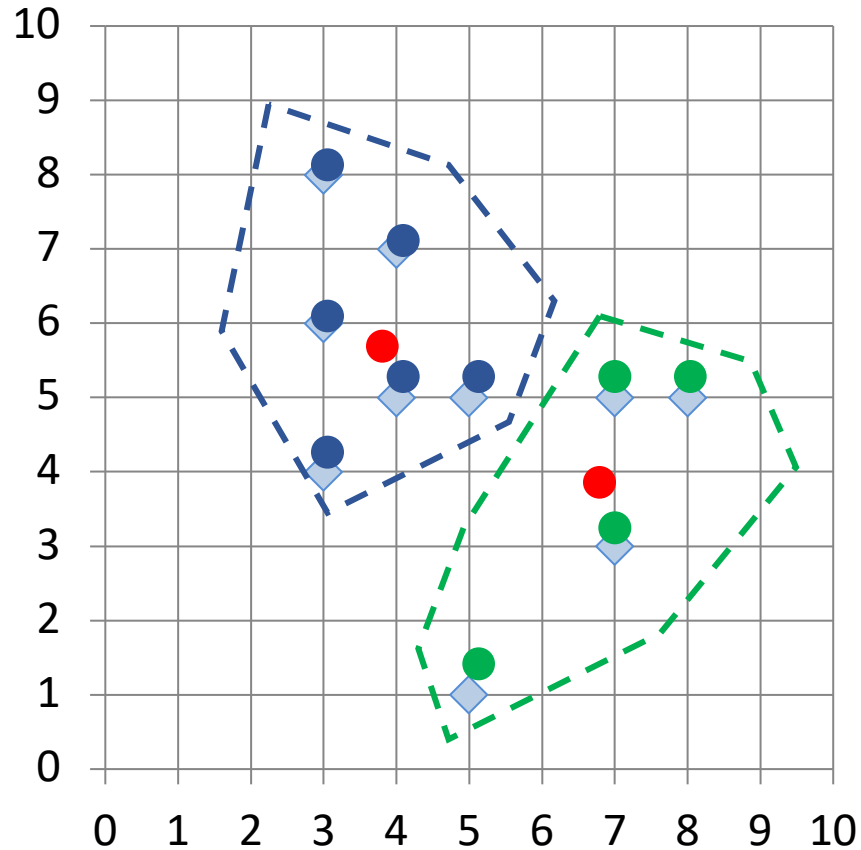
Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

K-Means Clustering

m_1 (3.67, 5.83)

m_2 (6.75, 3.50)

stop when no more new assignment



Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

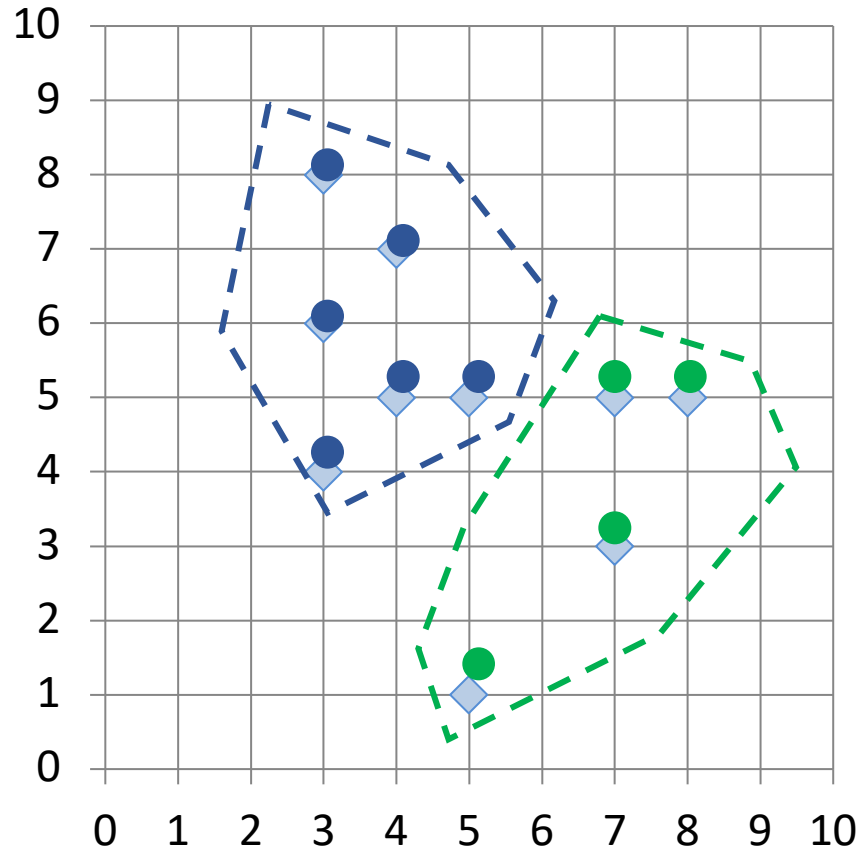
m1 (3.67, 5.83)

m2 (6.75, 3.50)

***K-Means* Clustering**

K-Means Clustering ($K=2$, two clusters)

stop when no more new assignment



Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

K-Means Clustering

m1 (3.67, 5.83)

m2 (6.75, 3.50)

K-Means Clustering

Point	P	P(x,y)	m1 distance	m2 distance	Cluster
p01	a	(3, 4)	1.95	3.78	Cluster1
p02	b	(3, 6)	0.69	4.51	Cluster1
p03	c	(3, 8)	2.27	5.86	Cluster1
p04	d	(4, 5)	0.89	3.13	Cluster1
p05	e	(4, 7)	1.22	4.45	Cluster1
p06	f	(5, 1)	5.01	3.05	Cluster2
p07	g	(5, 5)	1.57	2.30	Cluster1
p08	h	(7, 3)	4.37	0.56	Cluster2
p09	i	(7, 5)	3.43	1.52	Cluster2
p10	j	(8, 5)	4.41	1.95	Cluster2

m1 (3.67, 5.83)

m2 (6.75, 3.50)

gensim

Fork me on GitHub



gensim

topic modelling for humans

[Download](#)
latest version from the Python Package Index

[Direct install with:
easy_install -U gensim](#)

[Home](#) [Tutorials](#) [Install](#) [Support](#) [API](#) [About](#)

```
>>> from gensim import corpora, models, similarities
>>>
>>> # Load corpus iterator from a Matrix Market file on disk.
>>> corpus = corpora.MmCorpus('/path/to/corpus.mm')
>>>
>>> # Initialize Latent Semantic Indexing with 200 dimensions.
>>> lsi = models.LsiModel(corpus, num_topics=200)
>>>
>>> # Convert another corpus to the latent space and index it.
>>> index = similarities.MatrixSimilarity(lsi[another_corpus])
>>>
>>> # Compute similarity of a query vs. indexed documents
>>> sims = index[query]
```

Gensim is a FREE Python library

- ✓ Scalable statistical semantics
- ✓ Analyze plain-text documents for semantic structure
- ✓ Retrieve semantically similar documents

spaCy

The image shows a screenshot of the spaCy website's landing page. The background is a vibrant blue with a pattern of white icons representing various AI and data science concepts like neural networks, gears, and documents. At the top left is the 'spaCy' logo, and at the top right are navigation links for 'HOME', 'USAGE', 'API', 'DEMOS', and 'BLOG'. The main heading is 'Industrial-Strength Natural Language Processing in Python'. Below this are three white boxes, each with a bold title and a paragraph of text describing a key feature of the library: speed, ease of use, and deep learning integration.

spaCy

HOME USAGE API DEMOS BLOG

Industrial-Strength Natural Language Processing in Python

Fastest in the world

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research has confirmed that spaCy is the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. I like to think of spaCy as the Ruby on Rails of Natural Language Processing.

Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with [TensorFlow](#), [Keras](#), [Scikit-Learn](#), [Gensim](#) and the rest of Python's awesome AI ecosystem. spaCy helps you connect the statistical models trained by these libraries to the rest of your application.

<https://spacy.io/>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus, along with "Comment", "Share", and "Settings" icons. A "Table of contents" sidebar on the left lists various sections, with "Text Similarity" currently selected. The main content area displays three code cells:

- Cell [1]: `!python -m spacy download en_core_web_sm`
- Cell [2]: `!python -m spacy download en_core_web_lg` followed by `# Restart Runtime`
- Cell [3]: Python code to import spacy, load the model, and process tokens:

```
1 import spacy
2 nlp = spacy.load("en_core_web_lg")
3 tokens = nlp("apple banana cat dog notaword")
4 for token in tokens:
5     print(token.text, token.has_vector, token.vector_norm, token.is_oov)
```

The output of the third cell shows the following results:

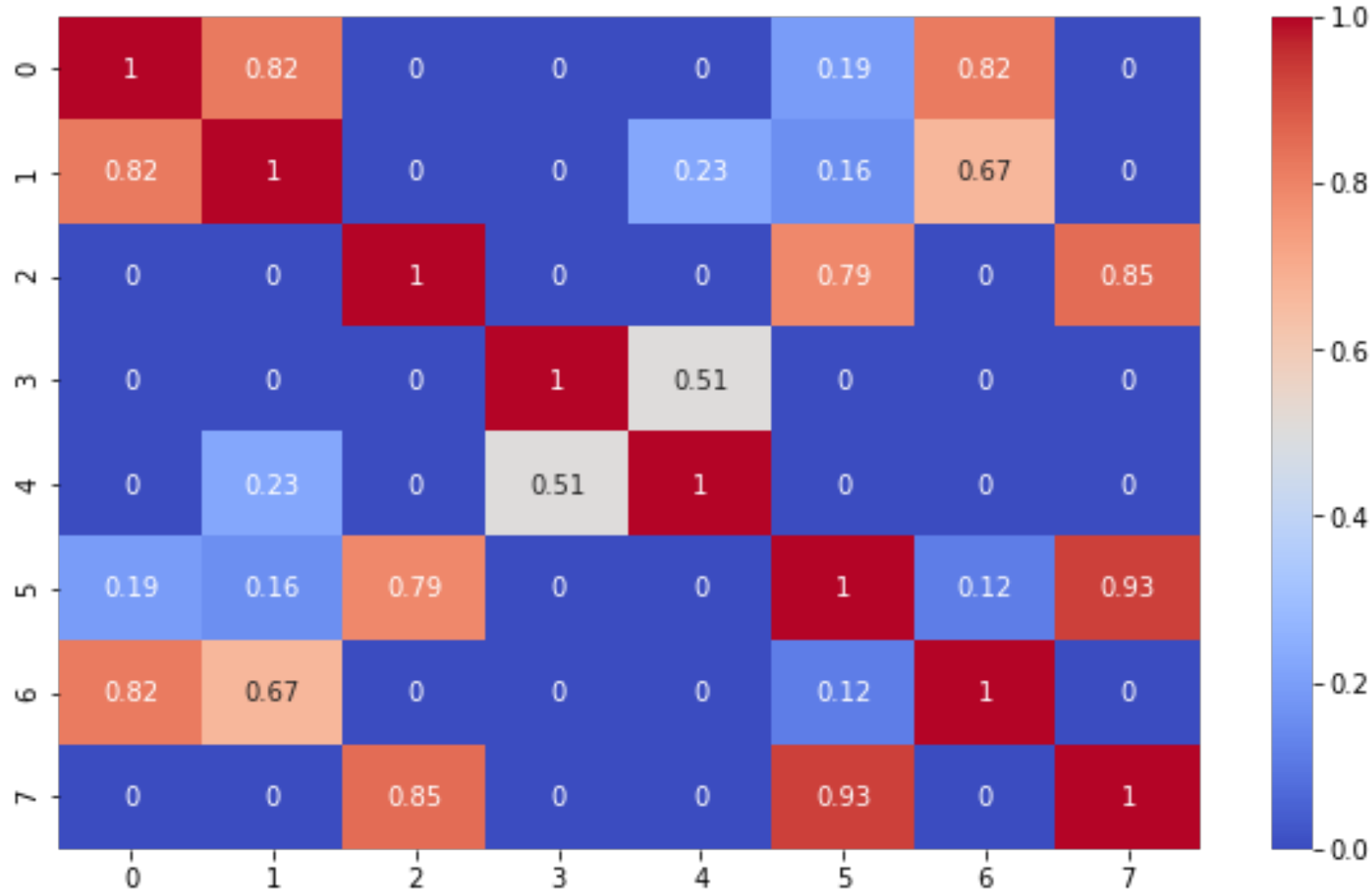
```
apple True 7.1346846 False
banana True 6.700014 False
cat True 6.6808186 False
dog True 7.0336733 False
notaword False 0.0 True
```

Below the code cells, there is a preview of the next cell's code: `import spacy`, `nlp = spacy.load("en_core_web_lg")`, `doc1 = nlp("I like cat.")`, `doc2 = nlp("I like dog.")`, and `doc1.similarity(doc2)`.

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

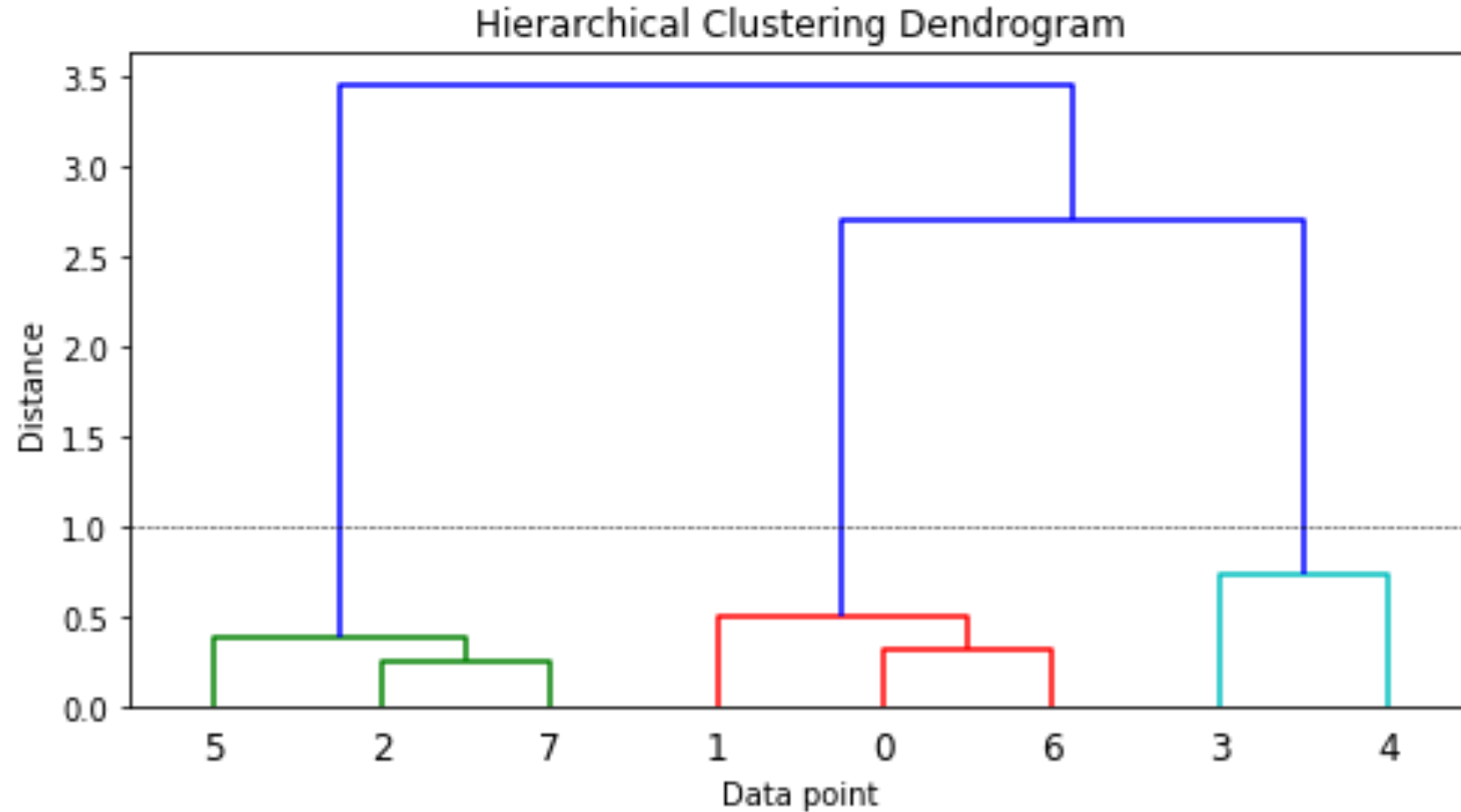
<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>



<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>



<https://tinyurl.com/aintpupython101>

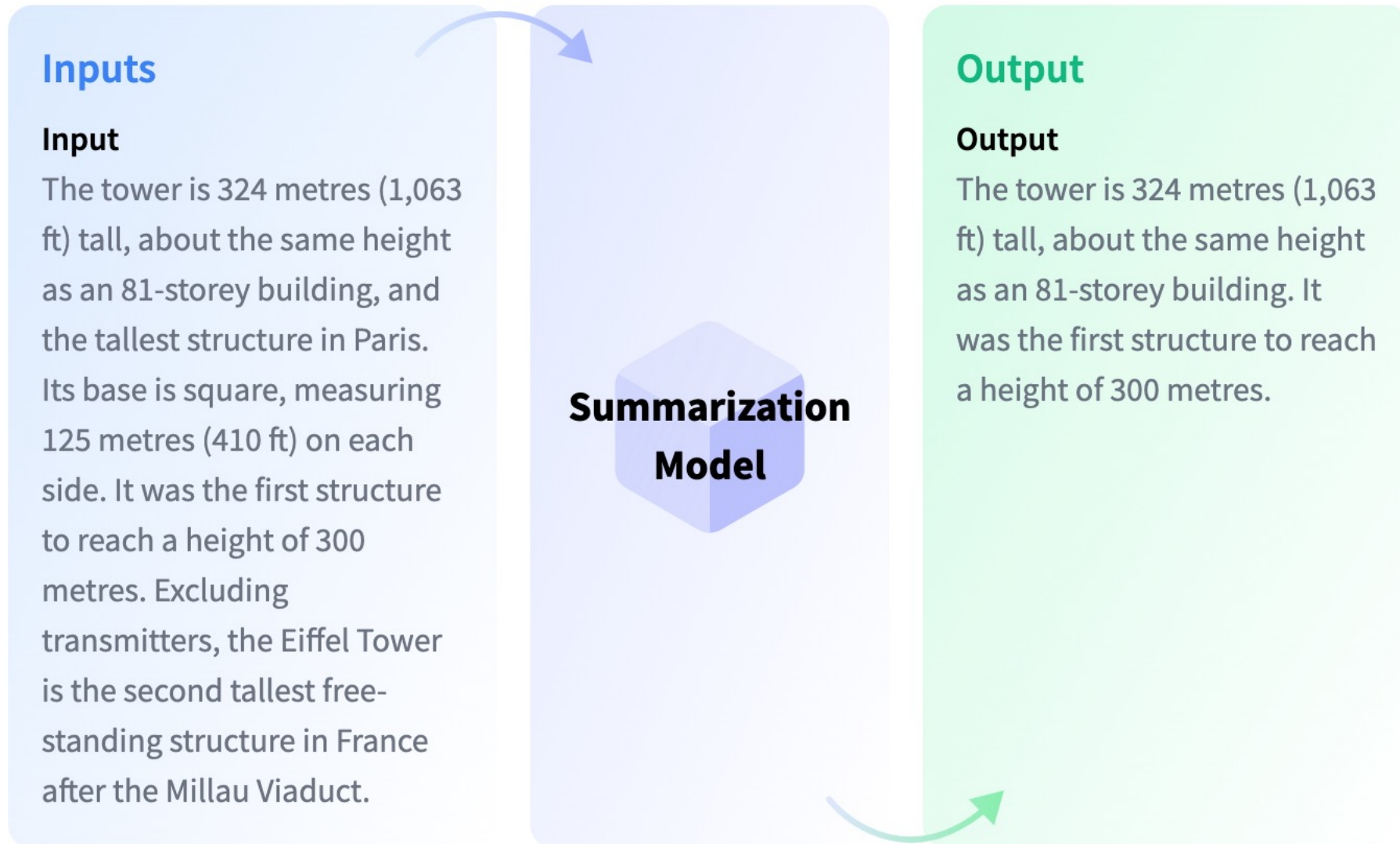
Text Summarization and Topic Models

Outline

- **Text Summarization**
 - **Extractive Text Summarization**
 - **Abstractive Text Summarization**
 - **PEGASUS: Abstractive Summarization**
- **Topic Models**
 - **Topic Modeling**
 - **Latent Dirichlet Allocation (LDA)**
 - **BERTopic**

Text Summarization

Text Summarization



Text Summarization

Summarization

Examples ▾

The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building, and the tallest structure in Paris. Its base is square, measuring 125 metres (410 ft) on each side. During its construction, the Eiffel Tower surpassed the Washington Monument to become the tallest man-made structure in the world, a title it held for 41 years until the Chrysler Building in New York City was finished in 1930. It was the first structure to reach a height of 300 metres. Due to the addition of a broadcasting aerial at the top of the tower in 1957, it is now taller than the Chrysler Building by 5.2 metres (17 ft). Excluding transmitters, the Eiffel Tower is the second tallest free-standing structure in France after the Millau Viaduct.

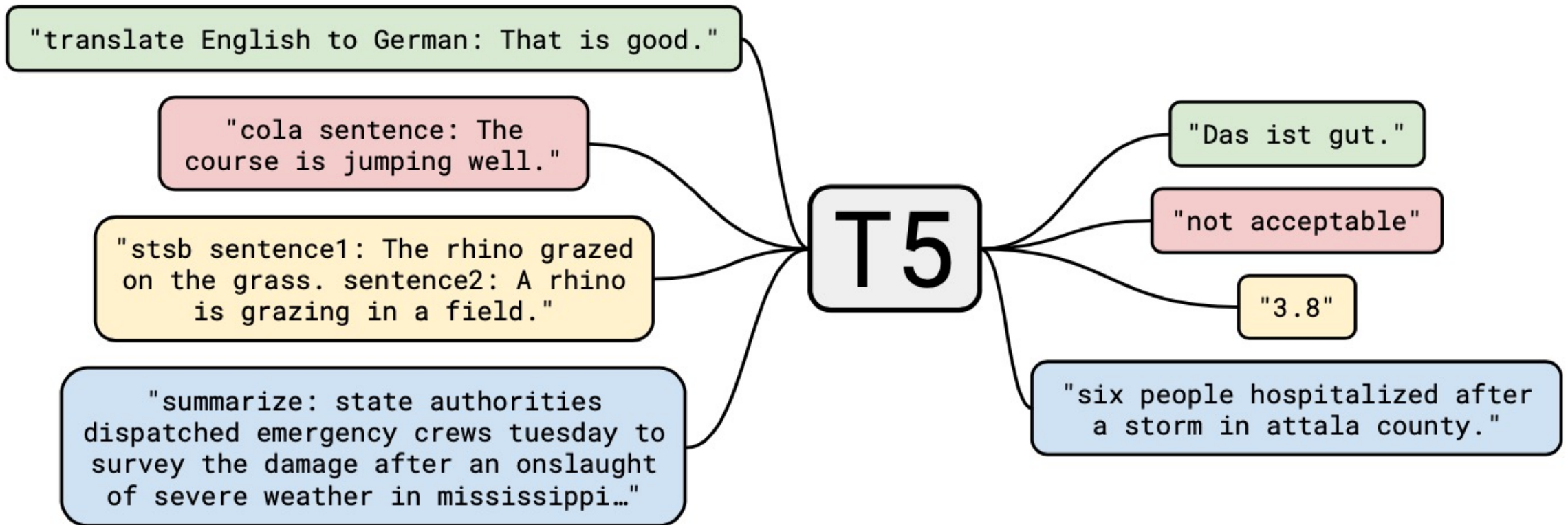
Compute

Computation time on cpu: cached

The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building . It was the first structure to reach a height of 300 metres . It is now taller than the Chrysler Building in New York City by 5.2 metres (17 ft) Excluding transmitters, the Eiffel Tower is the second tallest free-standing structure in France .

T5

Text-to-Text Transfer Transformer



Text Summarization and Information Extraction

- **Key-phrase extraction**
 - **extracting key influential phrases from the documents.**
- **Topic modeling**
 - **Extract various diverse concepts or topics present in the documents, retaining the major themes.**
- **Document summarization**
 - **Summarize entire text documents to provide a gist that retains the important parts of the whole corpus.**

Natural Language Processing (NLP) and Text Mining

Raw text

Sentence Segmentation

Tokenization

Part-of-Speech (POS)

Stop word removal

Stemming / **Lemmatization**

Dependency Parser

String Metrics & Matching

word's stem

am → am

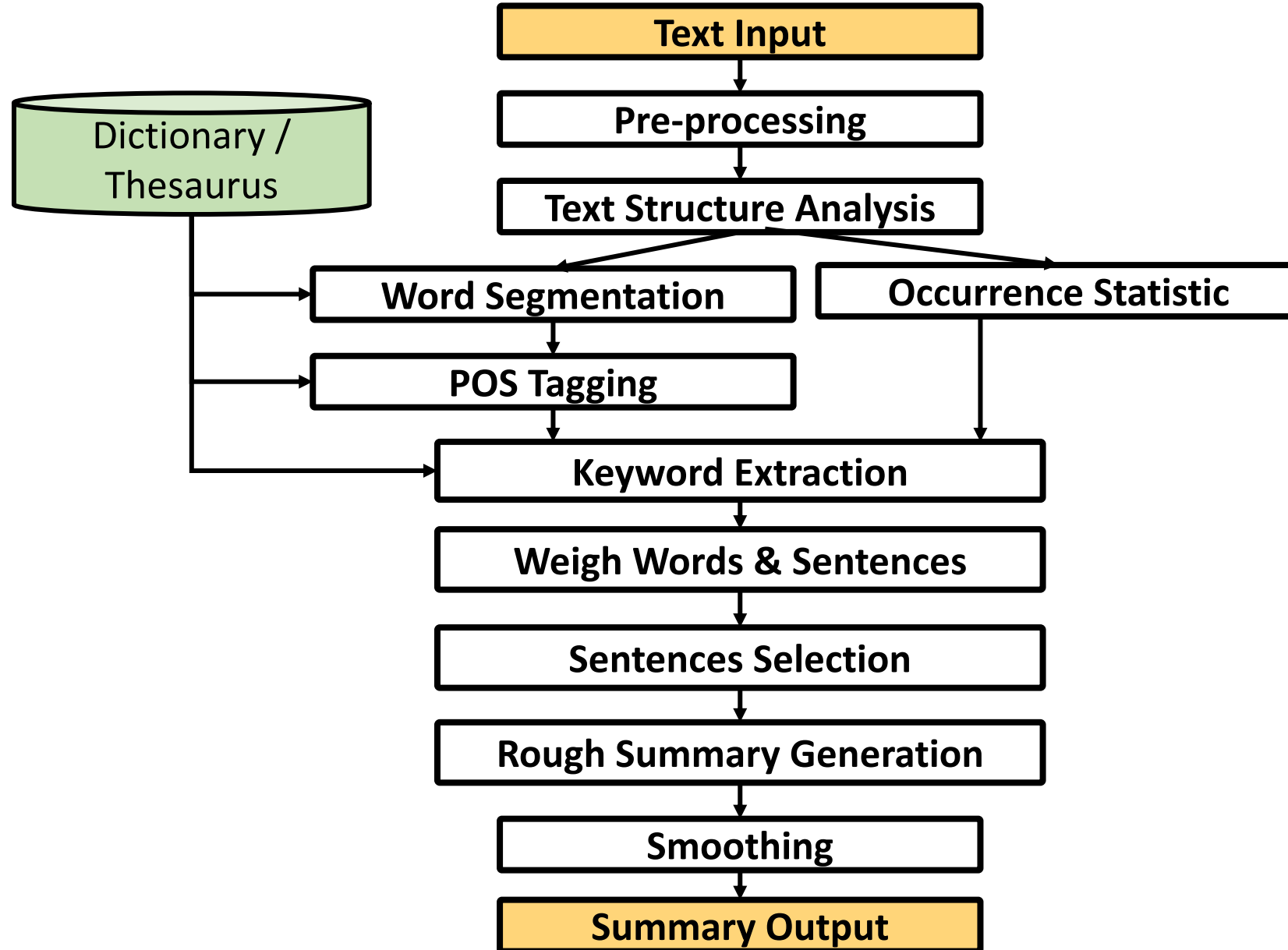
having → hav

word's lemma

am → be

having → have

Text Summarization



Topic Modeling

Topics

gene	0.04
dna	0.02
genetic	0.01
...	

life	0.02
evolve	0.01
organism	0.01
...	

brain	0.04
neuron	0.02
nerve	0.01
...	

data	0.02
number	0.02
computer	0.01
...	

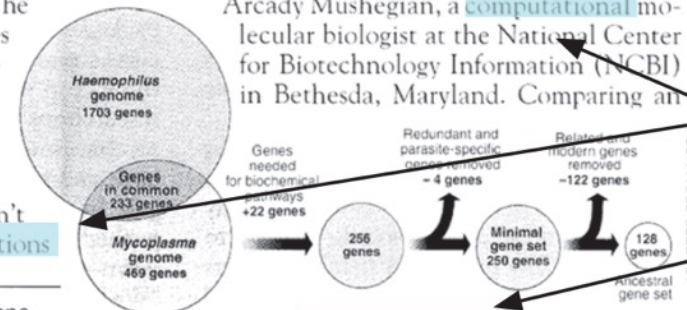
Documents

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions

"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

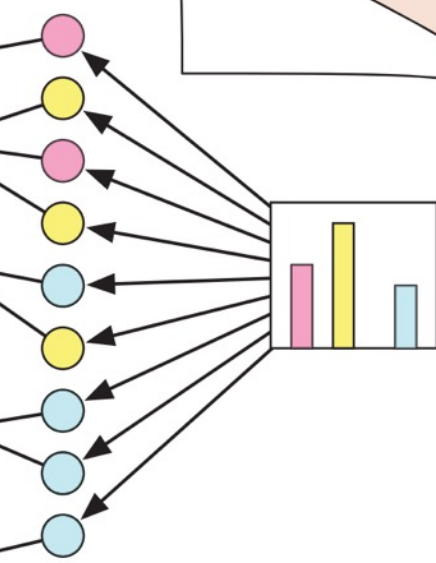


* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

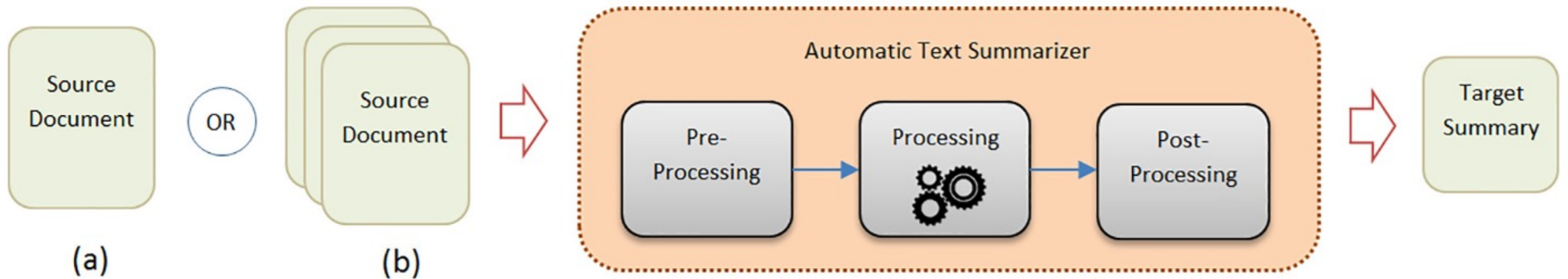
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

Topic proportions and assignments

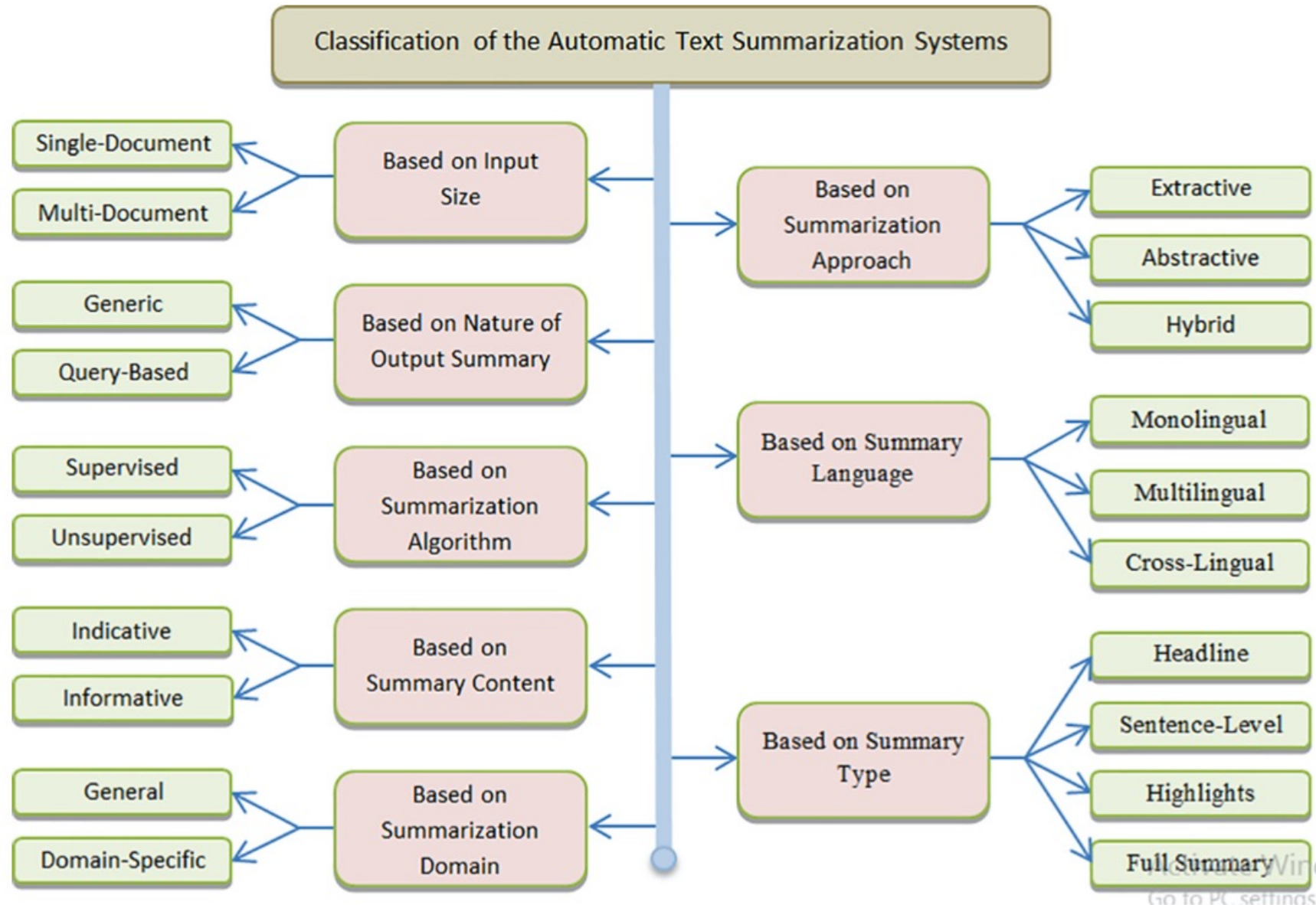


Automatic Text Summarization

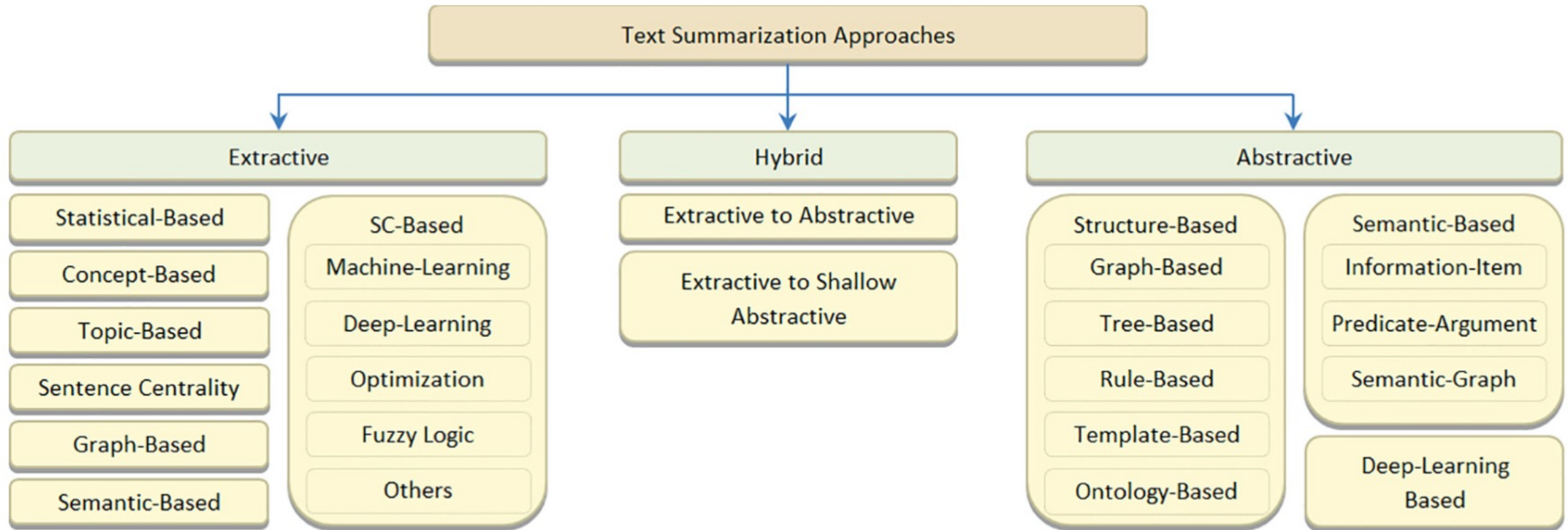


(a) Single-document or (b) Multi-document, automatic text summarizer

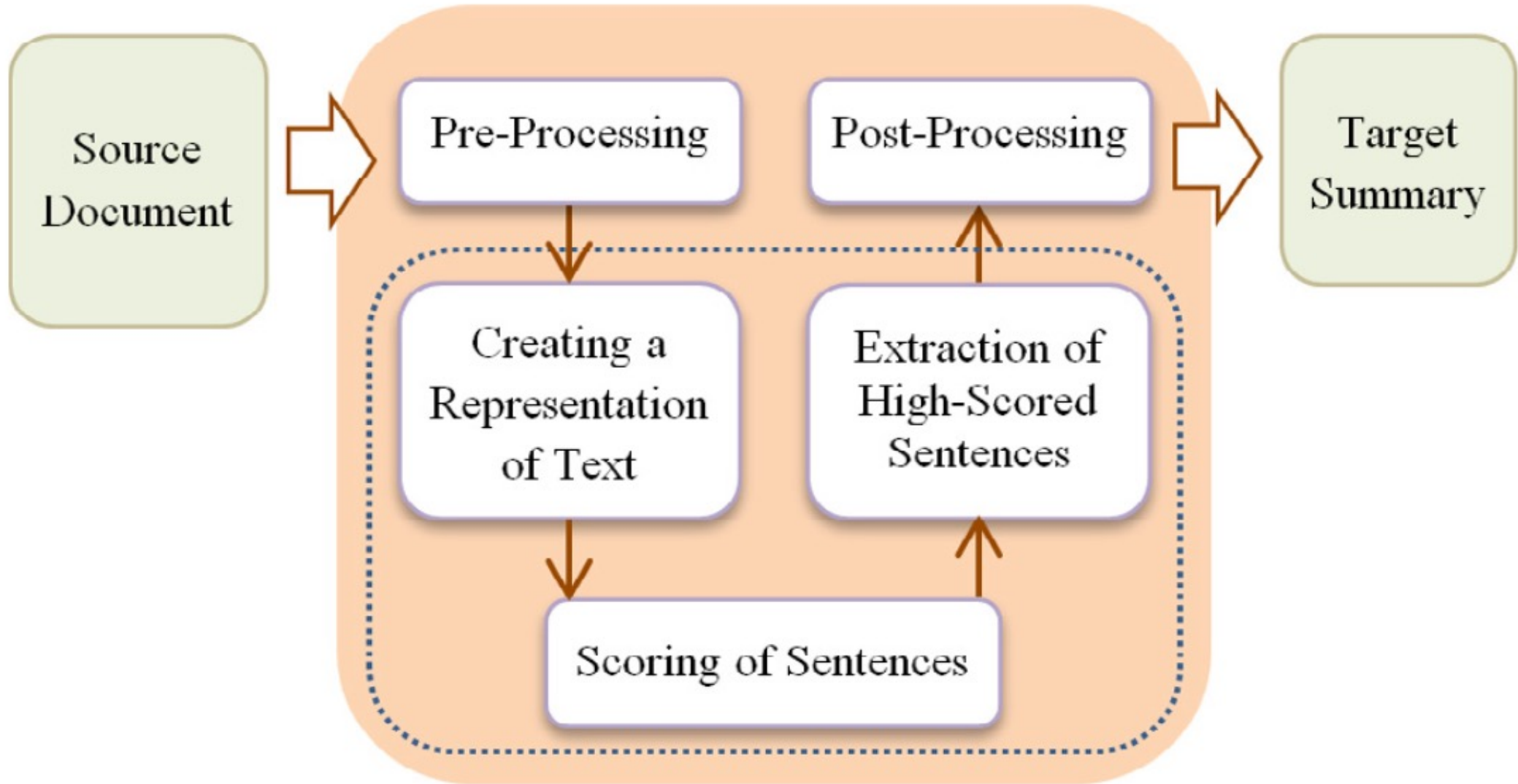
Classification of Automatic Text Summarization Systems



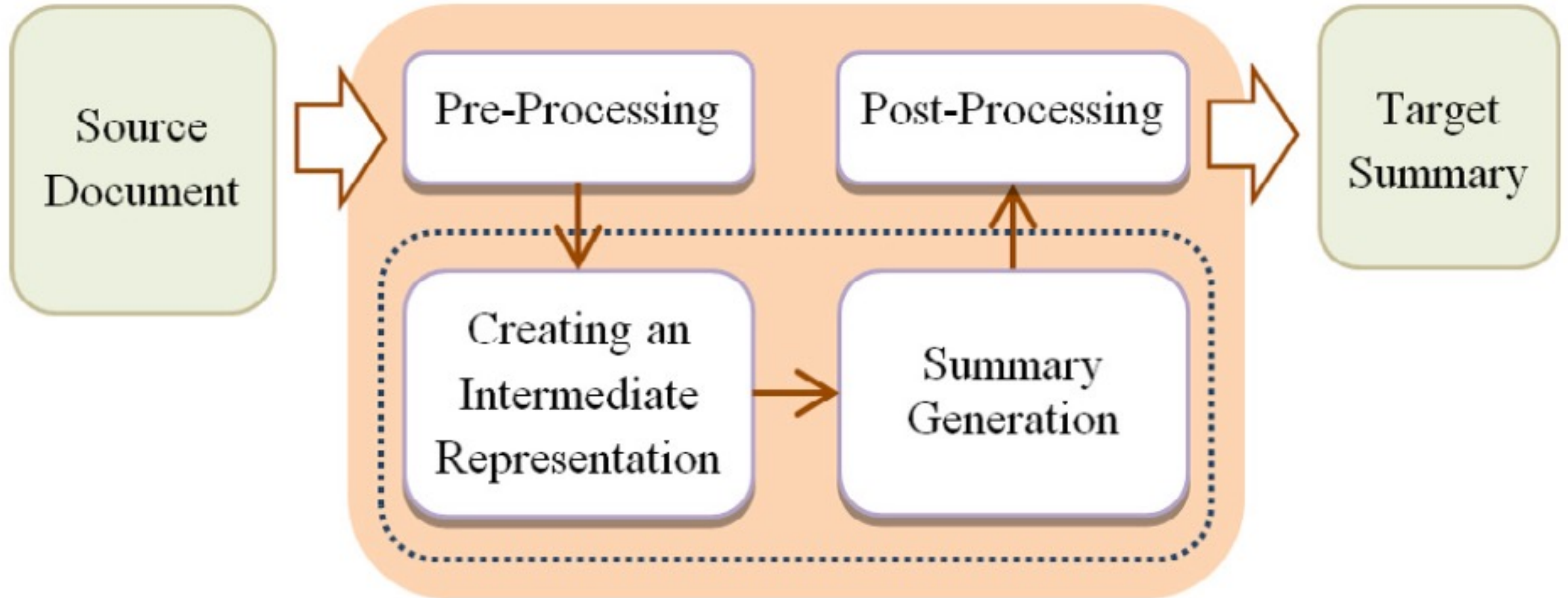
Automatic Text Summarization Approaches



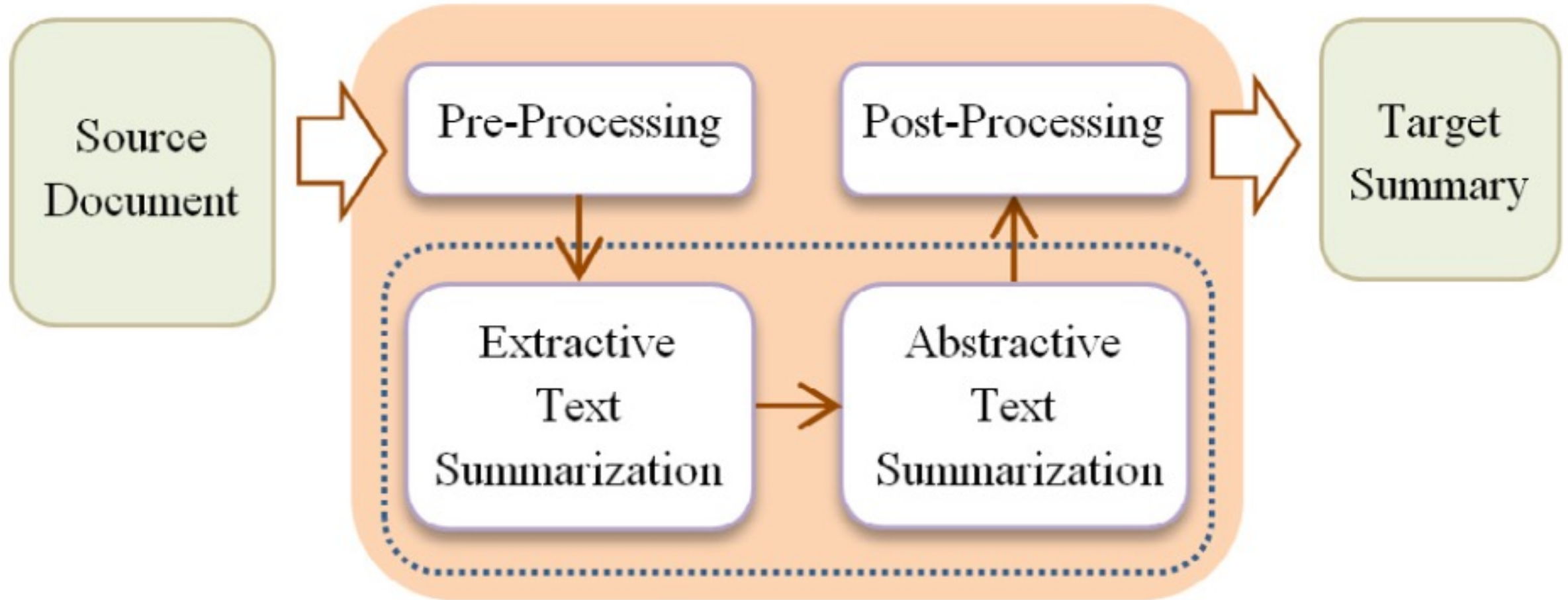
Extractive Text Summarization System



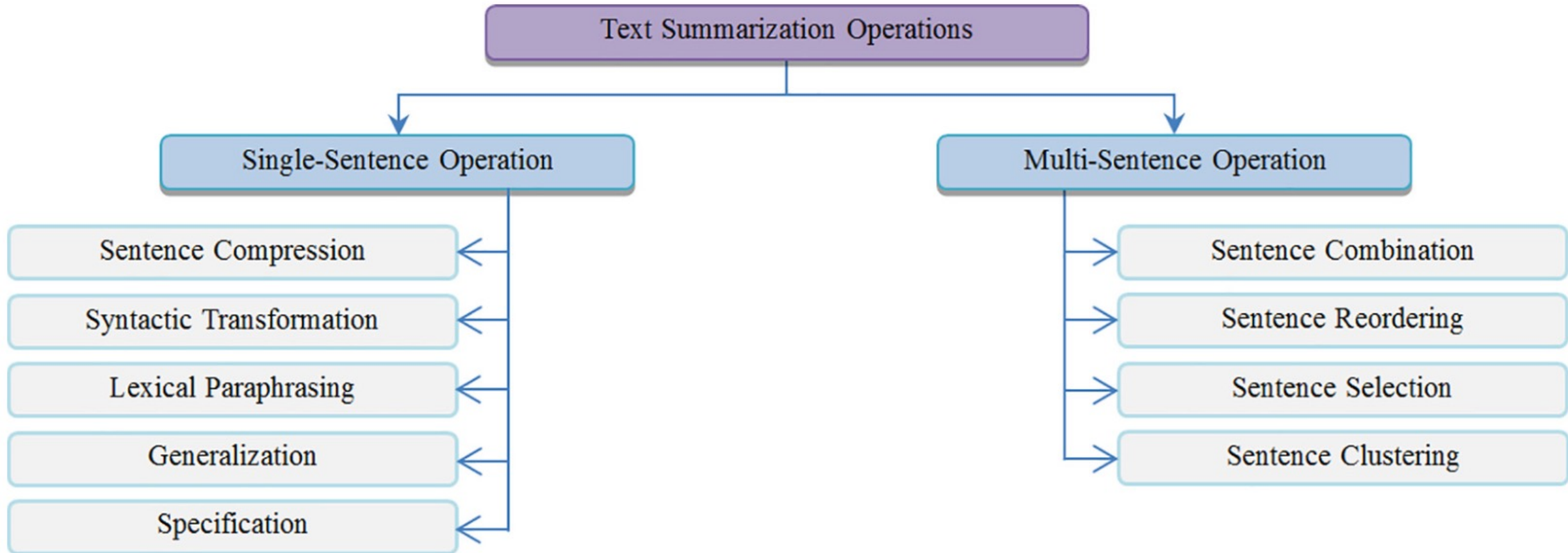
Abstractive Text Summarization System



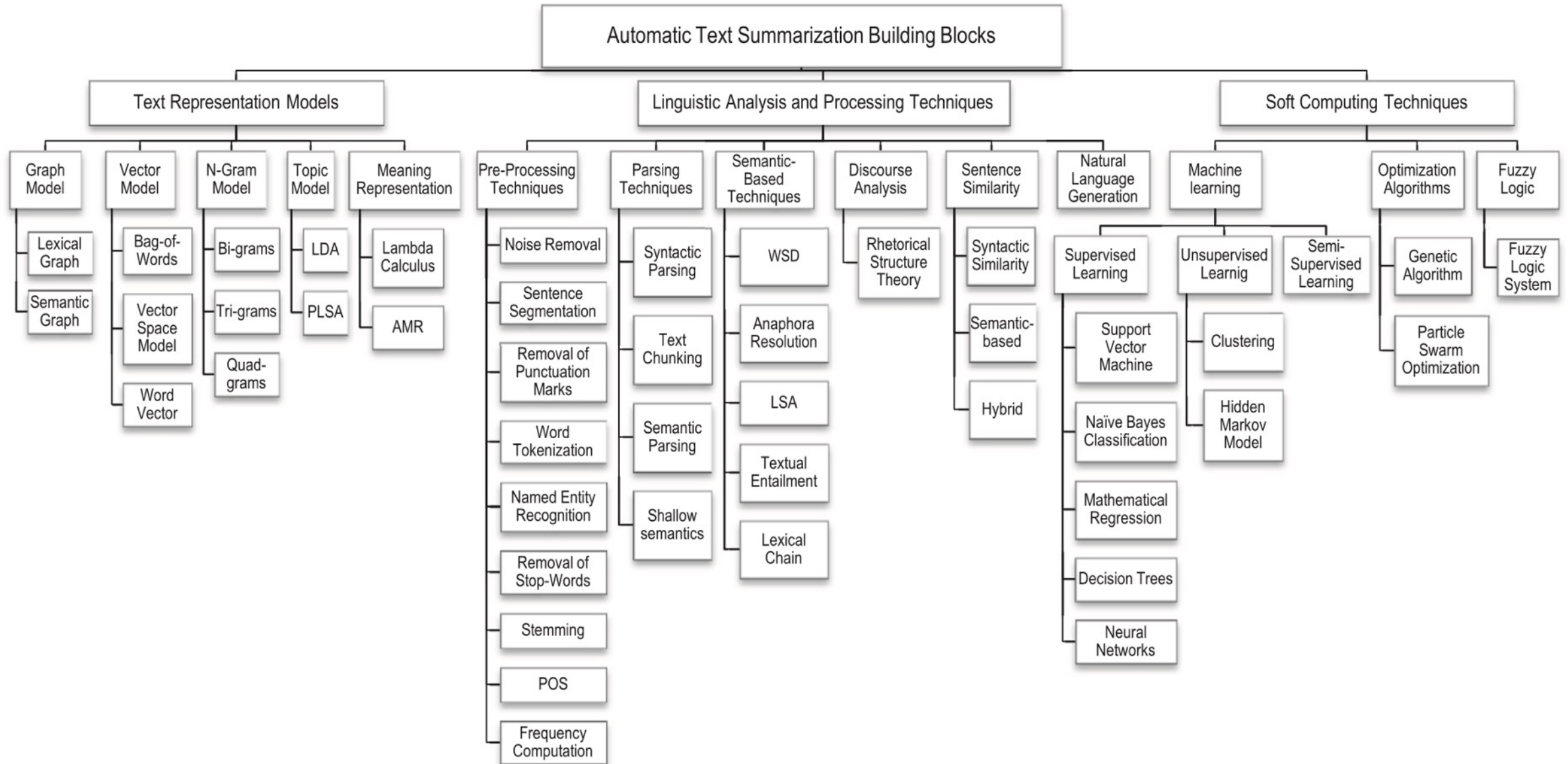
Hybrid Text Summarization System



Single-sentence and Multi-sentence Text Summarization Operations

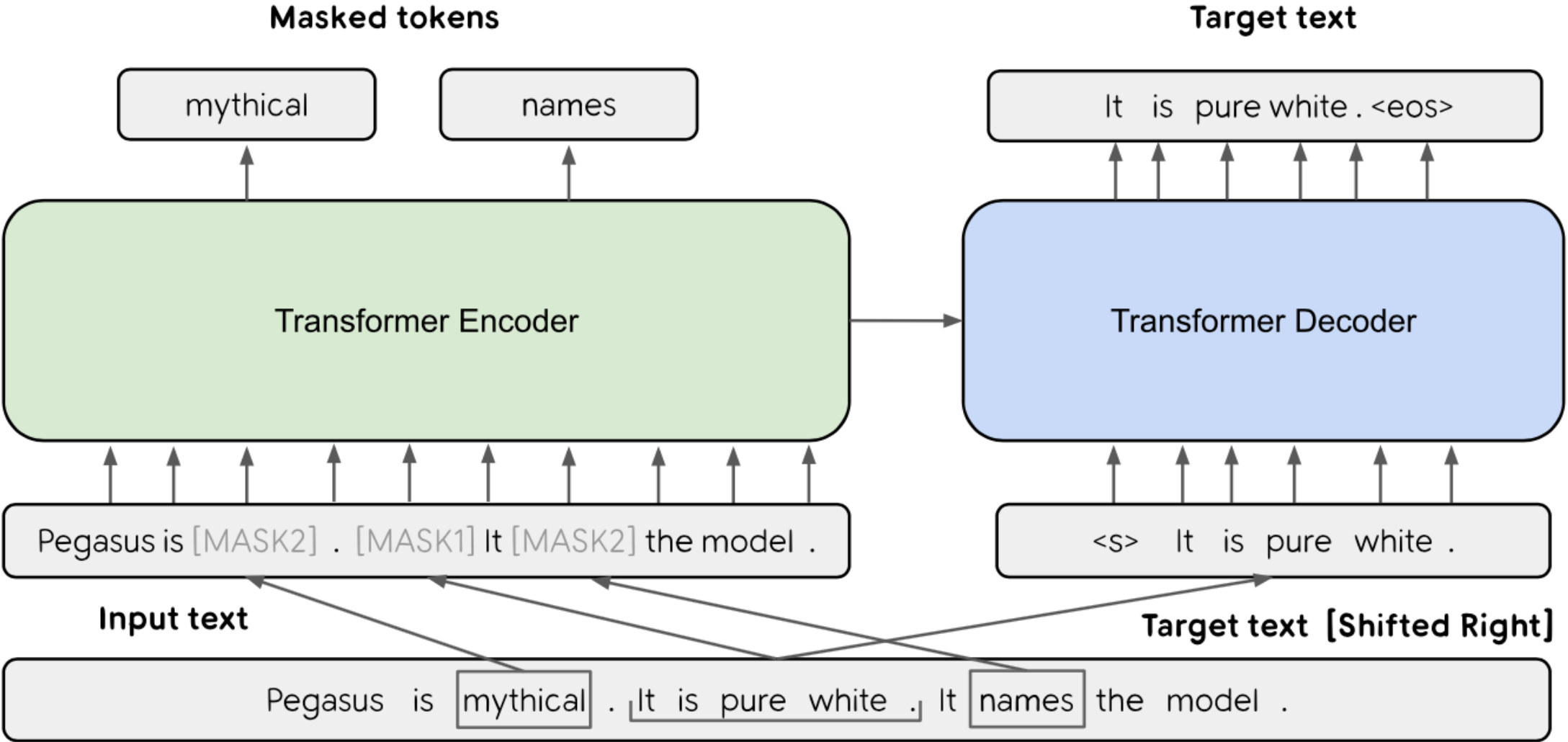


Automatic Text Summarization Building Blocks



PEGASUS:

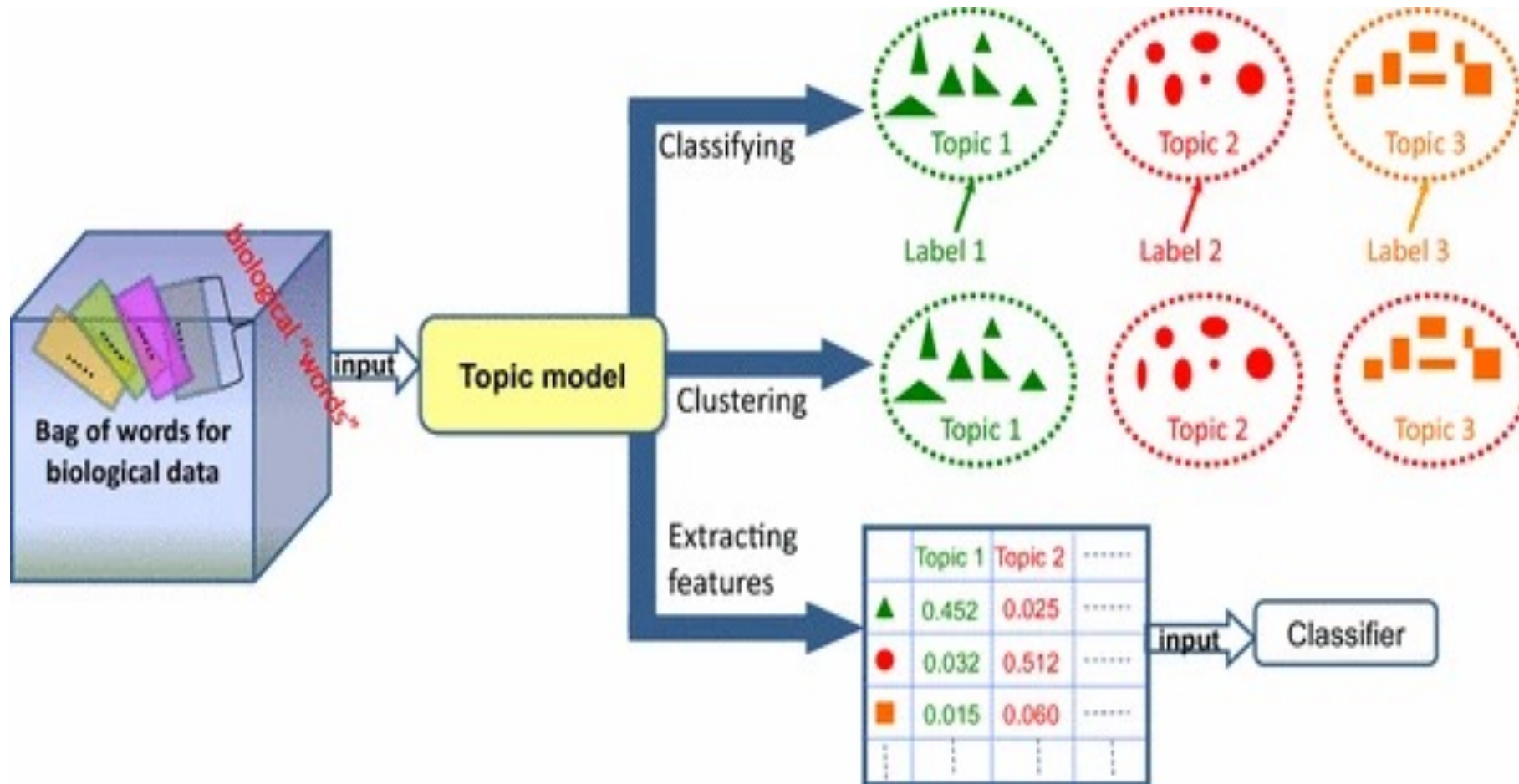
Pre-training with Extracted Gap-sentences for **Abstractive Summarization**



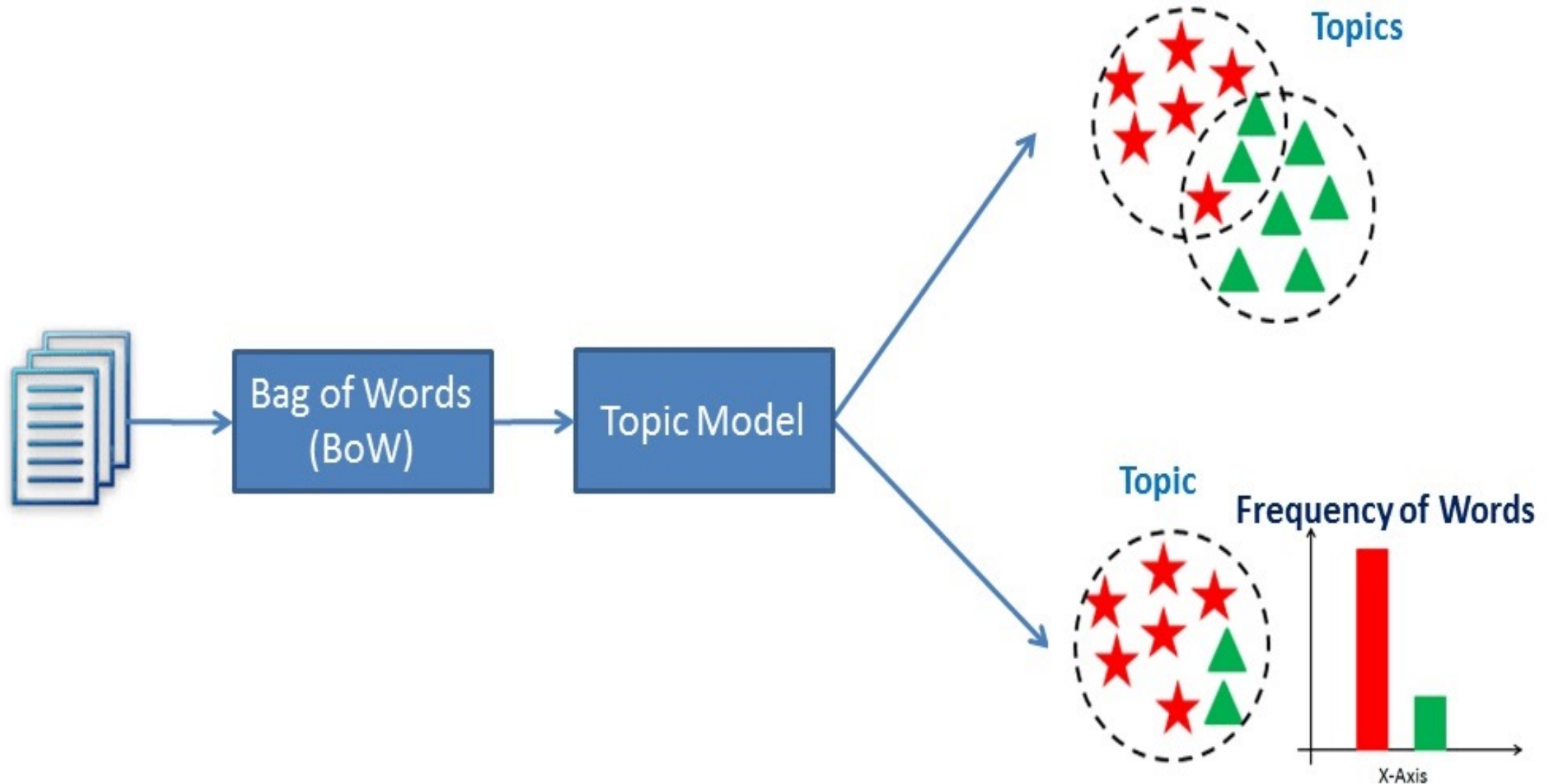
Source: Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu (2020). "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization." In International Conference on Machine Learning, pp. 11328-11339. PMLR, 2020.

Topic Modeling

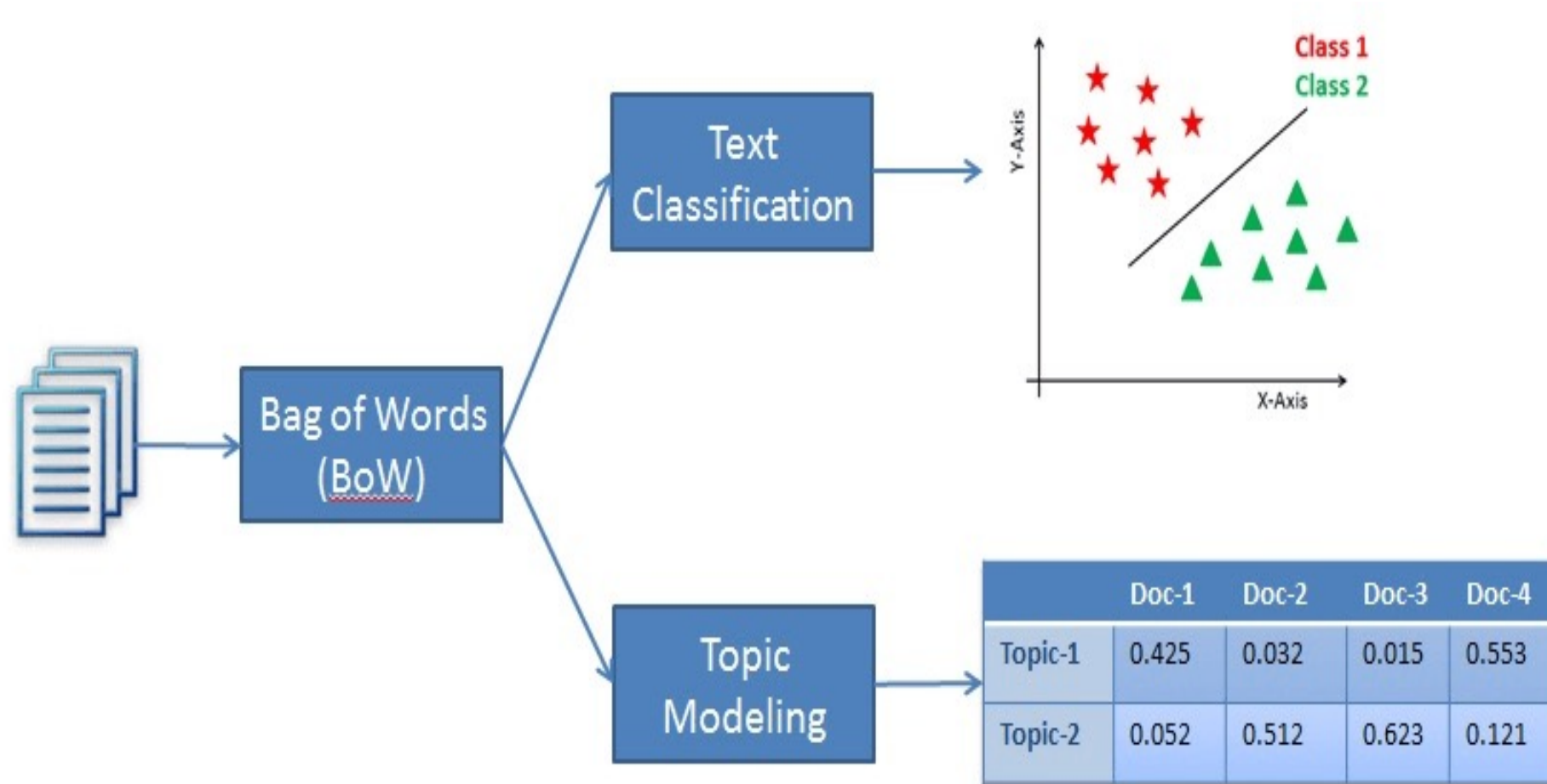
Topic Model in Bioinformatics



Topic Modeling

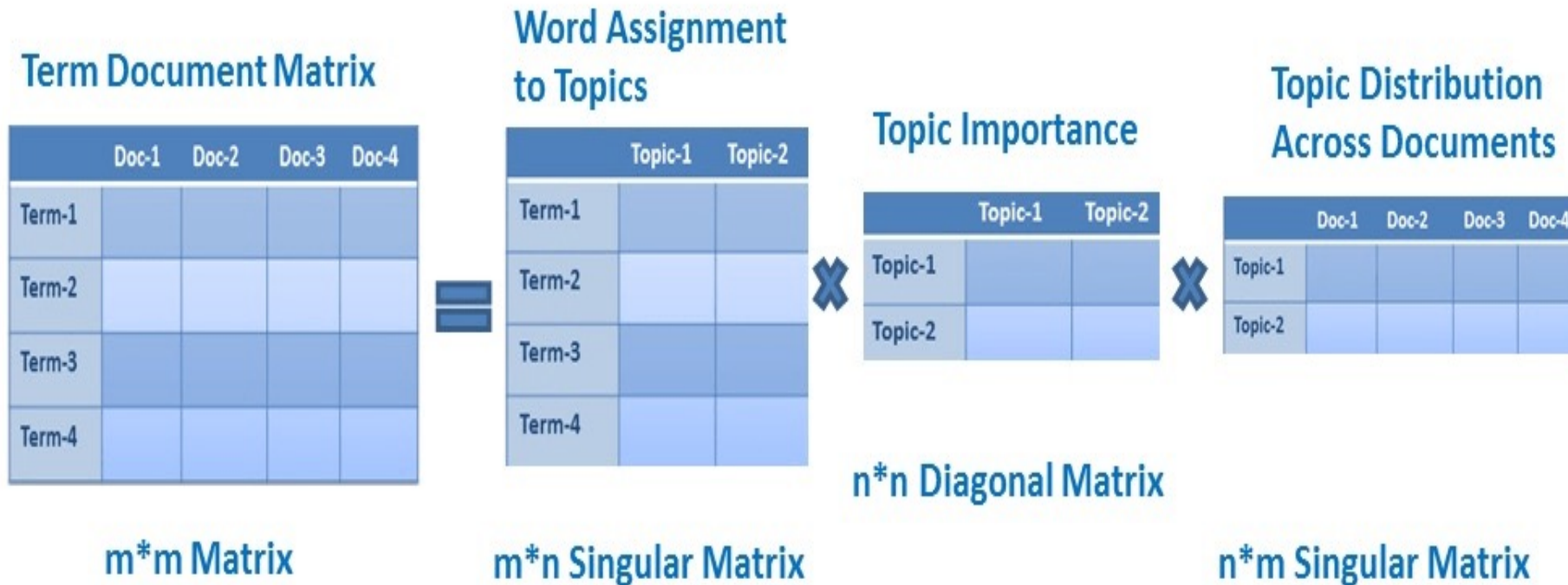


Topic Modeling (Unsupervised Learning) VS. Text Classification (Supervised Learning)



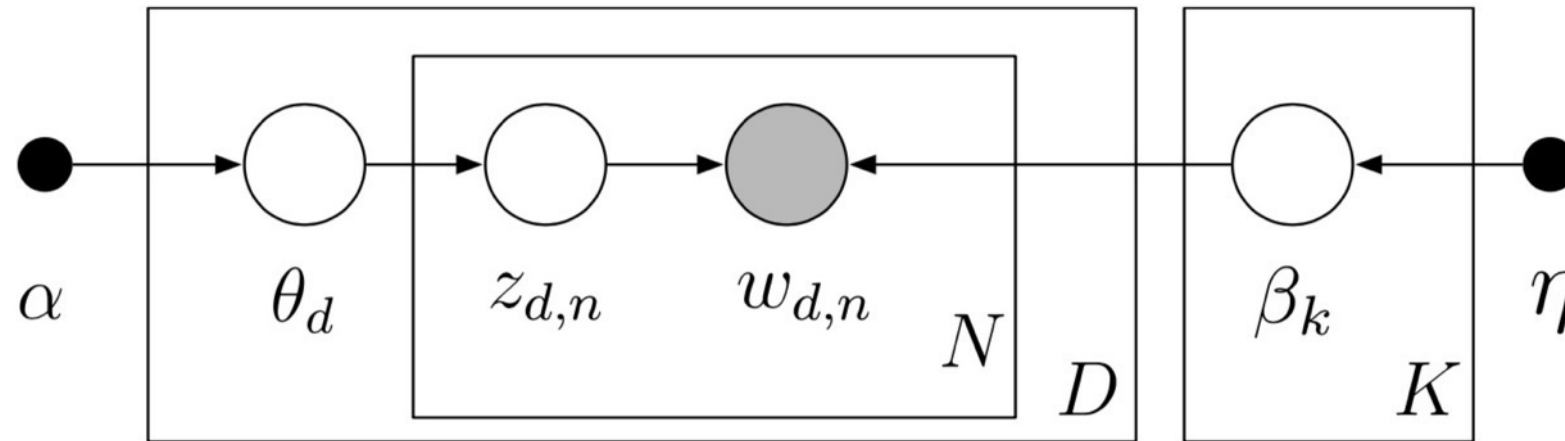
Topic Modeling

Term Document Matrix to Topic Distribution



Topic Modeling

Latent Dirichlet Allocation (LDA)



D documents

N words

K topics

Latent Dirichlet Allocation (Blei et al., 2003)

Latent Dirichlet Allocation

David M. Blei

*Computer Science Division
University of California
Berkeley, CA 94720, USA*

BLEI@CS.BERKELEY.EDU

Andrew Y. Ng

*Computer Science Department
Stanford University
Stanford, CA 94305, USA*

ANG@CS.STANFORD.EDU

Michael I. Jordan

*Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA*

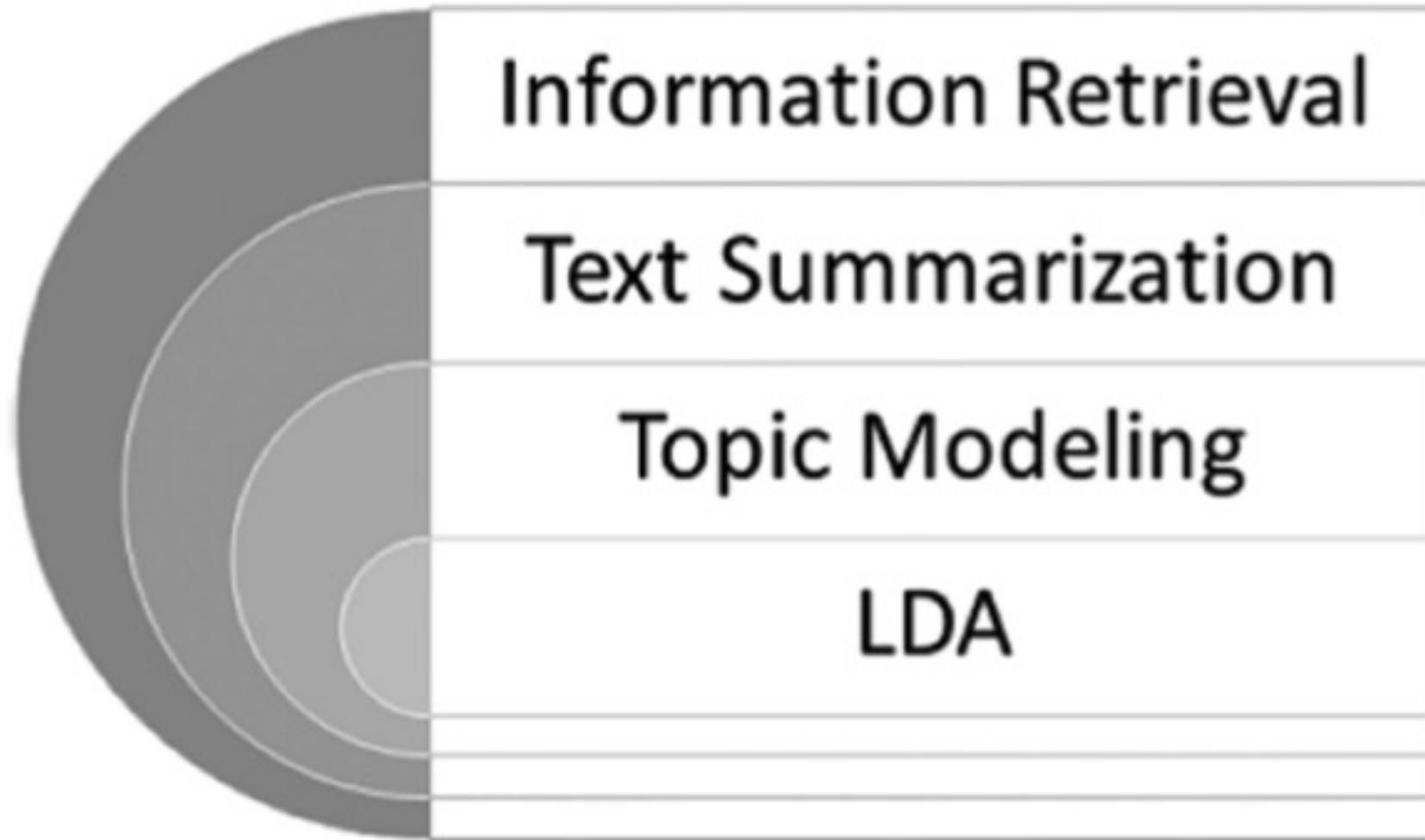
JORDAN@CS.BERKELEY.EDU

Editor: John Lafferty

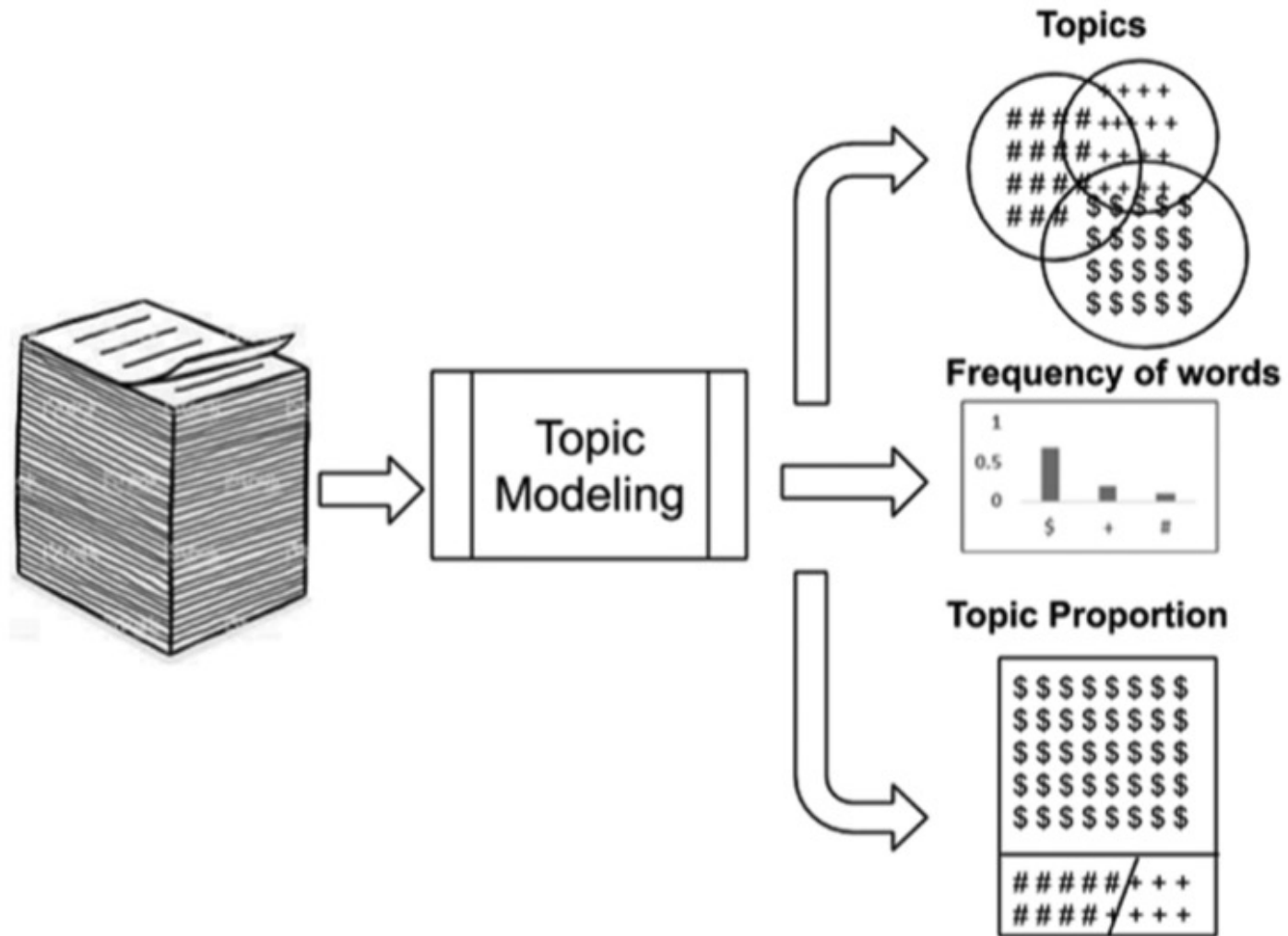
Abstract

We describe *latent Dirichlet allocation* (LDA), a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document. We present efficient approximate inference techniques based on variational methods and an EM algorithm for empirical Bayes parameter estimation. We report results in document modeling, text classification, and collaborative filtering, comparing to a mixture of unigrams model and the probabilistic LSI model.

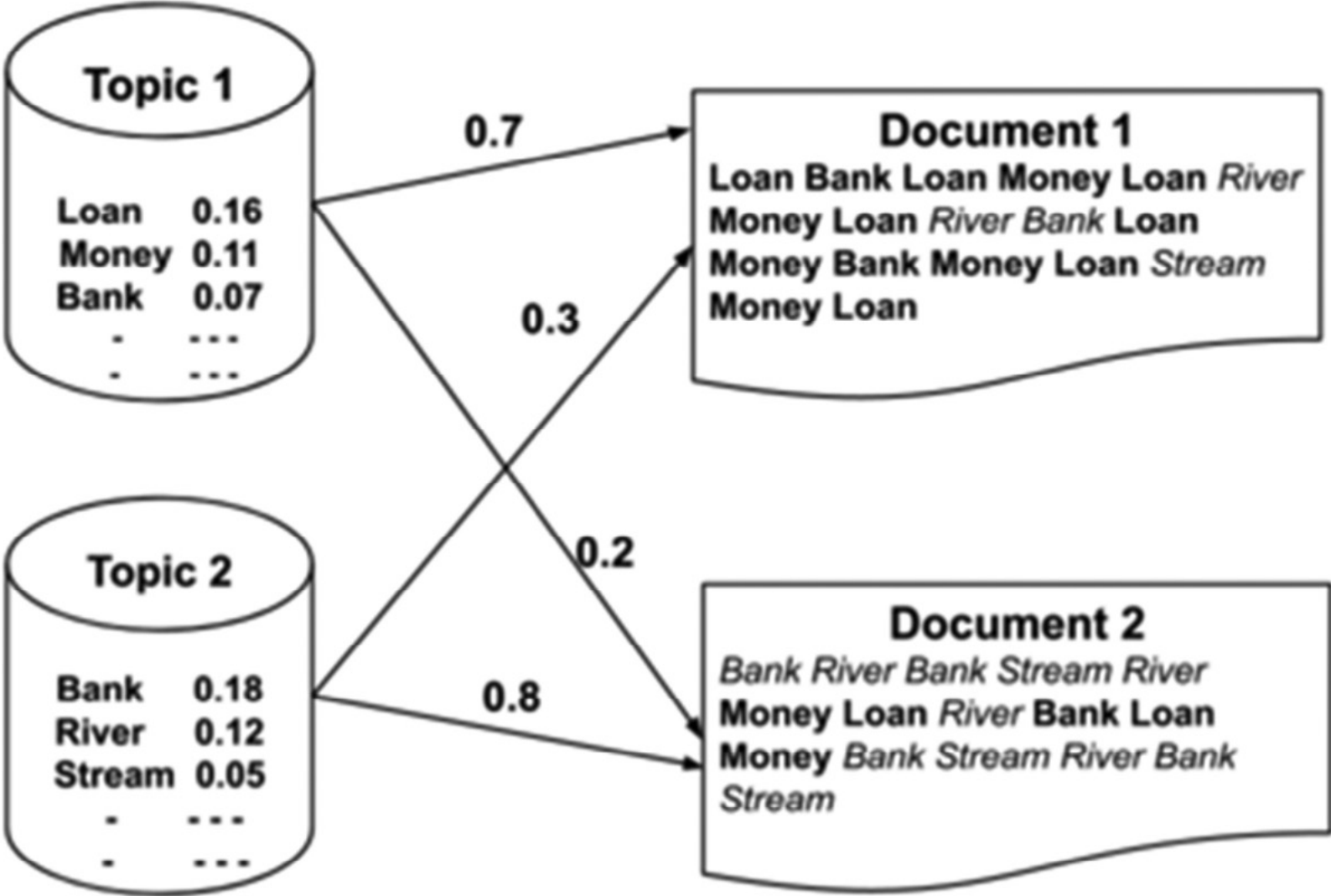
Topic Modeling Using Latent Dirichlet allocation (LDA)



Topic Modeling Technique



The Generative Process of Latent Dirichlet Allocation (LDA)

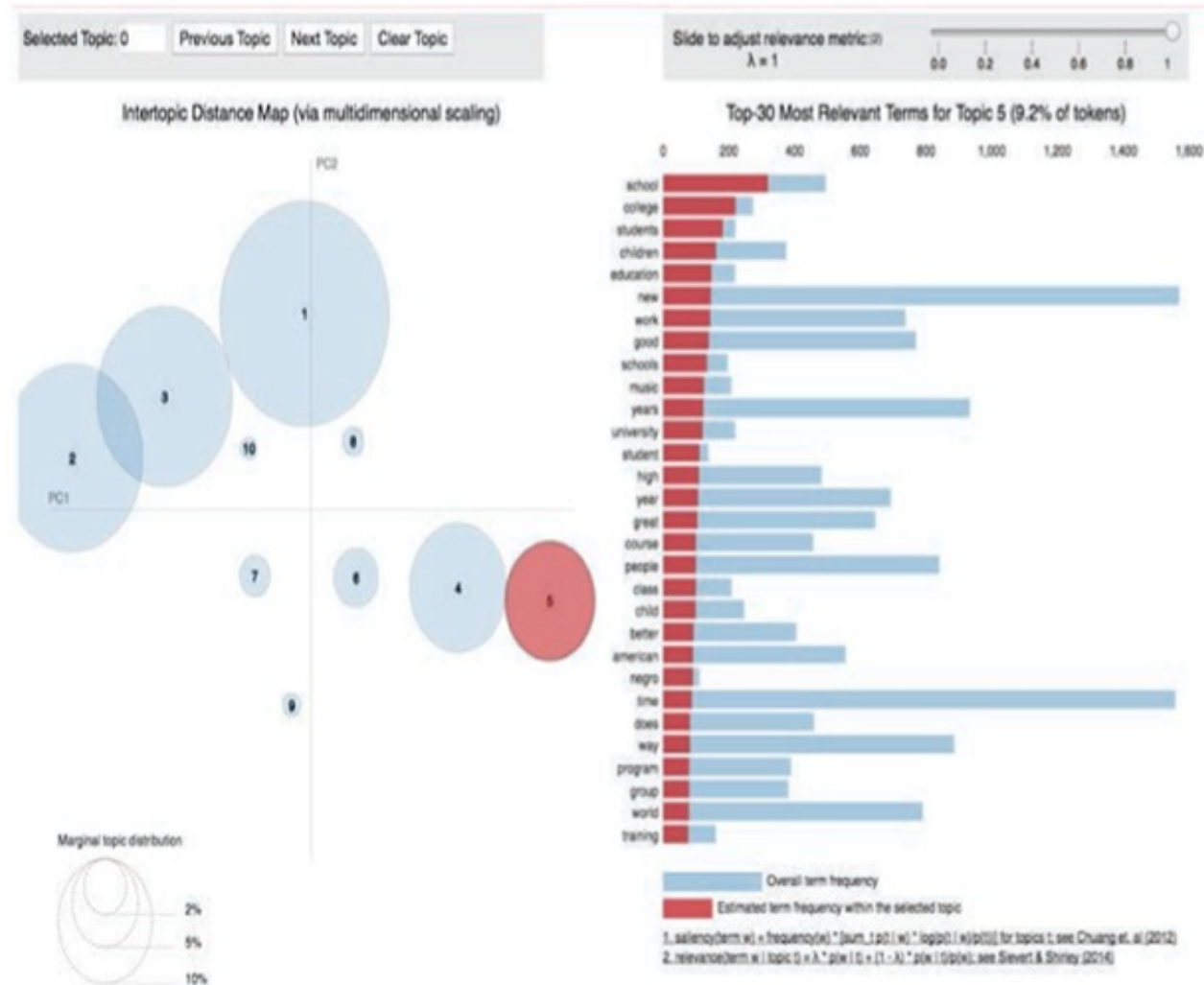


Source: Uttam Chauhan, and Apurva Shah (2021). "Topic modeling using latent Dirichlet allocation: A survey." ACM Computing Surveys (CSUR) 54, no. 7 (2021): 1-35.

Topic Visualization as Word Clouds



LDavis: Gensim Topic Model Visualization



BERTopic

Neural topic modeling with a class-based TF-IDF procedure



Maarten Grootendorst (2022). "BERTopic: Neural topic modeling with a class-based TF-IDF procedure."
arXiv preprint arXiv:2203.05794 (2022).

<https://github.com/MaartenGr/BERTopic>

Topic Modeling

- **Latent Dirichlet Allocation (LDA)**
 - Versatile for large datasets.
- **BERTopic**
 - Advanced, contextual language understanding.
- **Non-Negative Matrix Factorization (NMF)**
 - Fast, effective clustering.
- **Latent Semantic Analysis (LSA)**
 - Latent Semantic Indexing (LSI), Efficient, initial data exploration.
- **Hierarchical Dirichlet Process (HDP)**
 - Adaptable, nonparametric approach.

gensim



Fork me on GitHub



gensim

topic modelling for humans

[Download](#)
latest version from the Python Package Index

[Direct install with:
easy_install -U gensim](#)

[Home](#) [Tutorials](#) [Install](#) [Support](#) [API](#) [About](#)

```
>>> from gensim import corpora, models, similarities
>>>
>>> # Load corpus iterator from a Matrix Market file on disk.
>>> corpus = corpora.MmCorpus('/path/to/corpus.mm')
>>>
>>> # Initialize Latent Semantic Indexing with 200 dimensions.
>>> lsi = models.LsiModel(corpus, num_topics=200)
>>>
>>> # Convert another corpus to the latent space and index it.
>>> index = similarities.MatrixSimilarity(lsi[another_corpus])
>>>
>>> # Compute similarity of a query vs. indexed documents
>>> sims = index[query]
```

Gensim is a FREE Python library

- ✓ Scalable statistical semantics
- ✓ Analyze plain-text documents for semantic structure
- ✓ Retrieve semantically similar documents

spaCy

spaCy

HOME USAGE API DEMOS BLOG

Industrial-Strength Natural Language Processing in Python

Fastest in the world

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research has confirmed that spaCy is the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. I like to think of spaCy as the Ruby on Rails of Natural Language Processing.

Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with [TensorFlow](#), [Keras](#), [Scikit-Learn](#), [Gensim](#) and the rest of Python's awesome AI ecosystem. spaCy helps you connect the statistical models trained by these libraries to the rest of your application.

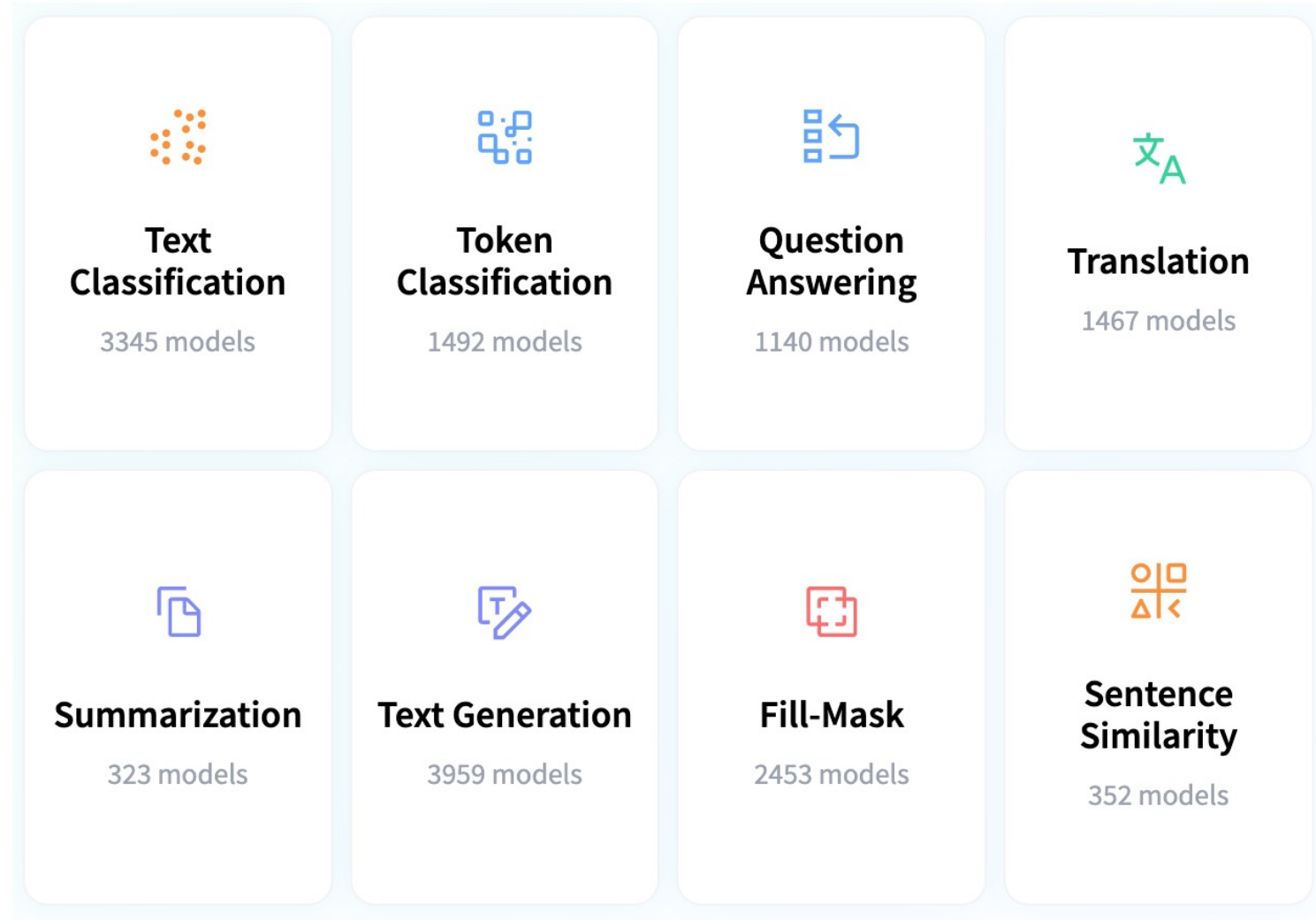
<https://spacy.io/>

NLP Benchmark Datasets

Task	Dataset	Link
Machine Translation	WMT 2014 EN-DE WMT 2014 EN-FR	http://www-lium.univ-lemans.fr/~schwenk/cslm_joint_paper/
Text Summarization	CNN/DM Newsroom DUC Gigaword	https://cs.nyu.edu/~kcho/DMQA/ https://summari.es/ https://www-nlpir.nist.gov/projects/duc/data.html https://catalog.ldc.upenn.edu/LDC2012T21
Reading Comprehension Question Answering Question Generation	ARC CliCR CNN/DM NewsQA RACE SQuAD Story Cloze Test NarrativeQA Quasar SearchQA	http://data.allenai.org/arc/ http://aclweb.org/anthology/N18-1140 https://cs.nyu.edu/~kcho/DMQA/ https://datasets.maluuba.com/NewsQA http://www.qizhexie.com/data/RACE_leaderboard https://rajpurkar.github.io/SQuAD-explorer/ http://aclweb.org/anthology/W17-0906.pdf https://github.com/deepmind/narrativeqa https://github.com/bdhingra/quasar https://github.com/nyu-dl/SearchQA
Semantic Parsing	AMR parsing ATIS (SQL Parsing) WikiSQL (SQL Parsing)	https://amr.isi.edu/index.html https://github.com/jkkummerfeld/text2sql-data/tree/master/data https://github.com/salesforce/WikiSQL
Sentiment Analysis	IMDB Reviews SST Yelp Reviews Subjectivity Dataset	http://ai.stanford.edu/~amaas/data/sentiment/ https://nlp.stanford.edu/sentiment/index.html https://www.yelp.com/dataset/challenge http://www.cs.cornell.edu/people/pabo/movie-review-data/
Text Classification	AG News DBpedia TREC 20 NewsGroup	http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html https://wiki.dbpedia.org/Datasets https://trec.nist.gov/data.html http://qwone.com/~jason/20Newsgroups/
Natural Language Inference	SNLI Corpus MultiNLI SciTail	https://nlp.stanford.edu/projects/snli/ https://www.nyu.edu/projects/bowman/multinli/ http://data.allenai.org/scitail/
Semantic Role Labeling	Proposition Bank OneNotes	http://propbank.github.io/ https://catalog.ldc.upenn.edu/LDC2013T19

Hugging Face Tasks

Natural Language Processing



<https://huggingface.co/tasks>

NLP with Transformers Github

The screenshot shows the GitHub repository page for 'nlp-with-transformers/notebooks'. The repository is public and has 170 forks and 1.1k stars. The main branch is 'main'. The repository contains several files and folders, including a README, a .gitignore, and several Jupyter notebooks (01_introduction.ipynb, 02_classification.ipynb, 03_transformer-anatomy.ipynb, 04_multilingual-ner.ipynb, 05_text-generation.ipynb). The repository is about the book 'Natural Language Processing with Transformers' by Lewis Tunstall, Leandro von Werra, and Thomas Wolf.

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search / Sign in Sign up

nlp-with-transformers / notebooks Public

Notifications Fork 170 Star 1.1k

Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags

Go to file Code

About

Jupyter notebooks for the Natural Language Processing with Transformers book

transformersbook.com/

Readme Apache-2.0 License 1.1k stars 33 watching 170 forks

Releases

No releases published

Packages

lewtun Merge pull request #21 from JingchaoZhang/patch-3	ae5b7c1 15 days ago	71 commits
.github/ISSUE_TEMPLATE	Update issue templates	25 days ago
data	Move dataset to data directory	4 months ago
images	Add README	last month
scripts	Update issue templates	25 days ago
.gitignore	Initial commit	4 months ago
01_introduction.ipynb	Remove Colab badges & fastdoc refs	27 days ago
02_classification.ipynb	Merge pull request #8 from nlp-with-transformers/remove-display-df	26 days ago
03_transformer-anatomy.ipynb	[Transformers Anatomy] Remove cells with figure references	22 days ago
04_multilingual-ner.ipynb	Merge pull request #8 from nlp-with-transformers/remove-display-df	26 days ago
05_text-generation.ipynb	Merge pull request #8 from nlp-with-transformers/remove-display-df	26 days ago

O'REILLY

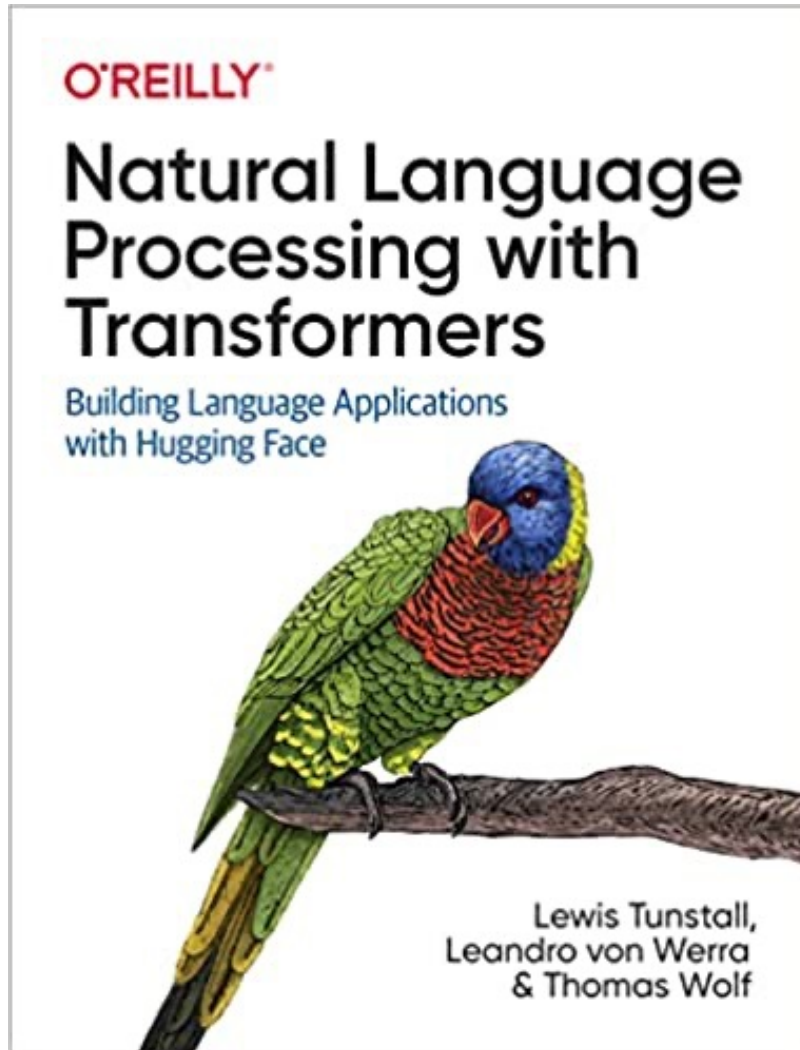
Natural Language Processing with Transformers

Building Language Applications with Hugging Face

Lewis Tunstall, Leandro von Werra & Thomas Wolf

<https://github.com/nlp-with-transformers/notebooks>

NLP with Transformers Github Notebooks



Running on a cloud platform

To run these notebooks on a cloud platform, just click on one of the badges in the table below:

Chapter	Colab	Kaggle	Gradient	Studio Lab
Introduction	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Classification	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Transformer Anatomy	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Multilingual Named Entity Recognition	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Generation	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Summarization	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Question Answering	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Making Transformers Efficient in Production	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Dealing with Few to No Labels	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Training Transformers from Scratch	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Future Directions	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab

Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using [Kaggle](#), [Gradient](#), or [SageMaker Studio Lab](#). These platforms tend to provide more performant GPUs like P100s, all for free!

<https://github.com/nlp-with-transformers/notebooks>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings

RAM Disk Editing

Natural Language Processing with Transformers

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

```
[1] 1 !git clone https://github.com/nlp-with-transformers/notebooks.git
    2 %cd notebooks
    3 from install import *
    4 install_requirements()
```

```
[3] 1 from utils import *
    2 setup_chapter()
```

```
[12] 1 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
      2 from your online store in Germany. Unfortunately, when I opened the package, \
      3 I discovered to my horror that I had been sent an action figure of Megatron \
      4 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
      5 dilemma. To resolve the issue, I demand an exchange of Megatron for the \
      6 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
      7 this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Text Classification

```
[13] 1 from transformers import pipeline
     2 classifier = pipeline("text-classification")
```

```
[14] 1 import pandas as pd
     2 outputs = classifier(text)
     3 pd.DataFrame(outputs)
```

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings A

RAM Disk Editing

Table of contents

- Text Classification with Transformers
 - The Dataset
 - From Datasets to DataFrames
 - From Text to Tokens
 - Character Tokenization
 - Word Tokenization
 - Subword Tokenization
 - Tokenizing the Whole Dataset
 - Training a Text Classifier
 - Transformers as Feature Extractors
 - Extracting the last hidden states
 - Creating a feature matrix
 - Visualizing the training set
 - Training a simple classifier
 - Fine-Tuning Transformers
 - Loading a pretrained model
 - Defining the performance metrics
 - Training the model
- Sidebar: Fine-Tuning with Keras
- Error analysis
- Saving and sharing the model

Text Classification with Transformers

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

```
[10] 1 !nvidia-smi
```

```
1 # Uncomment and run this cell if you're on Colab or Kaggle
2 !git clone https://github.com/nlp-with-transformers/notebooks.git
3 %cd notebooks
4 from install import *
5 install_requirements()
```

```
[12] 1 # hide
2 from utils import *
3 setup_chapter()
```

The Dataset

```
[13] 1 from datasets import list_datasets
2 all_datasets = list_datasets()
3 print(f"There are {len(all_datasets)} datasets currently available on the Hub")
4 print(f"The first 10 are: {all_datasets[:10]}")
```

There are 3783 datasets currently available on the Hub
The first 10 are: ['acronym_identification', 'ade_corpus_v2', 'adversarial_qa',

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>



python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment

Share



A

+ Code + Text

RAM
Disk

Editing



▾ Multilingual Named Entity Recognition (NER)

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

```
[ ] 1 #NER: https://huggingface.co/tasks/token-classification
    2 !pip install transformers
    3 from transformers import pipeline
    4 classifier = pipeline("ner")
    5 classifier("Hello I'm Omar and I live in Zürich.")
```

```
▶ 1 from transformers import pipeline
   2 classifier = pipeline("ner")
   3 classifier("Hello I'm Omar and I live in Zürich.")
```

```
↳ No model was supplied, defaulted to dbmdz/bert-large-cased-finetuned-conll103-english (https://huggingface.co/dbmdz/bert-large-cased-finetuned-conll103-eng)
[{'end': 14,
  'entity': 'I-PER',
  'index': 5,
  'score': 0.99770516,
  'start': 10,
  'word': 'Omar'},
 {'end': 35,
  'entity': 'I-LOC',
  'index': 10,
  'score': 0.9968976,
  'start': 29,
  'word': 'Zürich'}]
```

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus, along with "Comment", "Share", and "Settings" icons. A "Table of contents" sidebar on the left lists various topics, with "Text Similarity" currently selected. The main content area displays three code cells:

- Cell [1]: `!python -m spacy download en_core_web_sm`
- Cell [2]: `!python -m spacy download en_core_web_lg` followed by `# Restart Runtime`
- Cell [3]: `import spacy`, `nlp = spacy.load("en_core_web_lg")`, `tokens = nlp("apple banana cat dog notaword")`, and a `for` loop printing token details.

The output of the third cell shows a table of token information:

apple	True	7.1346846	False
banana	True	6.700014	False
cat	True	6.6808186	False
dog	True	7.0336733	False
notaword	False	0.0	True

Below the code cells, a snippet of Python code is shown for calculating document similarity:

```
1 import spacy
2 nlp = spacy.load("en_core_web_lg")
3 doc1 = nlp("I like cat.")
4 doc2 = nlp("I like dog.")
5 doc1.similarity(doc2)
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings A

RAM Disk Editing ^

Table of contents

- Build the model
- Train the model
- Evaluate the model
- Create a graph of accuracy and loss over time
- Text Classification: BBC News Articles
- Text Summarization and Topic Modeling
 - Text Summarization
 - Text Summarization with Gensim Summarization**
 - Topic Modeling
 - Topic Modeling with Gensim LSI model
 - Topic Modeling with Gensim LDA model
 - Topic Modeling with Scikit-learn LDA and NMF
 - Topic Modeling Visualization

+ Code + Text

Text Summarization with Gensim Summarization

- Source: Text Summarization with Gensim Summarization : https://radimrehurek.com/gensim/auto_examples/tutorials/run_summarization.html

```
1 from pprint import pprint as print
2 from gensim.summarization import summarize
```

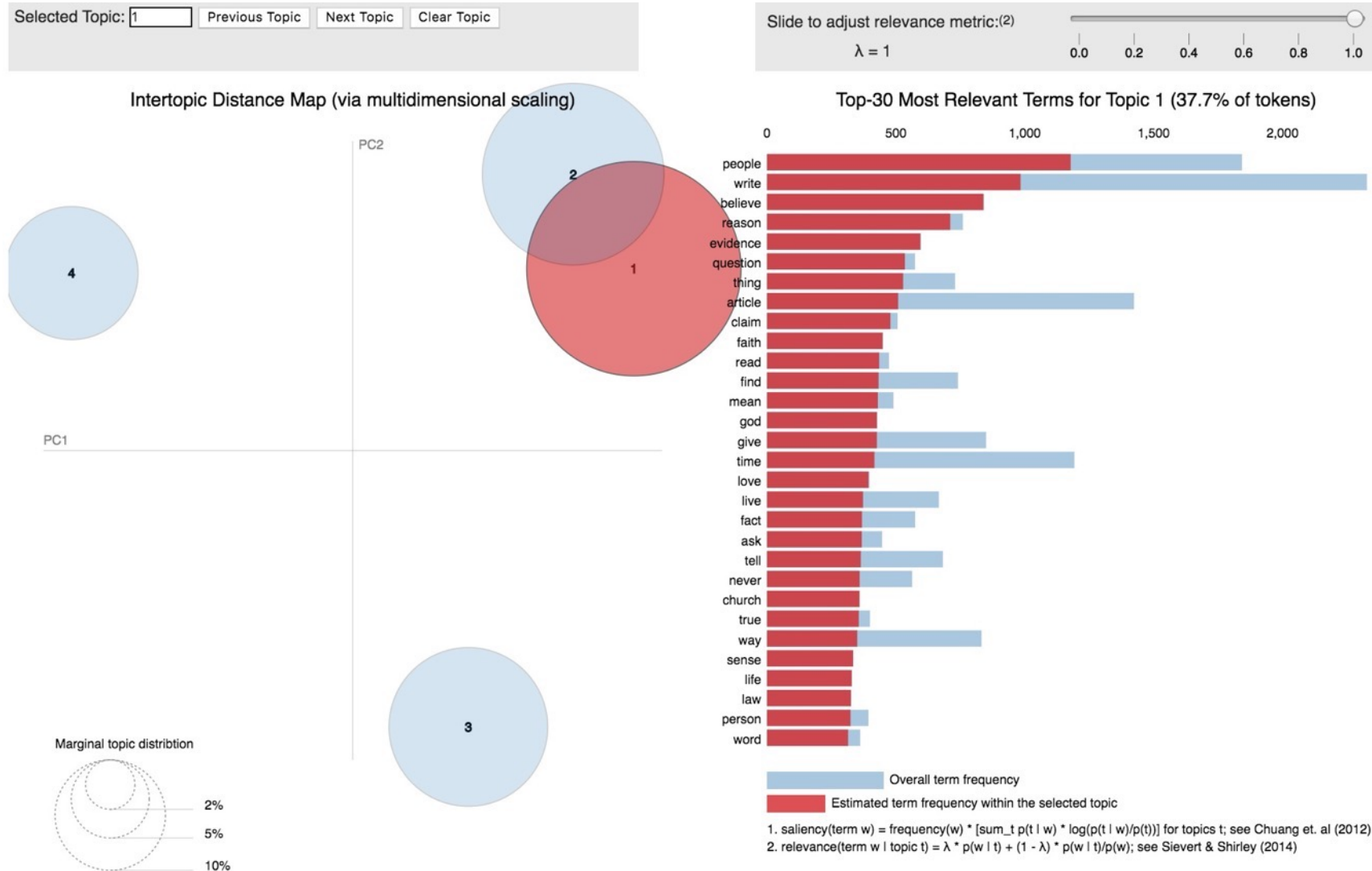
```
[ ] 1 text = (
2     "Thomas A. Anderson is a man living two lives. By day he is an "
3     "average computer programmer and by night a hacker known as "
4     "Neo. Neo has always questioned his reality, but the truth is "
5     "far beyond his imagination. Neo finds himself targeted by the "
6     "police when he is contacted by Morpheus, a legendary computer "
7     "hacker branded a terrorist by the government. Morpheus awakens "
8     "Neo to the real world, a ravaged wasteland where most of "
9     "humanity have been captured by a race of machines that live "
10    "off of the humans' body heat and electrochemical energy and "
11    "who imprison their minds within an artificial reality known as "
12    "the Matrix. As a rebel against the machines, Neo must return to "
13    "the Matrix and confront the agents: super-powerful computer "
14    "programs devoted to snuffing out Neo and the entire human "
15    "rebellion. "
16 )
17 print(text)
```

```
[ ] ('Thomas A. Anderson is a man living two lives. By day he is an average '
'computer programmer and by night a hacker known as Neo. Neo has always '
'questioned his reality, but the truth is far beyond his imagination. Neo '
```

<https://tinyurl.com/aintpuython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>



<https://tinyurl.com/aintpuython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled 'python101.ipynb' and has a star icon. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', with 'Saving...' in progress. On the right, there are icons for 'Comment', 'Share', 'Settings', and a user profile 'A'. Below the menu, there are tabs for '+ Code' and '+ Text'. A status bar shows 'RAM' and 'Disk' usage, and 'Editing' mode is active.

The notebook content is titled 'Text Summarization' and includes a list of sources:

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

The first code cell contains the following Python code:

```
1 #Source: https://huggingface.co/tasks/summarization
2 !pip install transformers
3 from transformers import pipeline
4 classifier = pipeline("summarization")
5 text = "Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents as of 2018, in an area of more than 105 km² (41 sq mi) and a metropolitan area population of over 12 million. Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents as of 2018, in an area of more than 105 km² (41 sq mi) and a metropolitan area population of over 12 million."
6 classifier(text, max_length=30)
```

The output of this cell shows an error message: "No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 (<https://huggingface.co/sshleifer/distilbart-cnn-12-6>) Your min_length=56 must be inferior than your max_length=30." followed by the JSON output: [{"summary_text": " Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents . The City of Paris"}].

The second code cell contains the following Python code:

```
1 #!pip install transformers
2 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
3 from your online store in Germany. Unfortunately, when I opened the package, \
4 I discovered to my horror that I had been sent an action figure of Megatron \
5 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
6 dilemma. To resolve the issue, I demand an exchange of Megatron for the \
7 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
8 this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
9 from transformers import pipeline
10 summarizer = pipeline("summarization")
11 outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
12 print(outputs[0]['summary_text'])
```

<https://tinyurl.com/aintpupython101>

Text Summarization

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

```
from transformers import pipeline
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.

Summary

- **Text Similarity**

- Analyzing and quantifying the likeness between text documents.

- **Text Clustering**

- Grouping similar text documents using various algorithms.

- **Text Summarization**

- Condensing text data into a shorter, coherent form.

- **Topic Models**

- Identifying underlying themes or topics within text collections.

References

- Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Denis Rothman (2021), Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more, Packt Publishing.
- Savaş Yildirim and Meysam Asgari-Chenaghlu (2021), Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques, Packt Publishing.
- Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta (2020), Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems, O'Reilly Media.
- Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed (2021). "Automatic text summarization: A comprehensive survey." Expert Systems with Applications 165 (2021): 113679.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2019). "Exploring the limits of transfer learning with a unified text-to-text transformer." arXiv preprint arXiv:1910.10683 (2019).
- Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu (2020). "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization." In International Conference on Machine Learning, pp. 11328-11339. PMLR, 2020.
- Uttam Chauhan, and Apurva Shah (2021). "Topic modeling using latent Dirichlet allocation: A survey." ACM Computing Surveys (CSUR) 54, no. 7 (2021): 1-35.
- Maarten Grootendorst (2022). "BERTopic: Neural topic modeling with a class-based TF-IDF procedure." arXiv preprint arXiv:2203.05794 (2022).
- Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning, O'Reilly.
- Selva Prabhakaran (2020), Topic modeling visualization – How to present the results of LDA models?, <https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/>
- Rajesh Arumugam (2018), Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications, Packt.
- Jake VanderPlas (2016), Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805.
- The Super Duper NLP Repo, <https://notebooks.quantumstat.com/>
- Jay Alammar (2018), The Illustrated Transformer, <http://jalammar.github.io/illustrated-transformer/>
- Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, <http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/>
- NLP with Transformer, <https://github.com/nlp-with-transformers/notebooks>
- Min-Yuh Day (2023), Python 101, <https://tinyurl.com/aintpupython101>