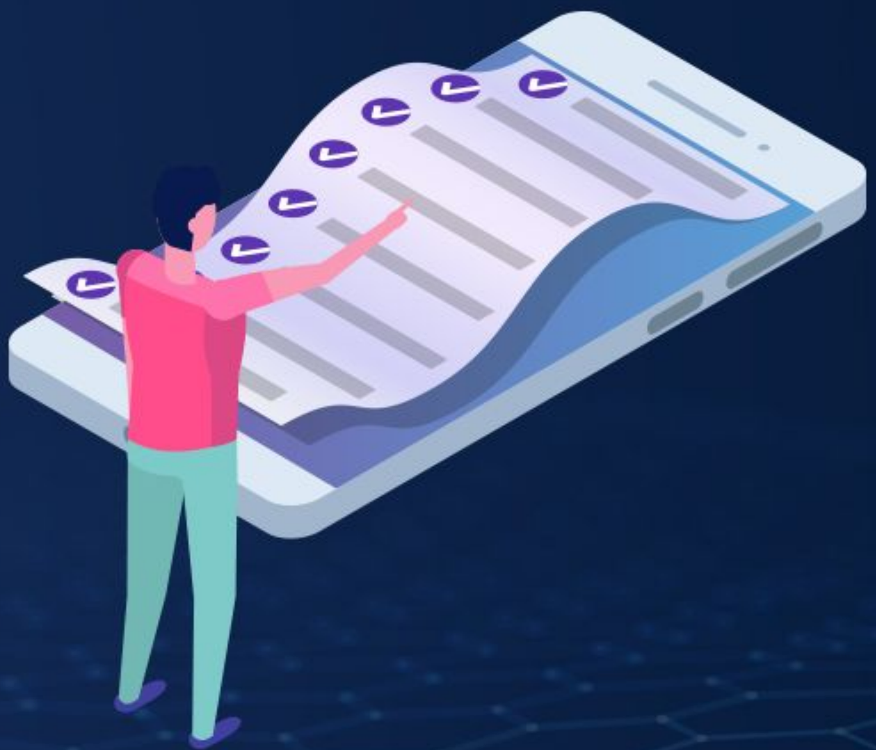
Natural Language Processing

Feature Engineering on Text Data

Learning Objectives

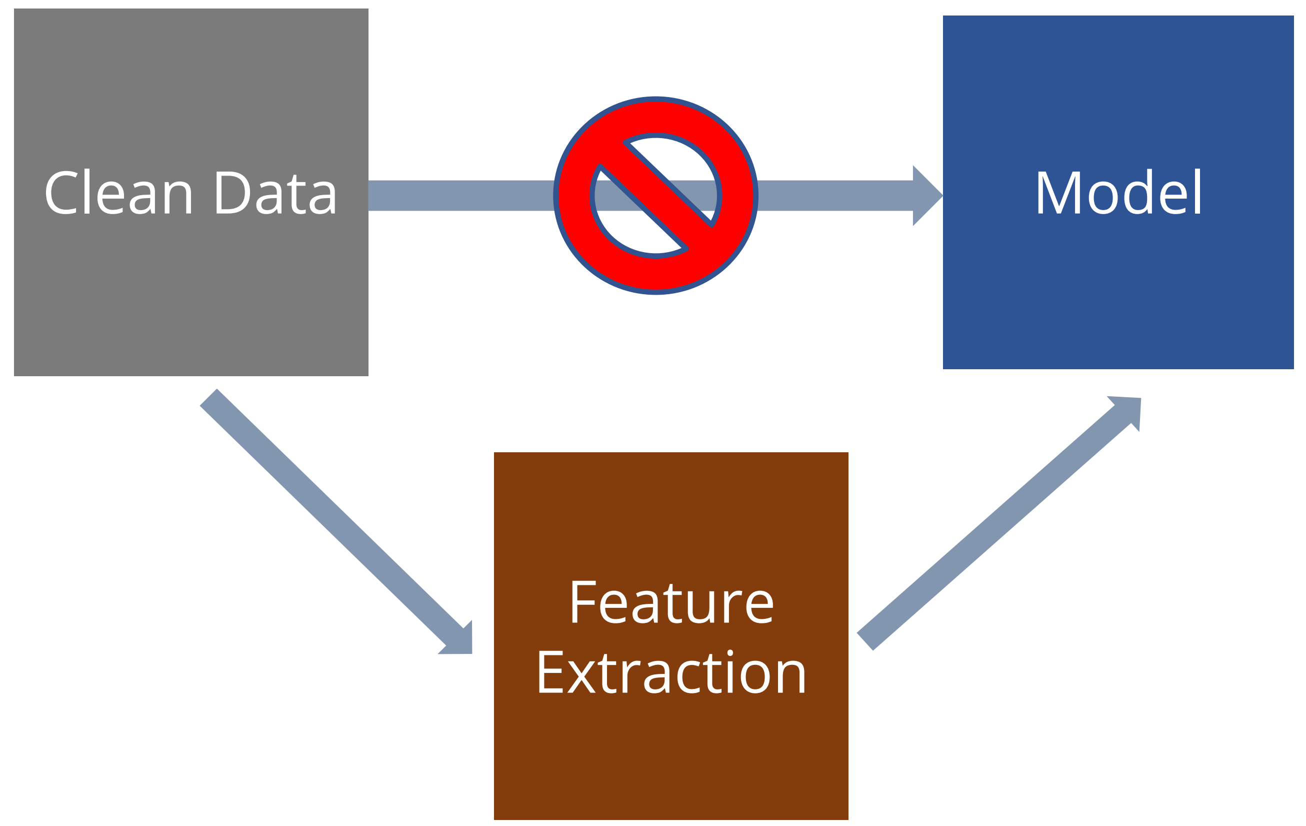
By the end of this lesson, you will be able to:

- Explain N-gram
- Demonstrate the different word embedding models
- Perform operations on word analogies
- Demonstrate the working of Bag-of-Words



Feature Extraction

What Is Feature Extraction?

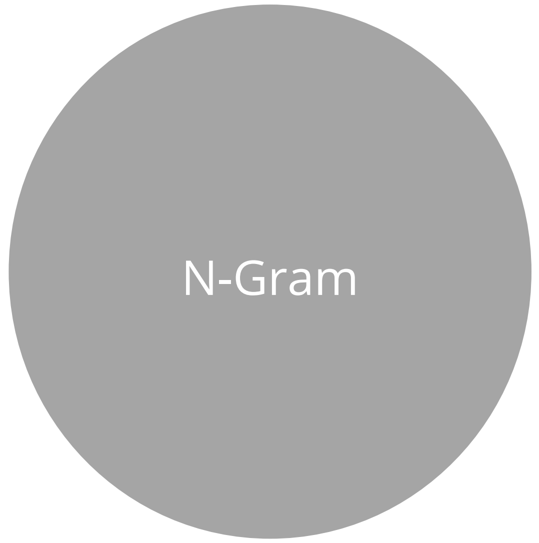


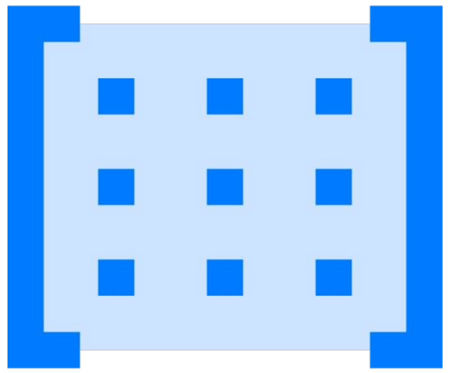
Computers do not have any standard representation of words

Once the text is cleaned and normalized, it needs to be transformed into features which can be used for modeling

Feature Extraction Techniques

Feature extraction technique depends on what kind of model is intended to be used.





TF-IDF

Bag-of-Words

N-Gram

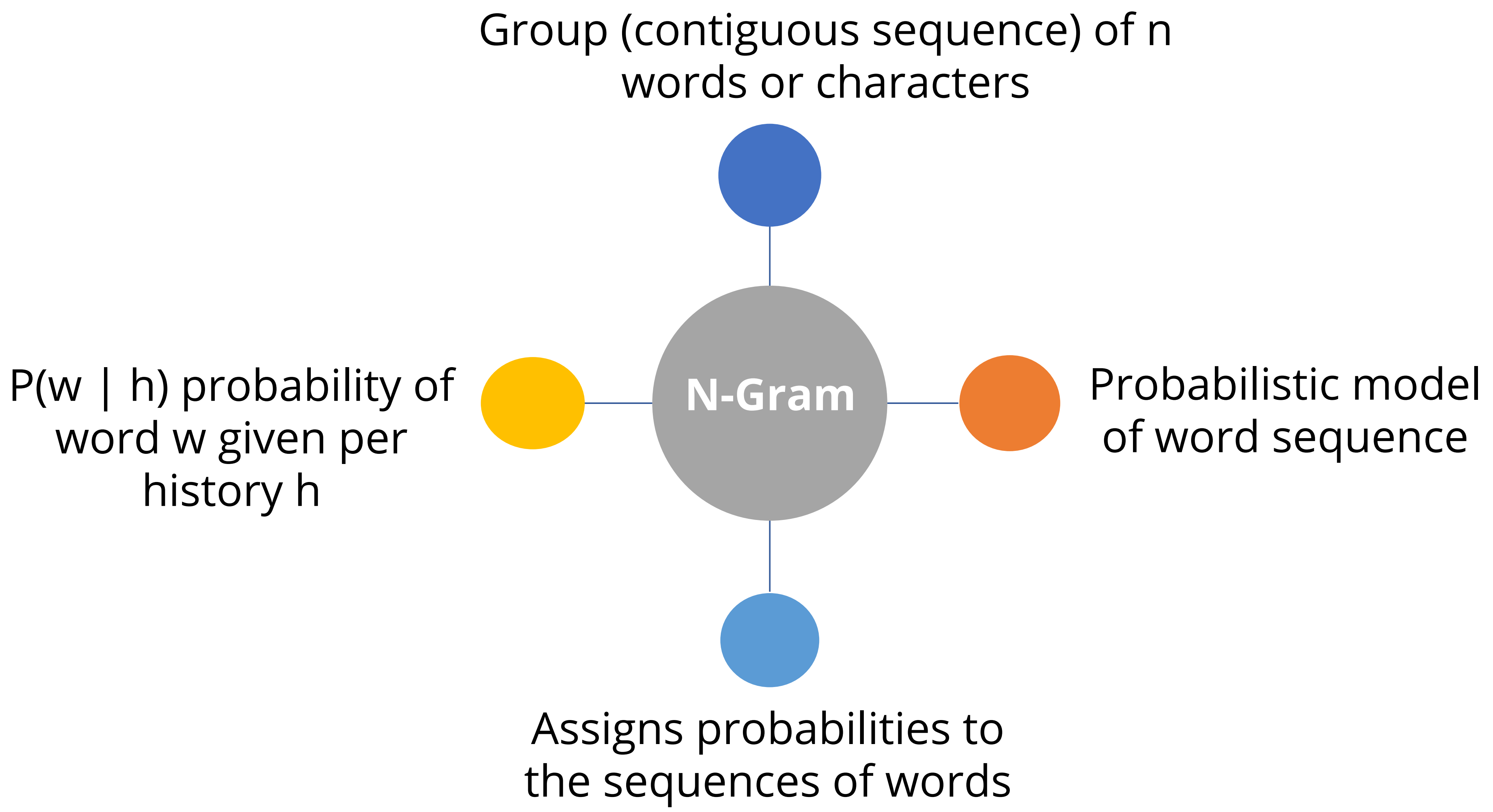
Document-Term Matrix

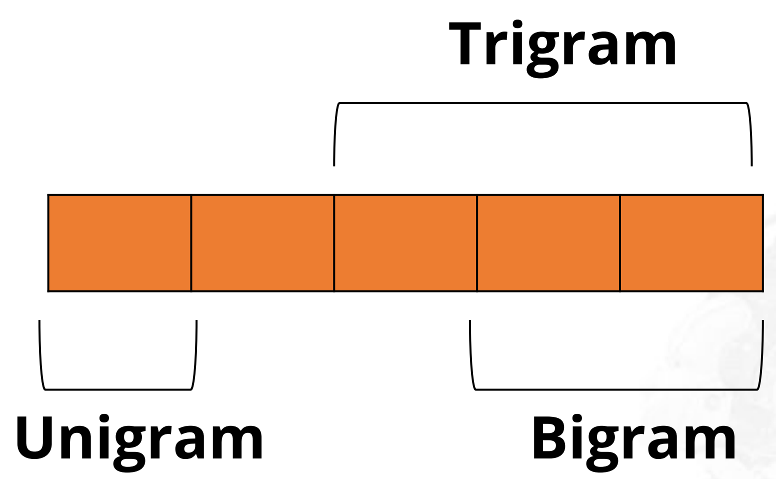
TF-IDF

N-Gram

N-Gram: Introduction

N-grams are combinations of adjacent words or letters of length n in the source text.


Group (contiguous sequence) of n words or characters
P(w | h) probability of word w given per history h
N-Gram
Probabilistic model of word sequence
Assigns probabilities to the sequences of words

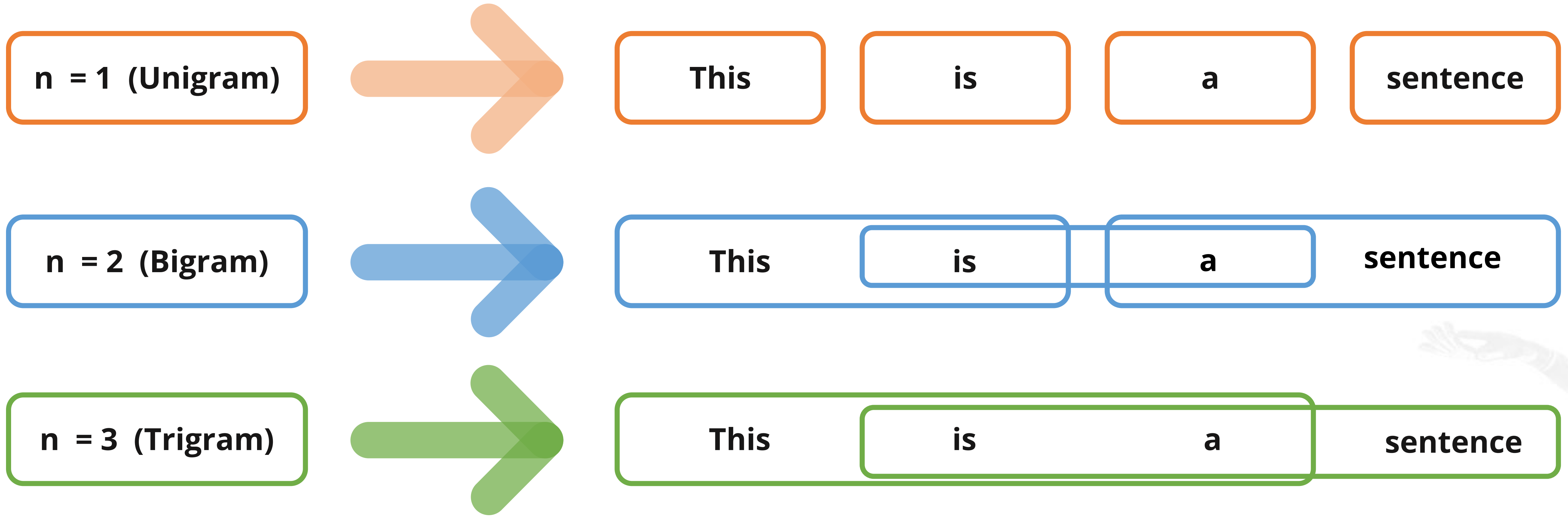


n >= 1

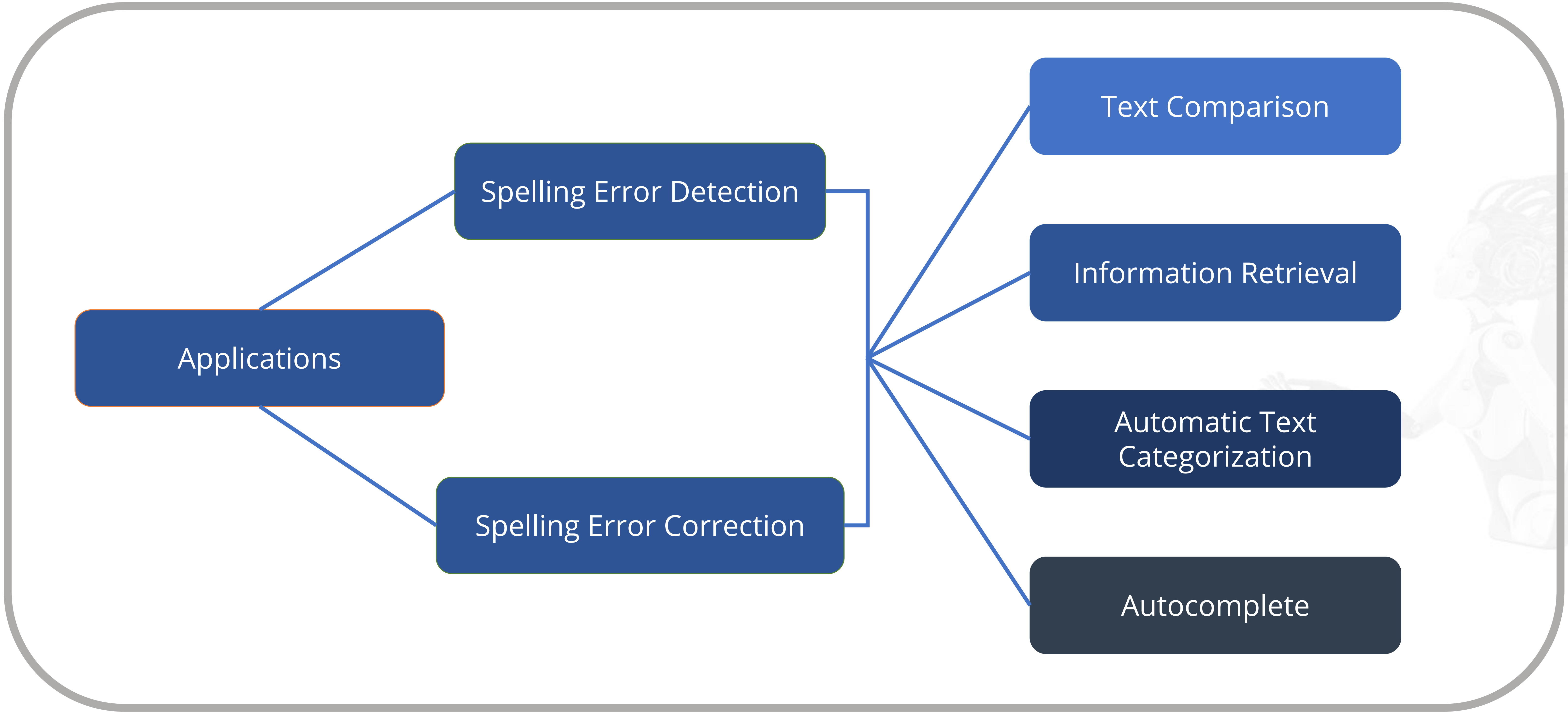
n = 1	Unigram
n = 2	Bigram
n = 3	Trigram
. .	.
. .	.
n = n	N-Gram

N-Gram: Example

Example: This is a sentence



N-Gram: Applications



Bag-of-Words

Bag-of-Words

1 Used to perform document-level task

2 Is a vectorization technique to represent text data

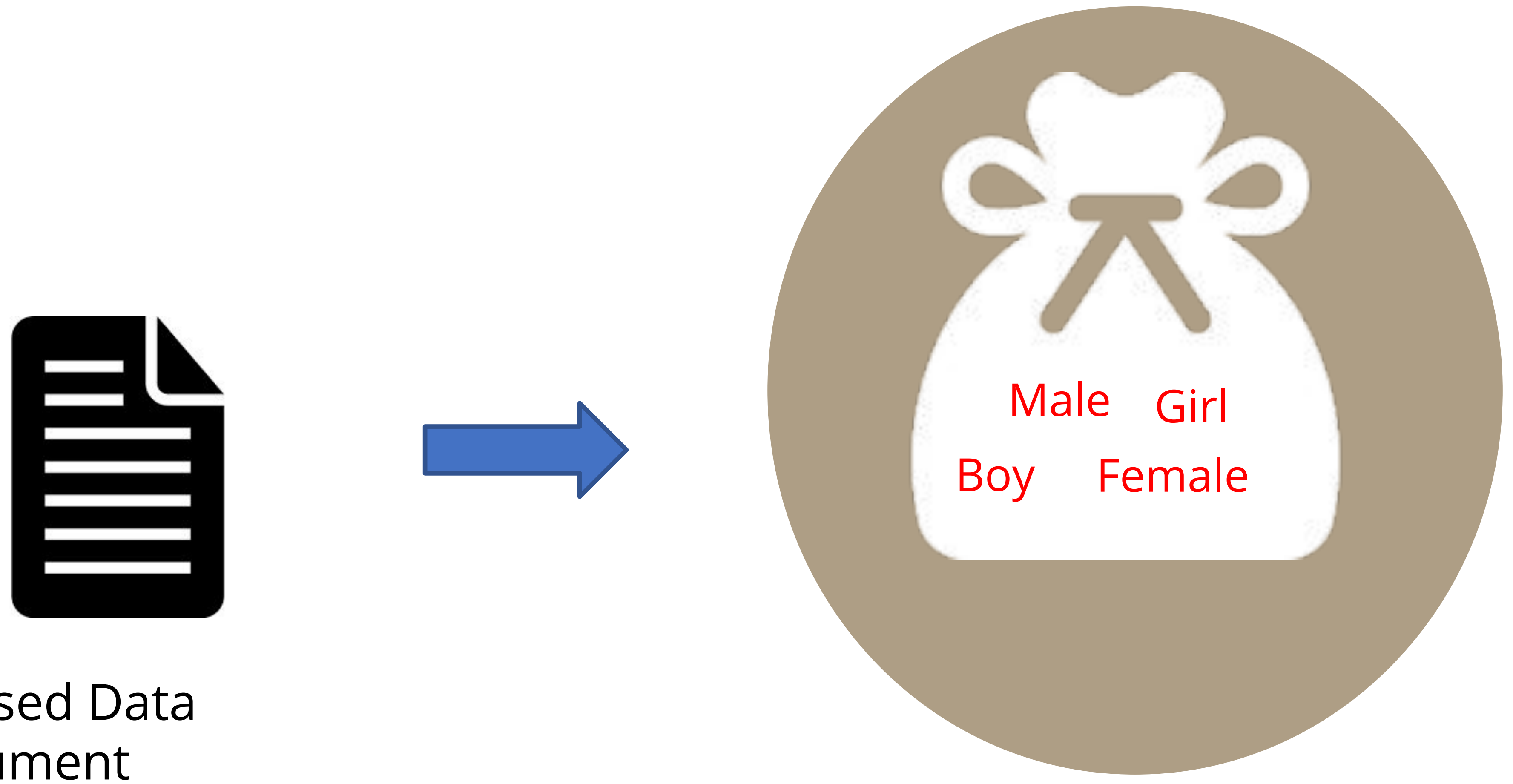
3 Has no effect of grammar and order of words in sentence

Example Usage:

Sentiment Analysis

Spam Detection

Bag-of-Words

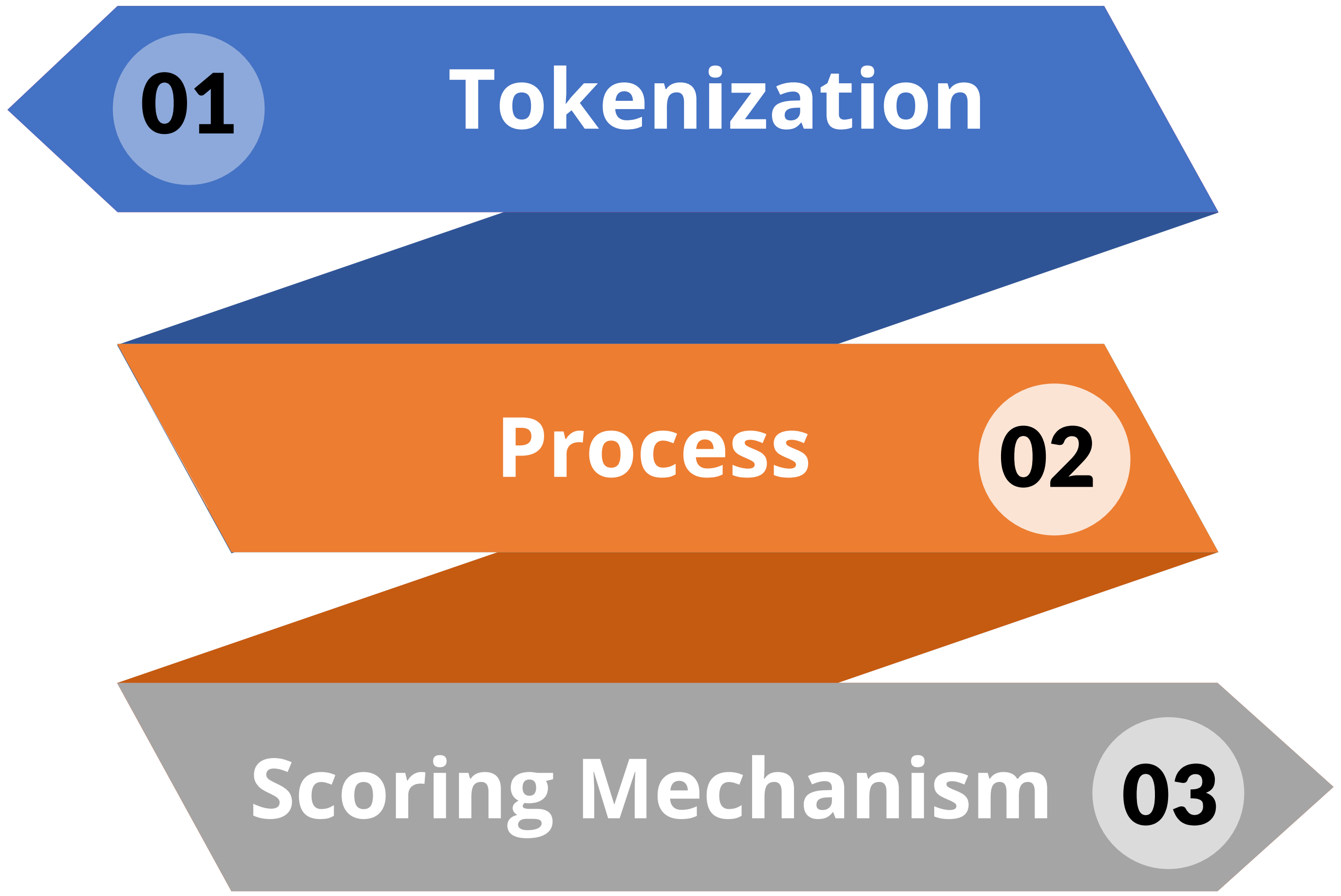


Processed Data
- Document
- Tweet
- Review comments

Unordered collection of words

Bag-of-Words

Bag-of-Words model is the way of extracting features from text and representing the text data, while modeling the text with a machine learning algorithm.

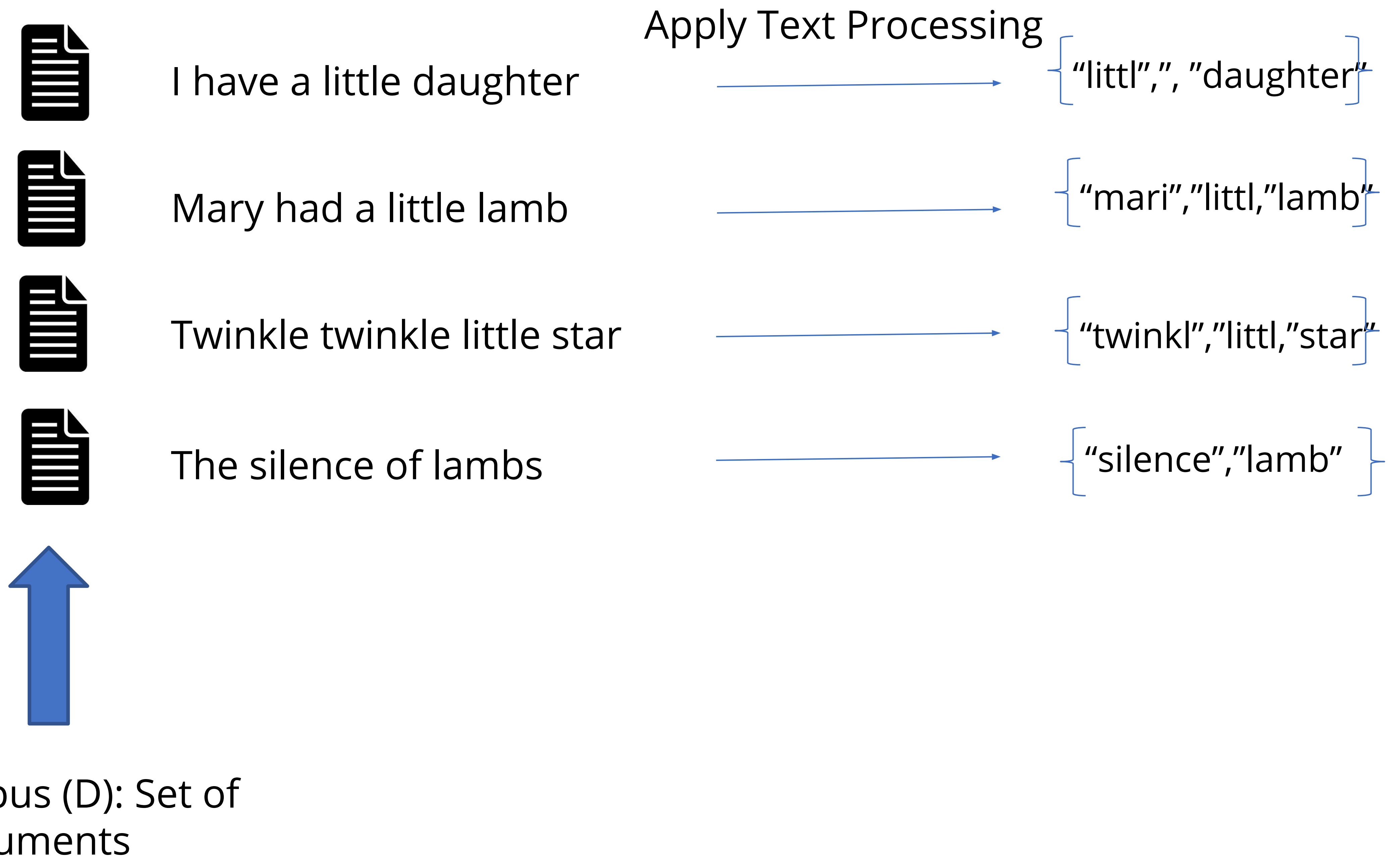


Tokenization:
While creating the bag of words, tokenized word of each observation is used.

Process:
- Collect data
- Create a vocabulary by listing all unique words
- Create document vectors after scoring

Scoring mechanism:
- Word hashing
- TF-IDF
- Boolean value

Bag-of-Words: Example



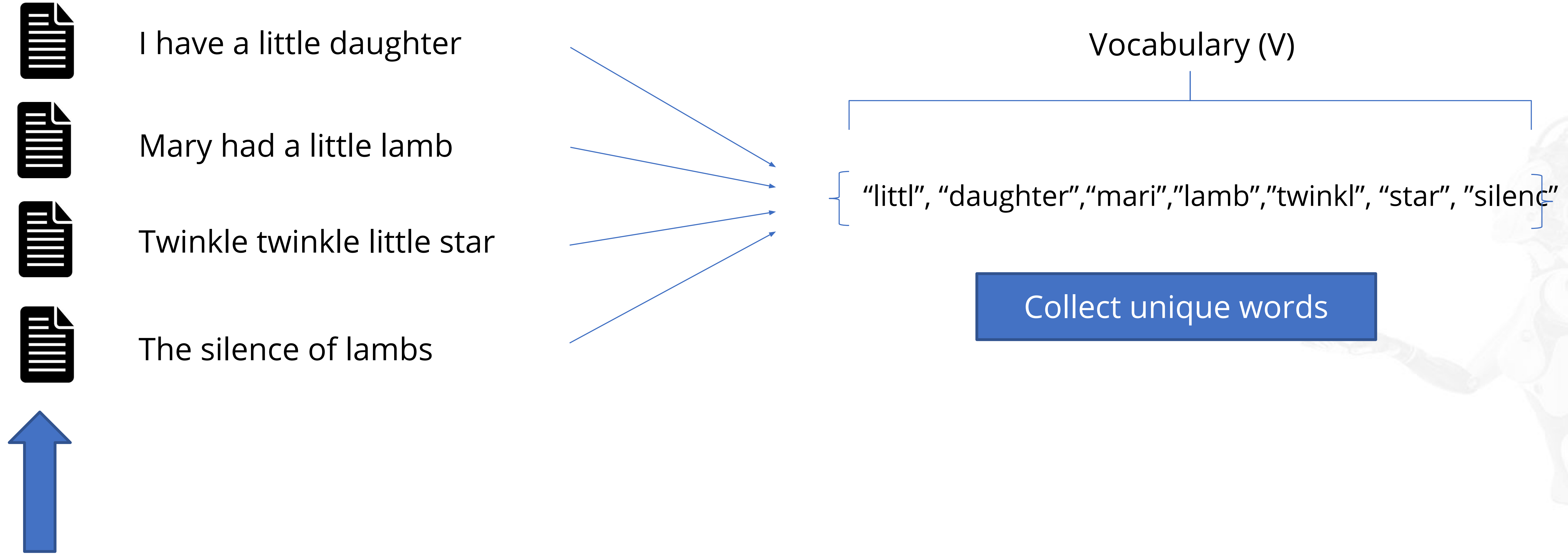
Corpus (D): Set of Documents

Inefficient

Difficult to compare

Multiple occurrences of word: difficult to handle

Bag-of-Words: Example



I have a little daughter

Mary had a little lamb

Twinkle twinkle little star

The silence of lambs

Vocabulary (V)

{ “littl”, “daughter”,“mari”,”lamb”,”twinkl”, “star”, ”silenc” }

Collect unique words

Corpus (D): Set of Documents

Bag-of-Words: Vector Representation Example

Term or Word

	daughter	lamb	littl	mari	star	silenc	twinkl
I have a little daughter	1	0	1	0	0	0	0
Mary had a little lamb	0	1	1	1	0	0	0
Twinkle twinkle little star	0	0	1	0	1	0	2
The silence of lambs	0	1	0	0	0	1	0

Corpus (D): Set of Documents

Term Frequency

Frequency of a term or word-occurrence in a document

Document-Term Matrix

Bag-of-Words: Recap of Terms Used

Term

Each processed word is called term

Term Frequency

Frequency of a term-occurrence in a document

Term Matrix

Matrix showing frequency of each term-occurrence in documents

TF-IDF

TF-IDF

The Term Frequency-Inverse Document Frequency is abbreviated as TF-IDF

- Bag-of-Words assumes that each word is equally important
- In real-world scenario, each word has its own weight based on the context

Example:

- Cost occurs more frequently in an economy related document. To overcome this limitation TF-IDF is used which assigns weights to the words based on their relevance in the document.

TF-IDF

It represents the numerical statistics

It has two parts:
- Term Frequency (TF)
- Inverse Document Frequency (IDF)

Applications of TF-IDF are:
- Text Mining
- User Modeling

TF-IDF: Example

	Random	Forest	is	an	ensemble	learning	method	machine	technique	application	of	ai
Doc1	1	1	1	1	1	1	1	0	0	0	0	0
Doc2	0	0	1	0	1	1	1	1	1	0	0	0
Doc3	0	0	1	1	0	1	0	1	0	1	1	1

Document Frequency	1	1	3	2	2	3	2	2	1	1	1	1

Sum of occurrence of a word across documents

TF-IDF: Example

	Random	Forest	is	an	ensemble	learning	method	machine	technique	application	of	ai
Doc1	1/1	1/1	1/3	1/2	1/2	1/3	1/2	0/2	0/1	0/1	0/1	0/1
Doc2	0/1	0/1	1/3	0/2	1/2	1/3	1/2	1/2	1/1	0/1	0/1	0/1
Doc3	0/1	0/1	1/3	1/2	0/2	1/3	0/2	1/2	0/1	1/1	1/1	1/1

1	1	3	2	2	3	2	2	1	1	1	1

Document Frequency

↓

Sum of occurrence of a word across documents

Term Frequency

TF-IDF: Example

	Random	Forest	is	an	ensemble	learning	method	machine	technique	application	of	ai
Doc1	1	1	1/3	1/2	1/2	1/3	1/2	0	0	0	0	0
Doc2	0	0	1/3	0	1/2	1/3	1/2	1/2	1	0	0	0
Doc3	0	0	1/3	1/2	0	1/3	0	1/2	0	1	1	1

Term Frequency ⇨

- Is proportional to frequency of occurrence of a word or term in a document
- Is inversely proportional to the number of documents in which a word or term occurs

TF-IDF: Example

	Random	Forest	is	an	ensemble	learning	method	machine	technique	application	of	ai
Doc1	1	1	1/3	1/2	1/2	1/3	1/2	0	0	0	0	0
Doc2	0	0	1/3	0	1/2	1/3	1/2	1/2	1	0	0	0
Doc3	0	0	1/3	1/2	0	1/3	0	1/2	0	1	1	1

Term Frequency

- Highlights the words or terms which are unique to the document
- These words are better for characterizing

TF-IDF

TF-IDF = TF(t,d) * IDF(t,D)
t is terms
d is document

TF = Term Frequency
IDF = Inverse Document Frequency

TF= count(t,d)	Count of term ‘t’ in document ‘d’

|d|	Total number of terms in document ‘d’
IDF = log(|D|)	Log of total number of documents in collection ‘D’

|{d ⊂ D} : {t ⊂ d}	Number of documents where ‘t’ is present

TF-IDF

TF

Term Frequency (TF)

Frequent occurrence of a term in a document is measured by term frequency.
TF (t, d) = Number of times t appears in document d / Total number of terms in the document d

IDF

Inverse Document Frequency (IDF)

IDF measures how important a term is.
IDF (t) = Log_e (Total number of documents / Number of documents with term t in it)

TF-IDF = TF (t,d) * IDF (t)
t is term
d is document

One-Hot Encoding

One-Hot Encoding

1. Used for deeper analysis of text
2. Performs numerical representation of each word
3. Used for categorical data
4. Higher the distinct categorical value, higher the sparsity

One-Hot Encoding

Treats each word as class

Assigns vector value 1 where the particular word is present and 0 at other places

How does it work?

One-Hot Encoding: Example

	daughter	lamb	littl	mari	star	silenc	twinkl
lamb	0	1	0	0	0	0	0
littl	0	0	1	0	0	0	0
silenc	0	0	0	0	0	1	0
twinkl	0	0	0	0	0	0	1

Word2vec

Word2vec

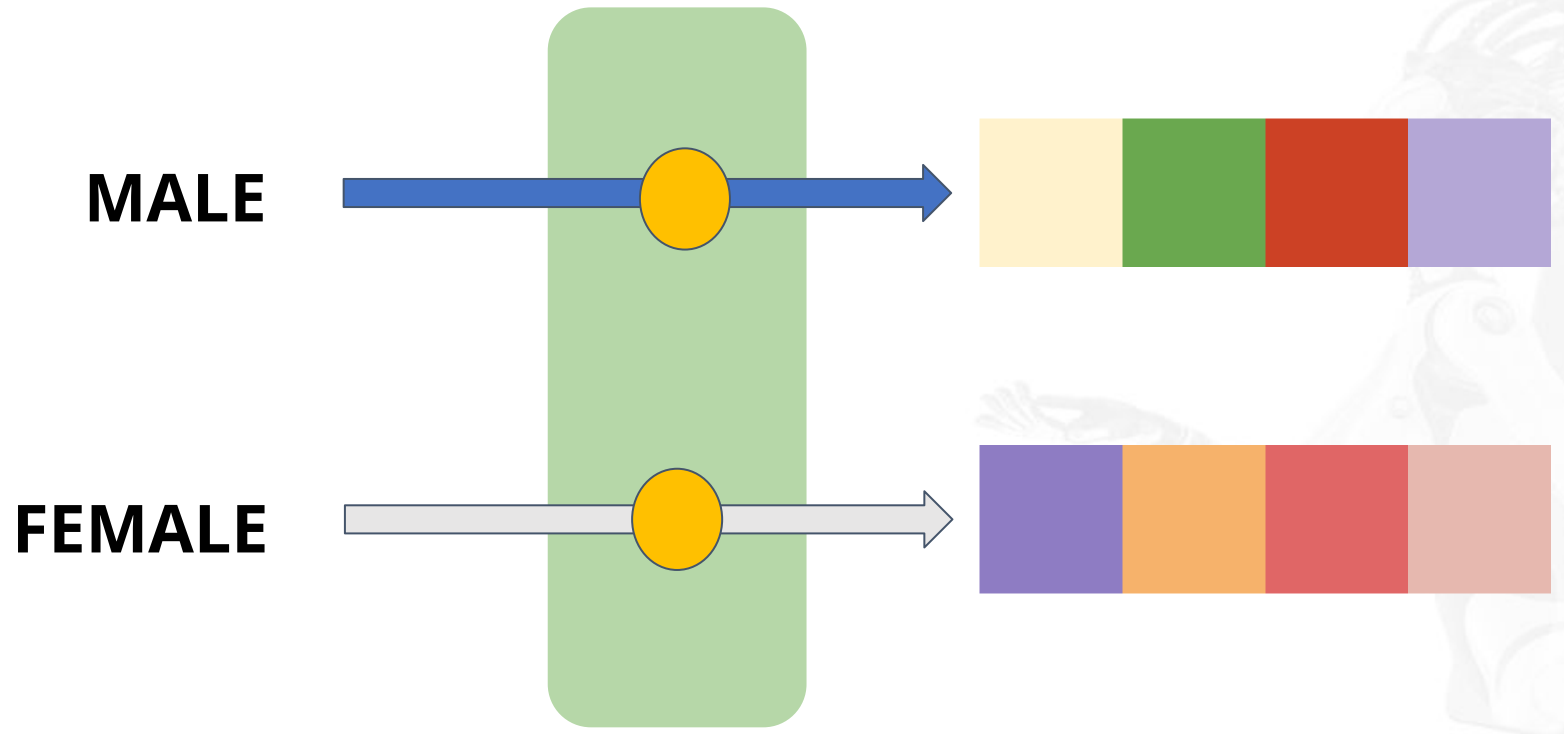
Word2vec is one of the most popular techniques of word embedding.

Word2vec is a two-layer neural network.

Word2vec

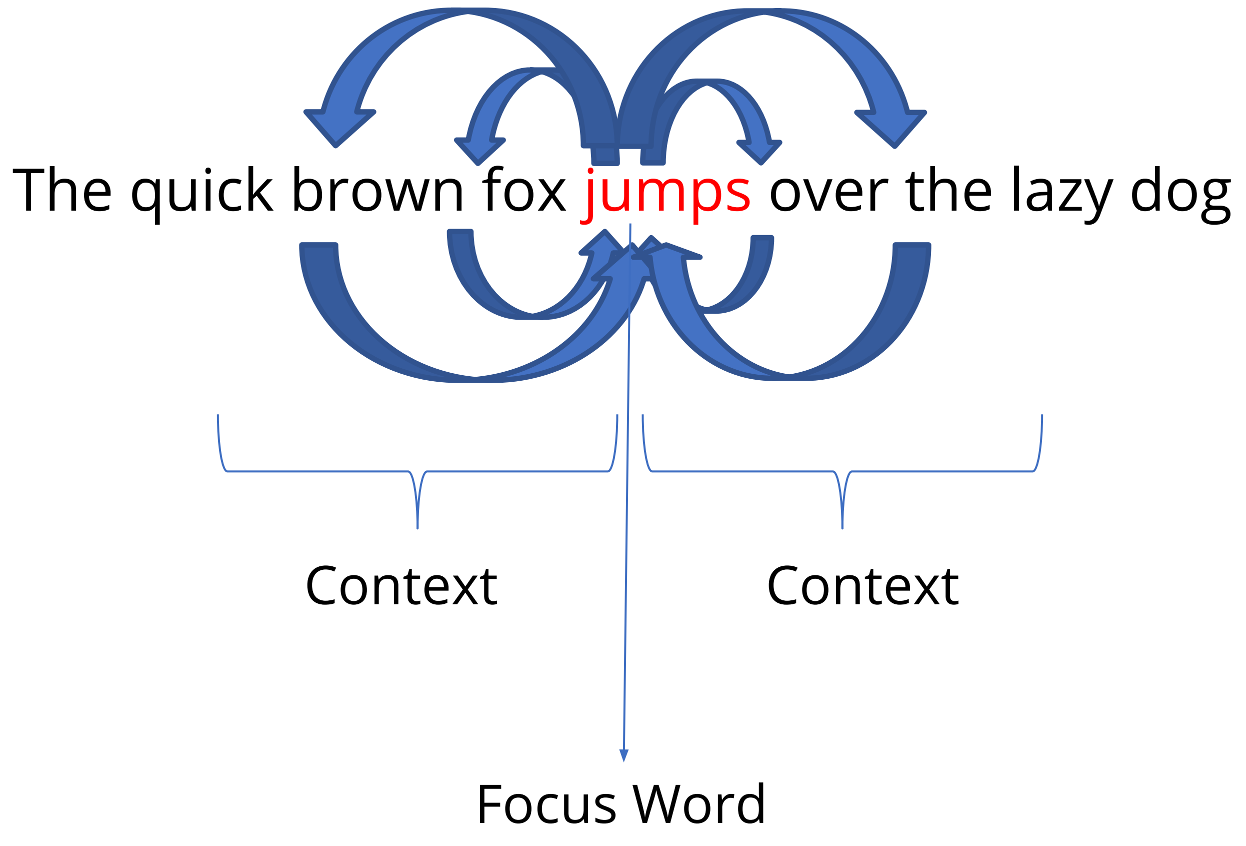
Input is text corpus and output is set of vectors.

Two flavors of algorithm:
- Continuous Bag-of-Words (CBOW)
- Skip-Gram

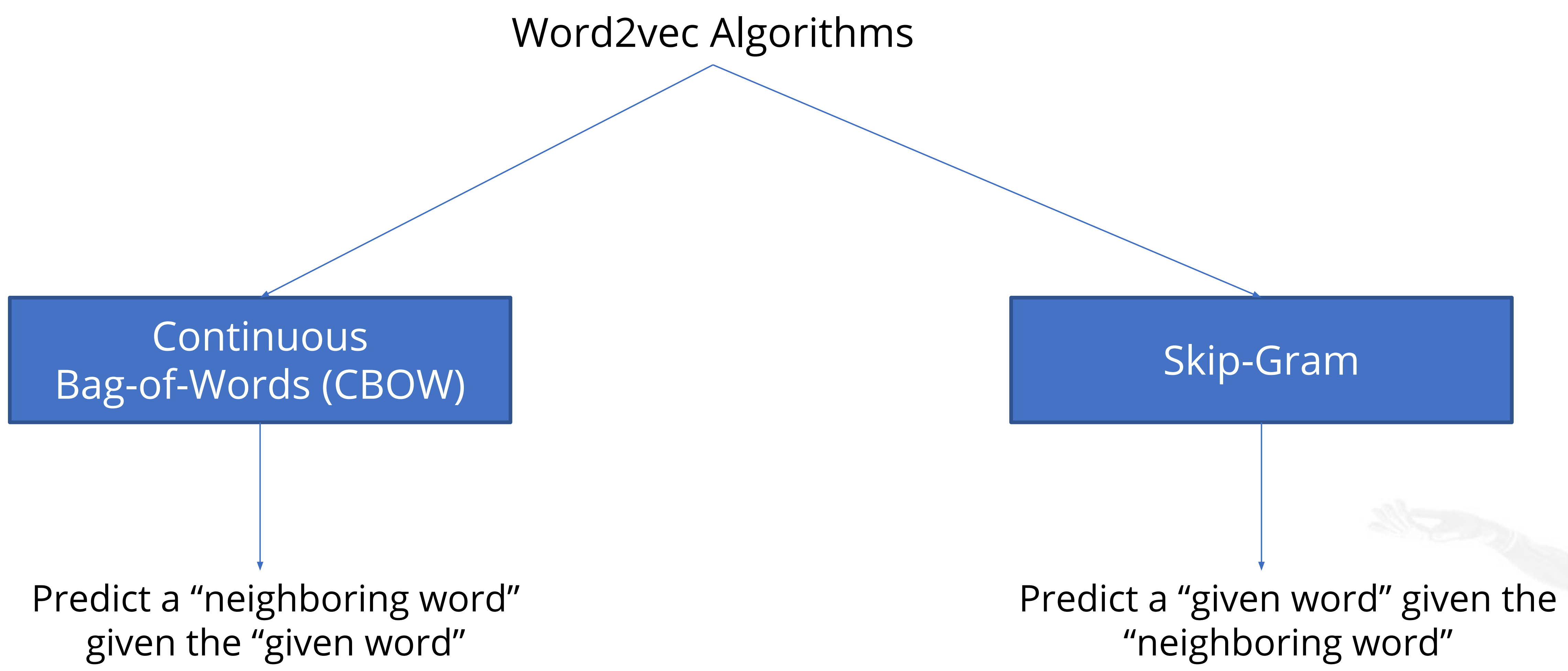


Word2vec

The core concept of Word2vec approach is to predict a word with the given neighboring word or predict a neighboring word with the given word which is likely to capture the contextual meaning of the word.

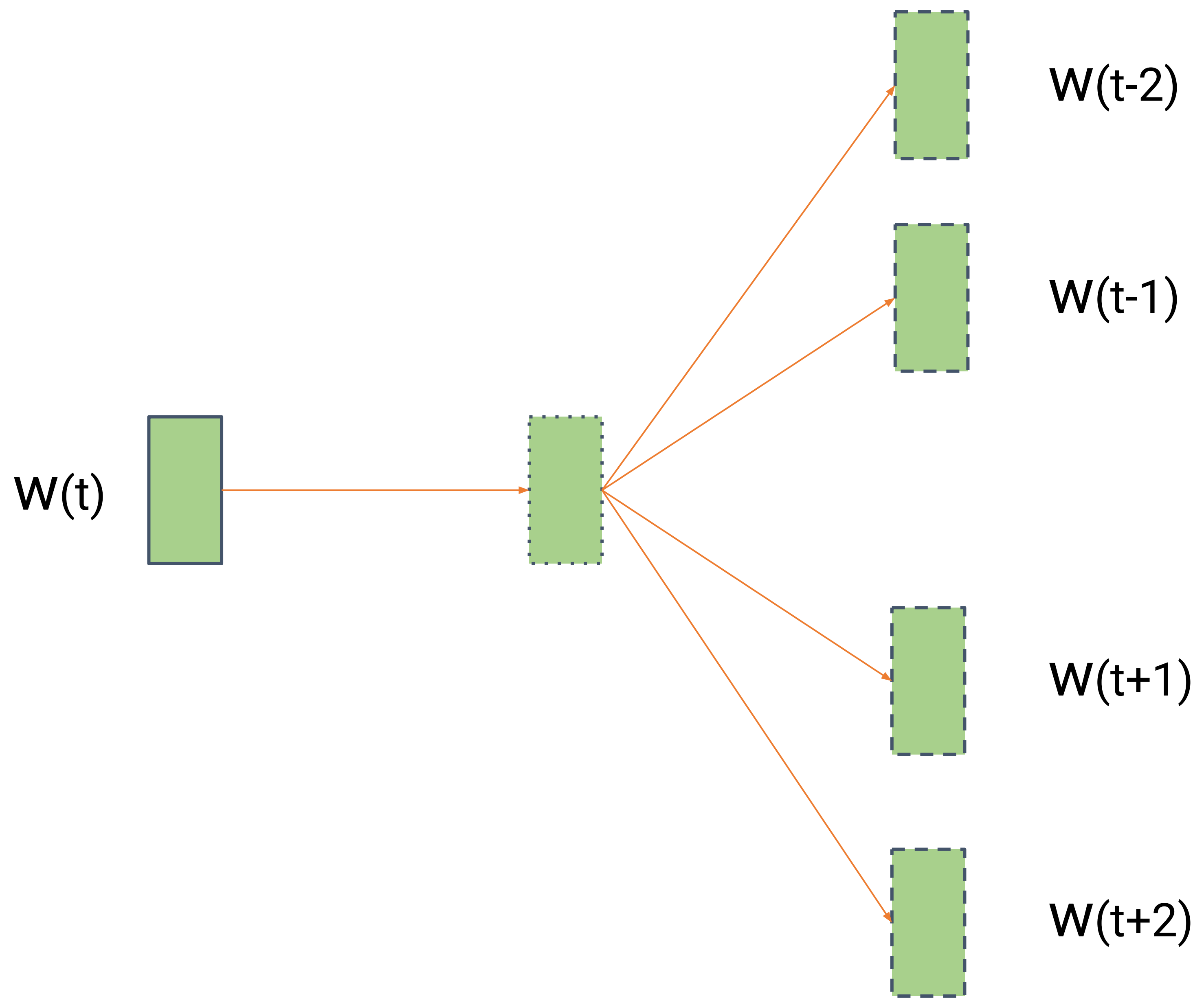


Word2vec Algorithms

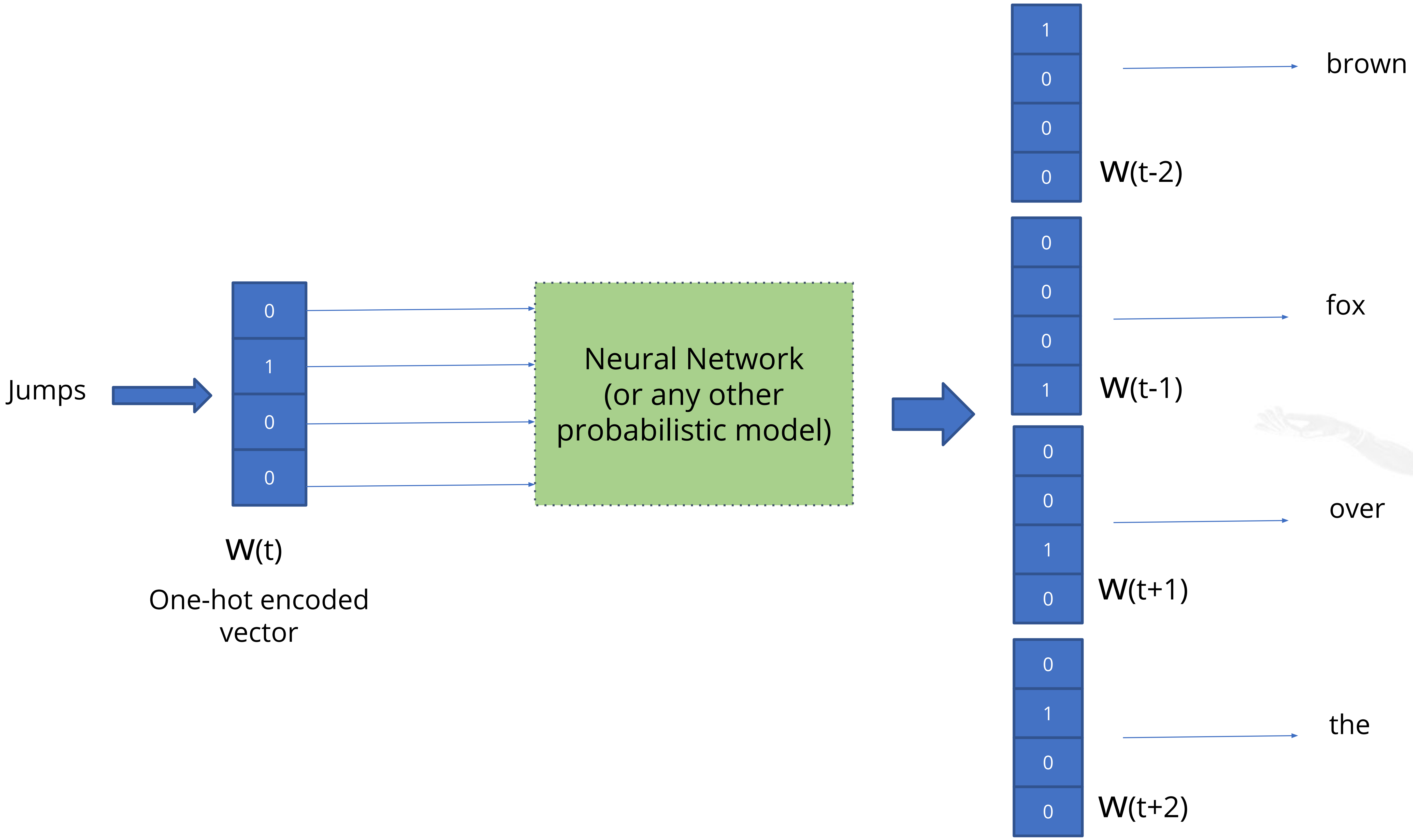

Word2vec Algorithms
Continuous Bag-of-Words (CBOW)
Skip-Gram
Predict a “neighboring word” given the “given word”
Predict a “given word” given the “neighboring word”

Skip-Gram Model

It is used to predict the source context words given in a target word.

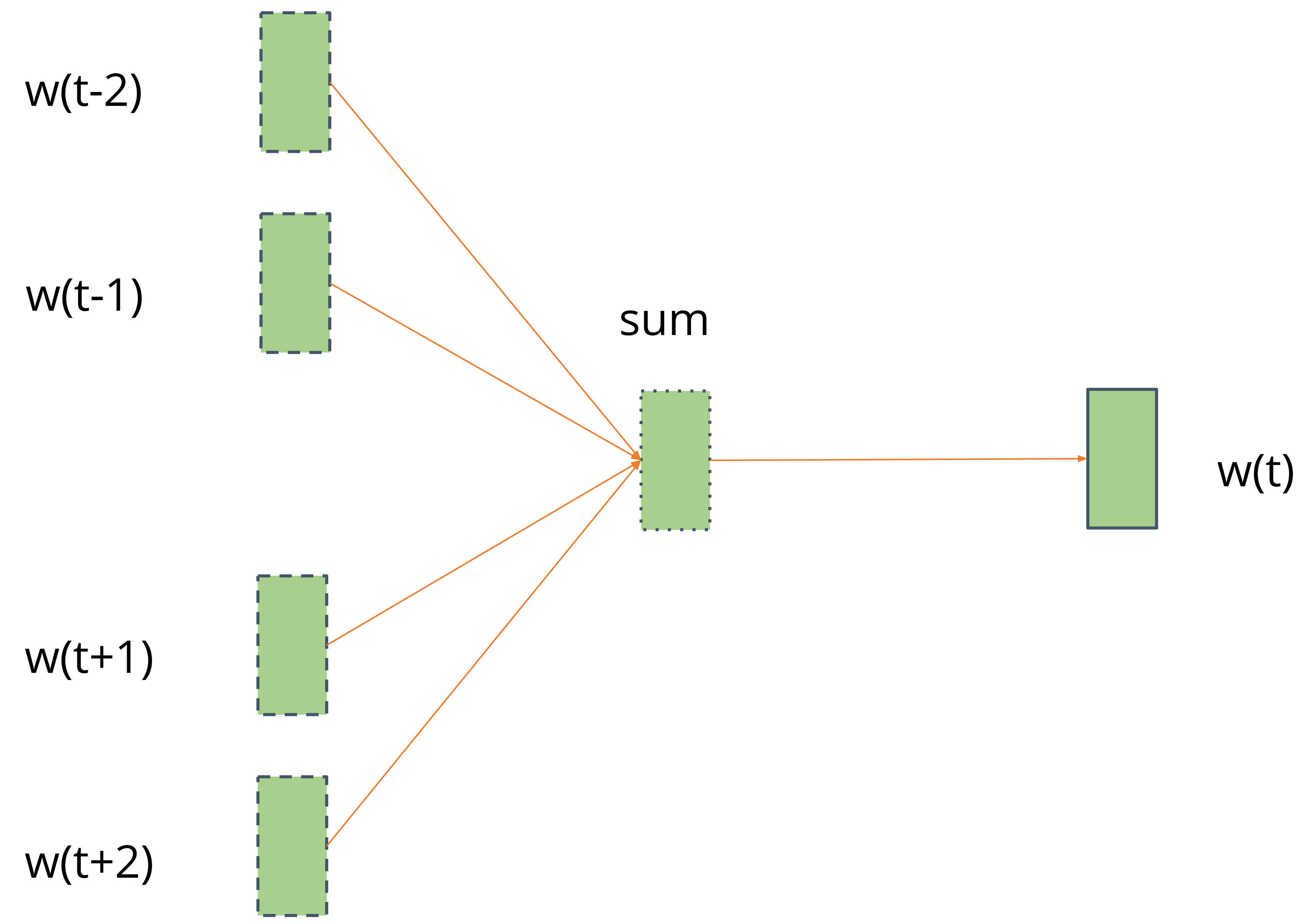


Skip-Gram Model: Example



CBOW Model

Common Bag-of-Words (CBOW) algorithm is used to predict the target word in the given context.



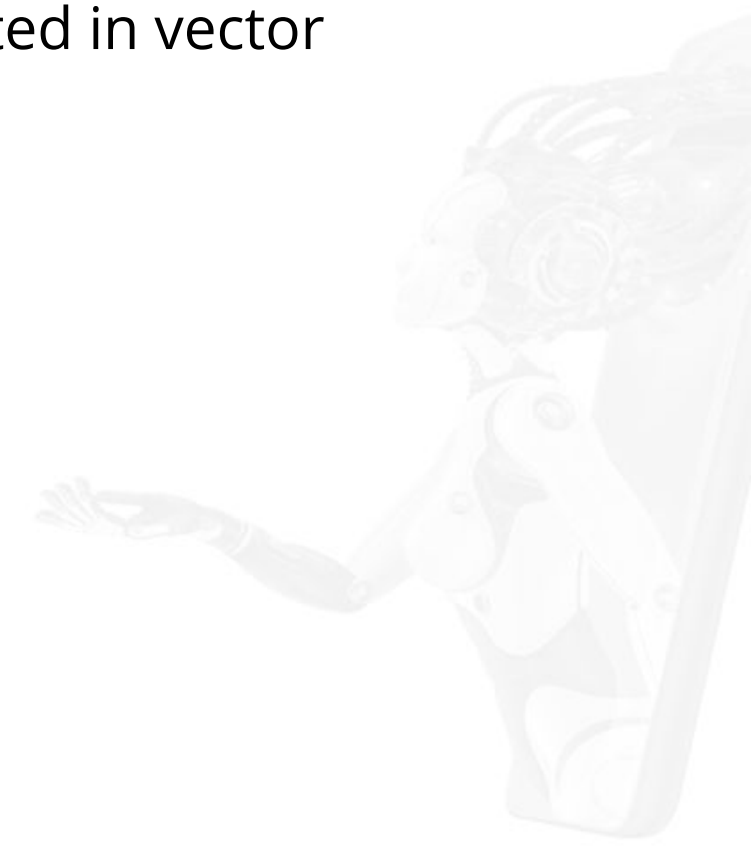
Word2vec: Advantages

Ready to be used in deep learning-ready architecture

Meaning of word is distributed in vector

Train vectors are reused

Vector size does not grow with vocabulary



Word2vec Model Creation

Problem Statement: In vector space model, the entities are transformed into vector representation. Based on the co-ordinate points, we can apply the techniques to find the most similar points in vector space. Create a word-to-vector model which gives you the similar word for happy.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

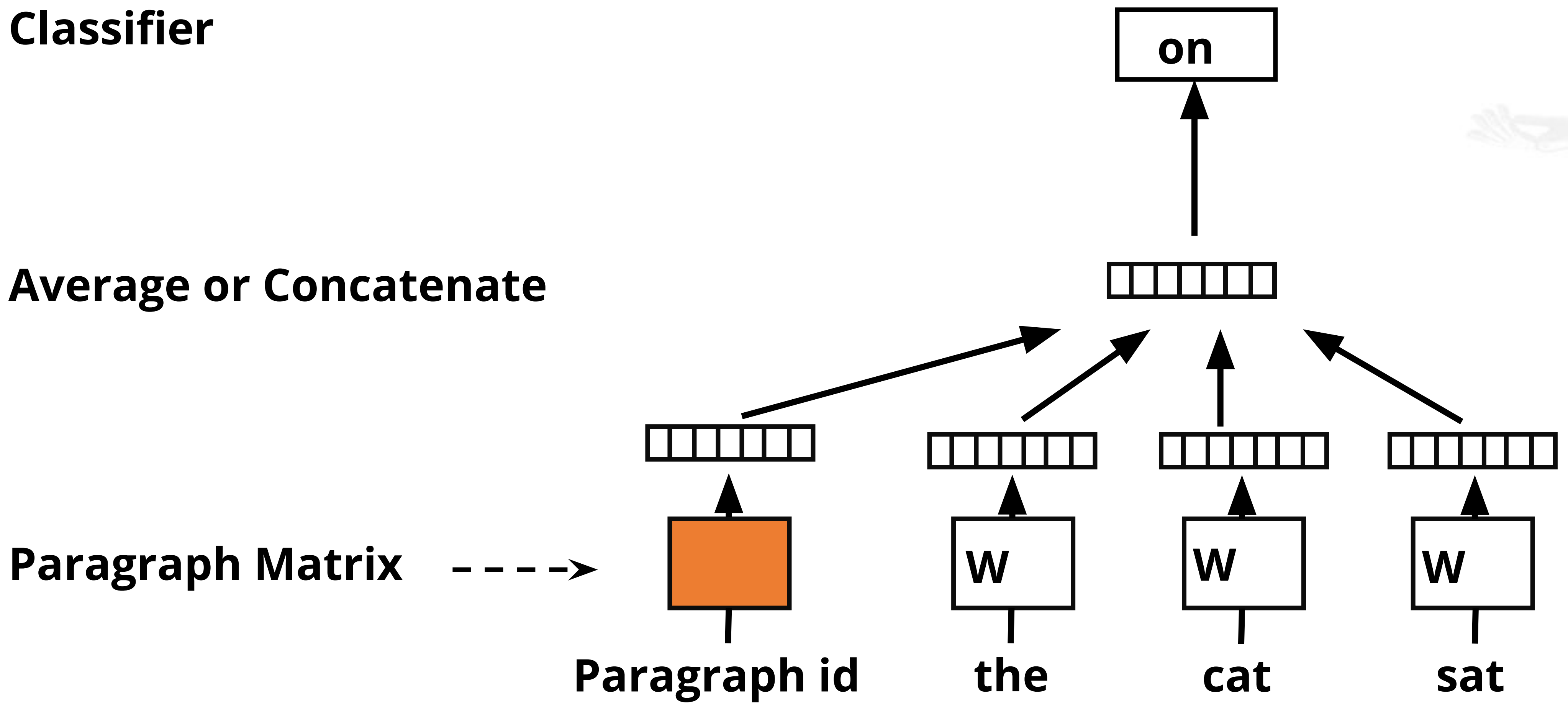
ASSISTED PRACTICE

Doc2vec Model

Doc2vec Model

The following are the uses of Doc2vec model:

- Creates numeric representation of a document
- Uses unsupervised algorithm
- Finds similarity between sentences, paragraphs, and documents



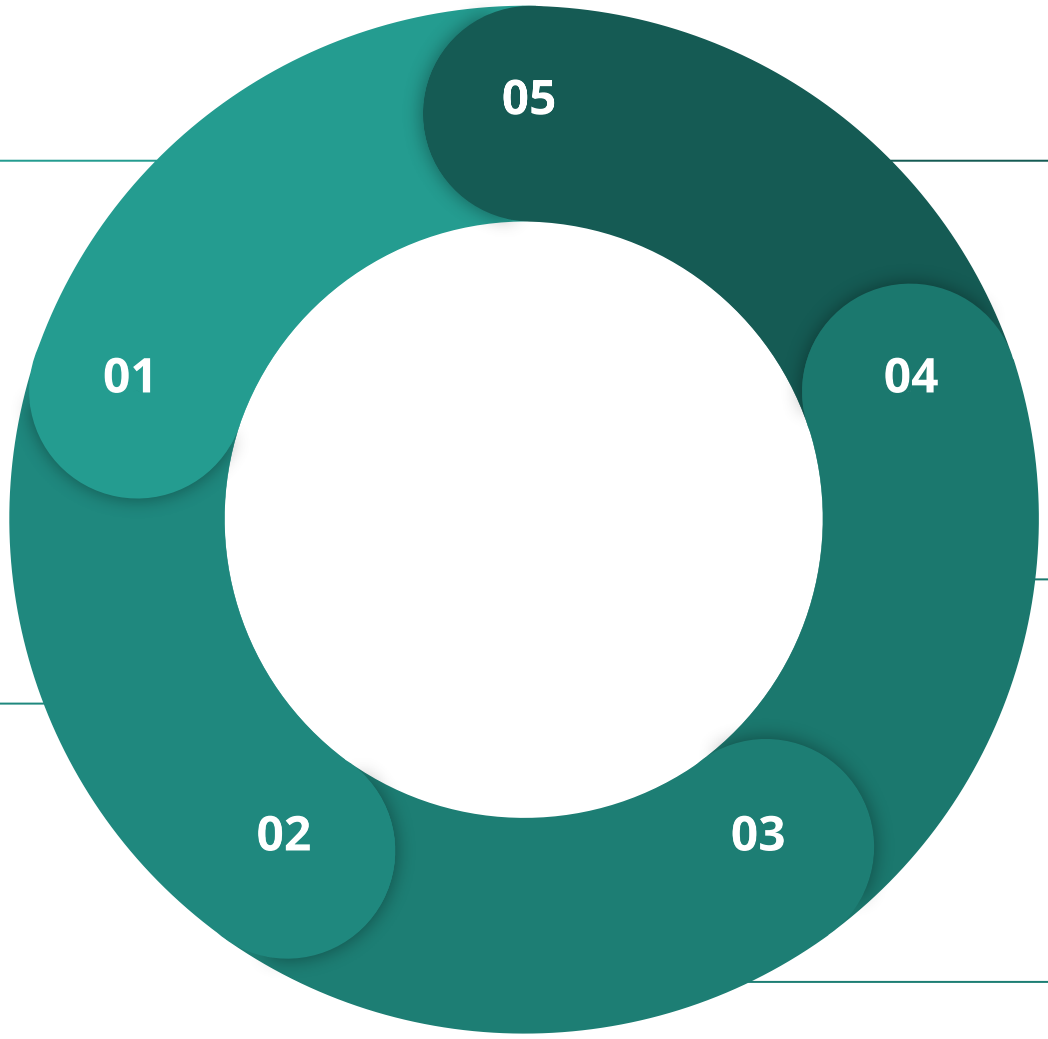
Doc2vec Model

- It is an extension of CBOW model.
- It is called distributed memory version of paragraph vector.
- This algorithm may not be the ideal choice for the corpus with lots of misspellings like tweets.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

It is a dimensionality reduction method that reduces the number of variables.



Standardization:
Standardize the range of continuous variables

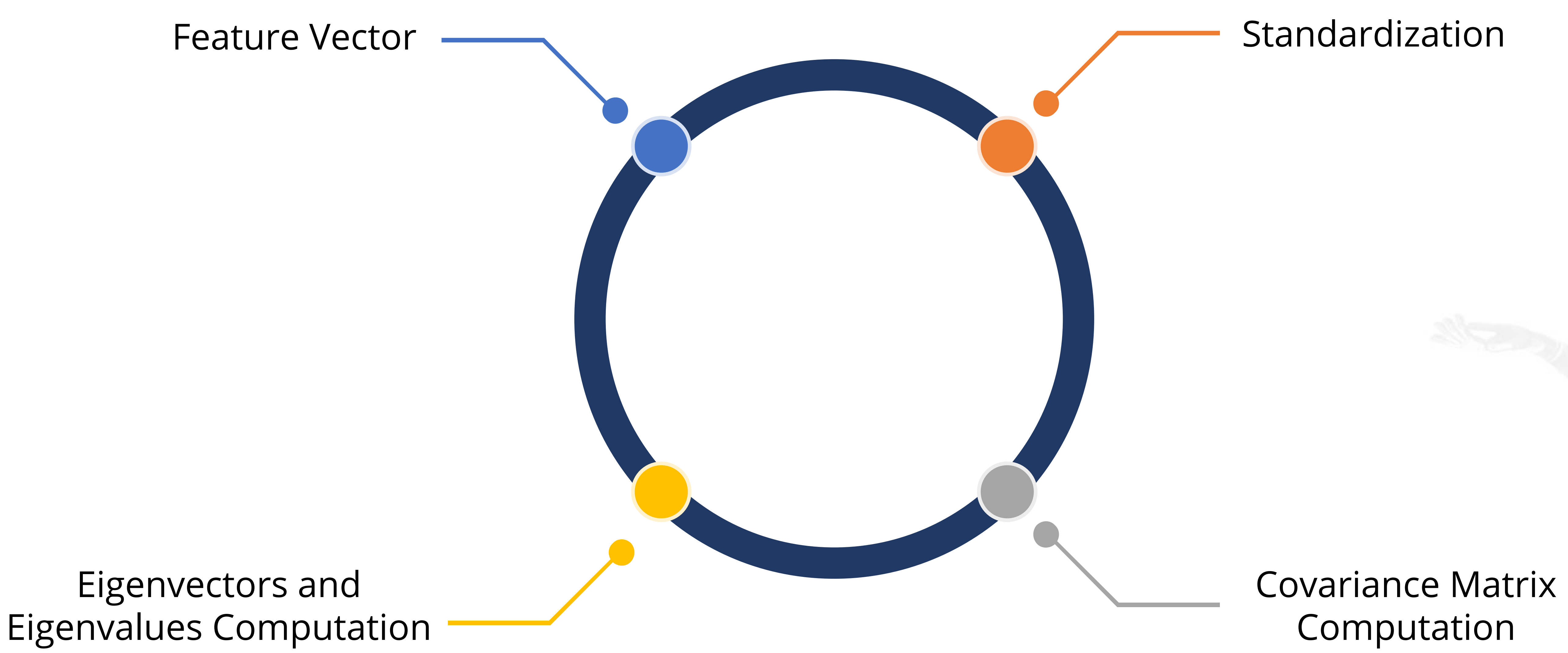
Covariance matrix computation:
Understand how variables vary from mean

Eigenvector and values computation:
Determine principal components of the data

Feature vector:
Find the principal components in the order of significance

Recasting:
Recast the data along with the principal component axes

Principal Component Analysis: Steps


Feature Vector
Standardization
Eigenvectors and Eigenvalues Computation
Covariance Matrix Computation

Step 1: Standardization

1. Standardize the range of continuous variables for their equal contribution
2. Higher range will dominate, which will create a bias
3. After standardization is done, all the variables will be on the same scale
4. It can be achieved by **z = (value - mean) / std deviation**

Step 2: Covariance Matrix Computation

1 It is used to identify the relationship between the variables

2 Variables should not be highly correlated

3 Covariance matrix (n x n) is calculated where n is number of dimensions

Step 3: Eigenvectors and Eigenvalues Computation

1. It is used to determine the principal components
2. New variables are constructed as linear combinations of initial variables and are called principal components
3. New variables will have less correlated data

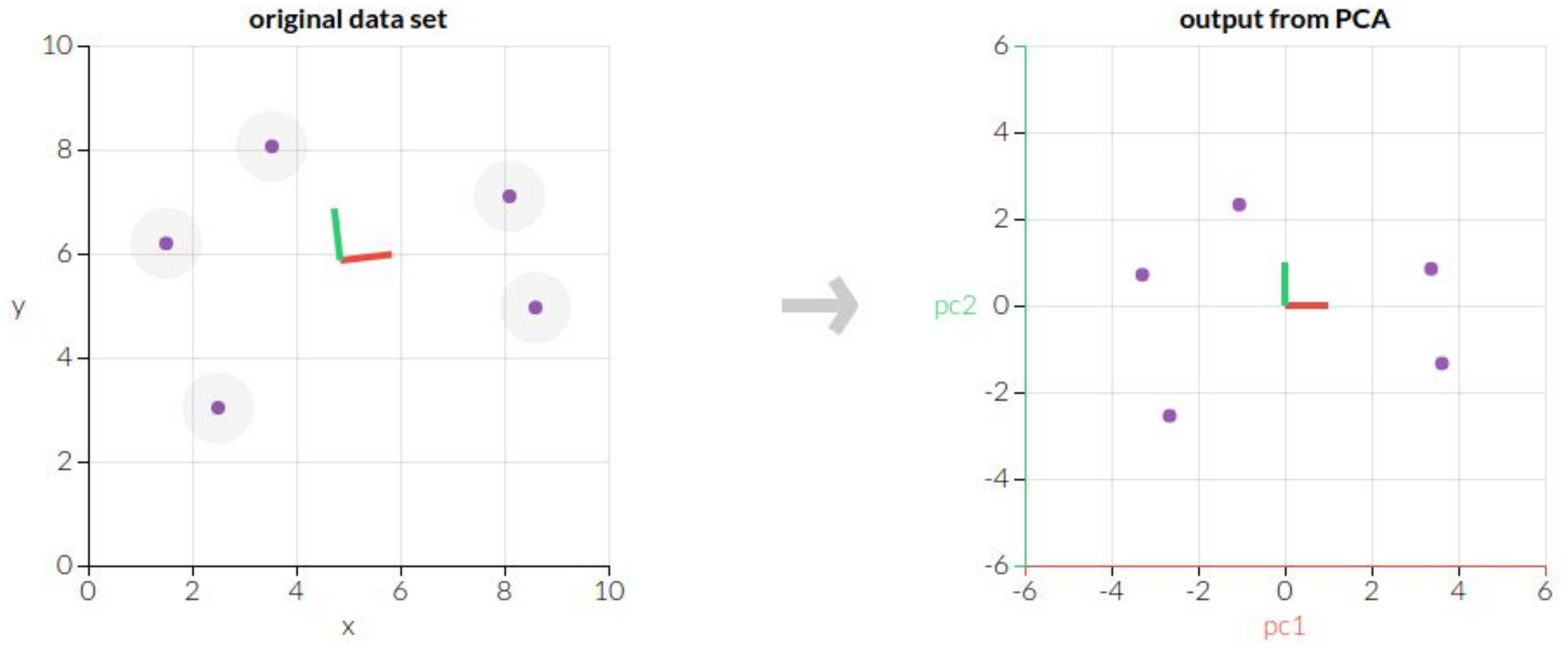
Step 4: Feature Vector

1 Decision is taken to keep all components or remove lesser significant variables

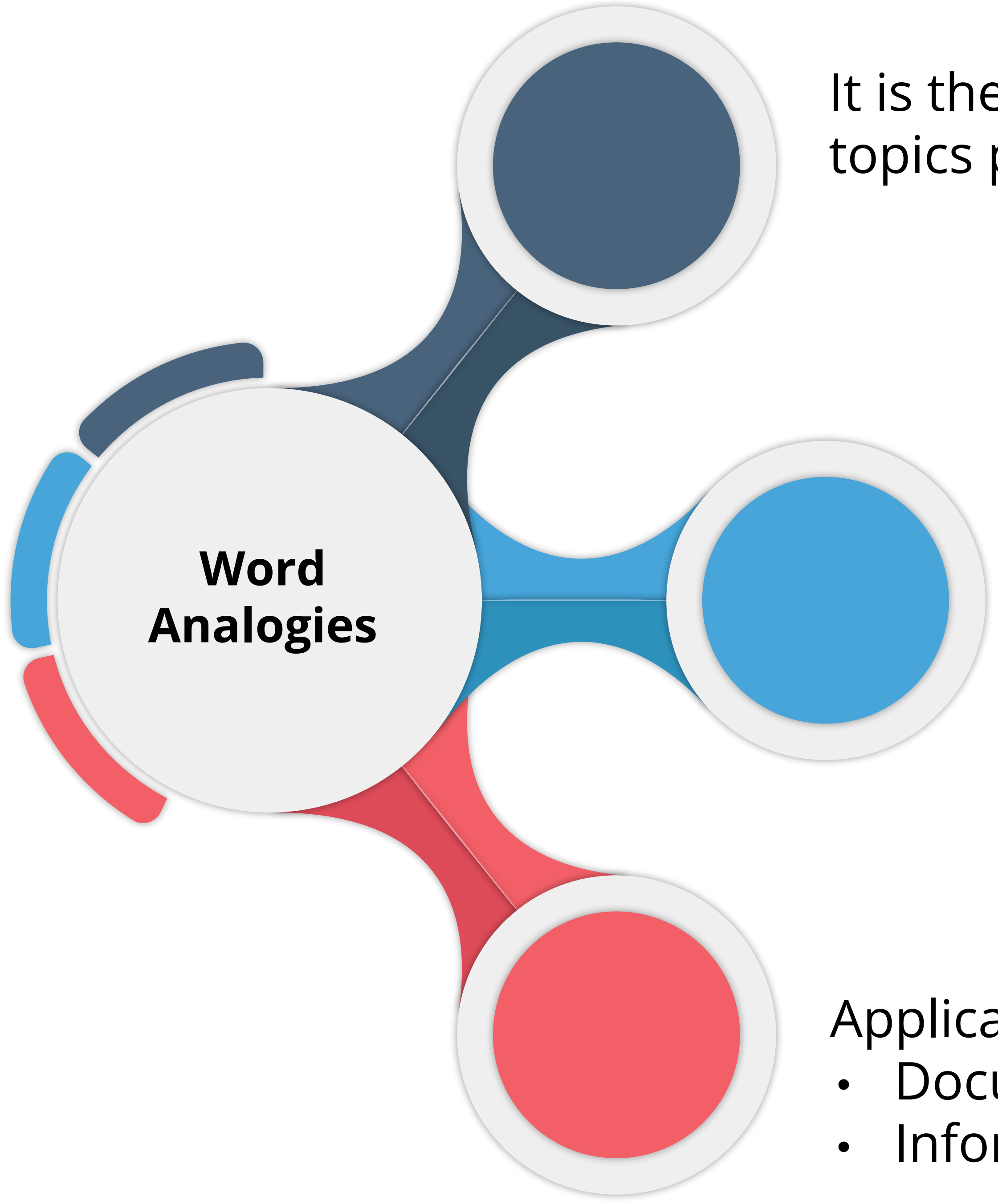
2 Remaining components will form the matrix of vectors

Principal Component Analysis

Two-dimensional data transformation after applying PCA:



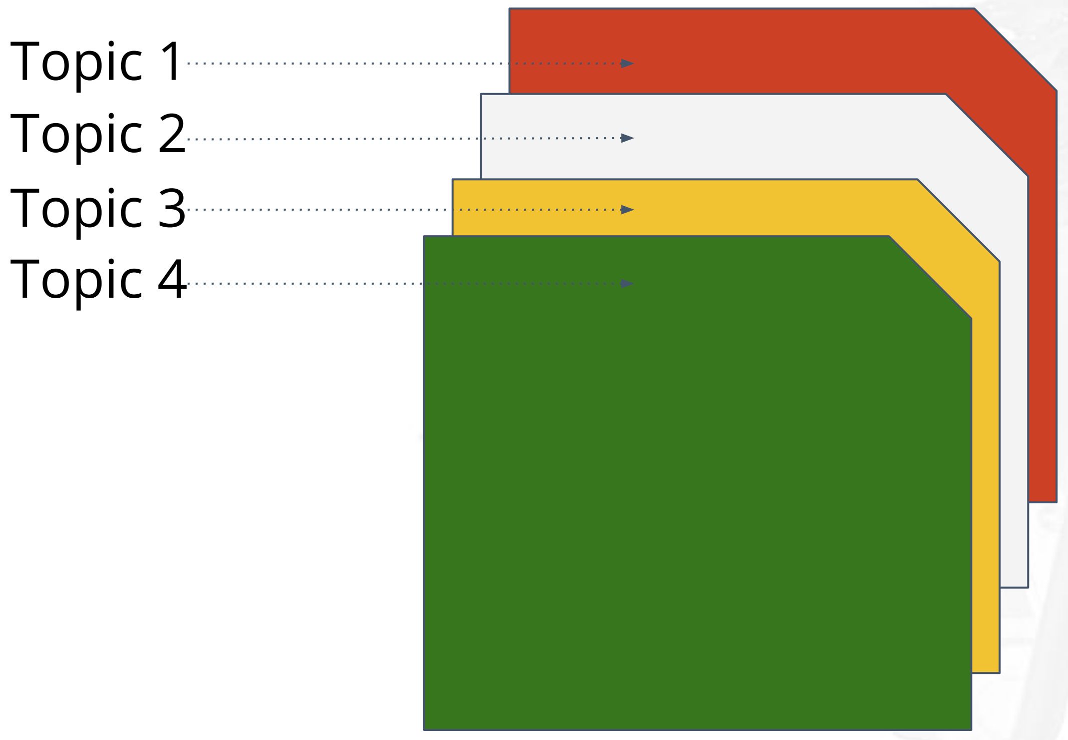
Principal Component Analysis



It is the process to automatically identify topics present in text object.

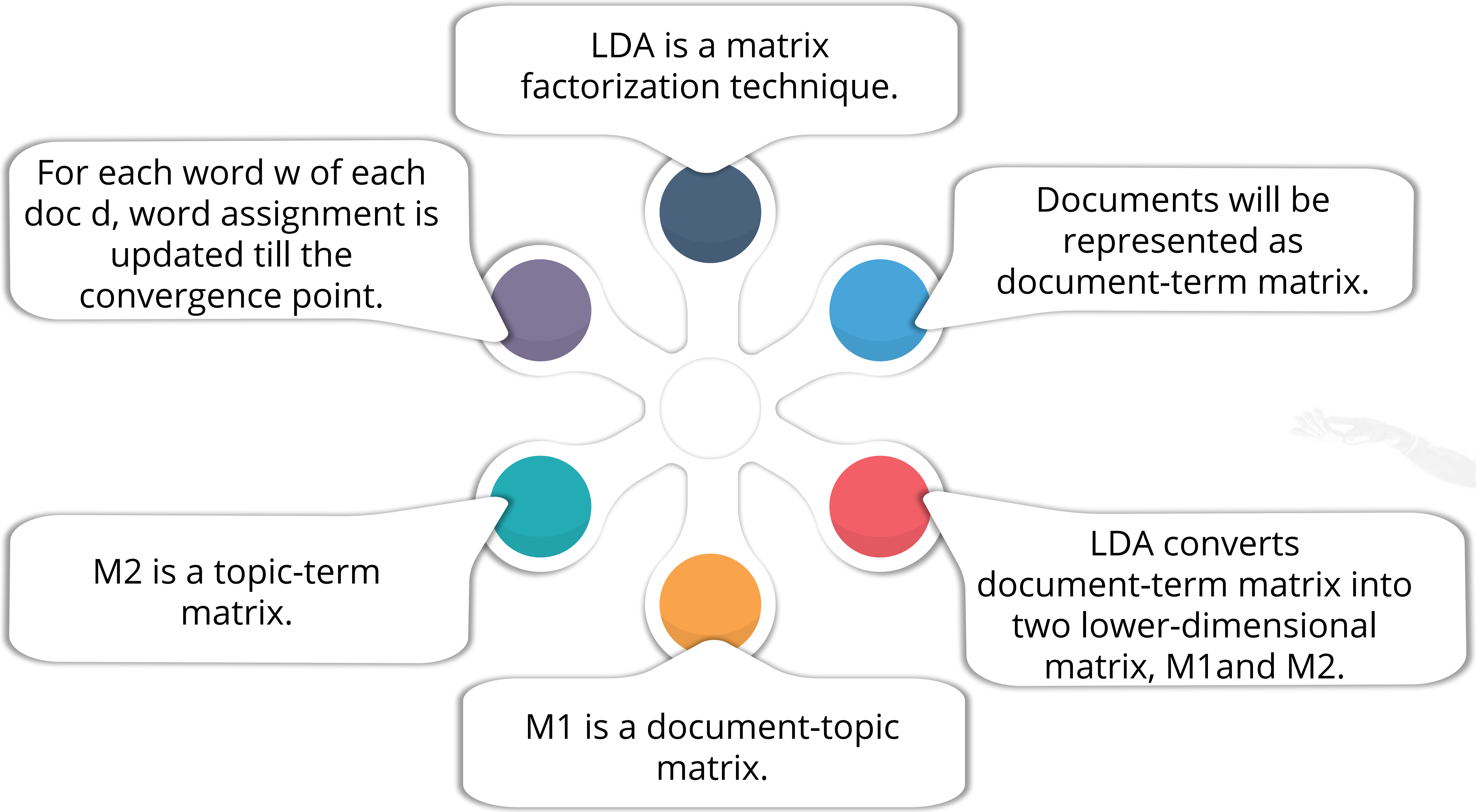
It is an unsupervised approach that involves techniques such as:
- TF-IDF
- Non-negative matrix factorization
- Latent Dirichlet Allocation
- LSA

Applications include:
- Document clustering
- Information retrieval



Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA)



LDA is a matrix factorization technique.

Documents will be represented as document-term matrix.

LDA converts document-term matrix into two lower-dimensional matrix, M1and M2.

M1 is a document-topic matrix.

M2 is a topic-term matrix.

For each word w of each doc d, word assignment is updated till the convergence point.

Latent Dirichlet Allocation: Example

Bag of Word Model

Term/Word

			daughter	lamb	littl	mari	star	silenc	twinkl
D1		I have a little daughter	1	0	1	0	0	0	0
D2		Mary had a little lamb	0	1	1	1	0	0	0
D3		Twinkle Twinkle little star	0	0	1	0	1	0	2
Dn		The silence of lambs	0	1	0	0	0	1	0

Document Term Matrix

For D3 P(t|d)=1/4 ¼ 2/4

Corpus (D) : Set of Documents

Probability of word occurring in Document

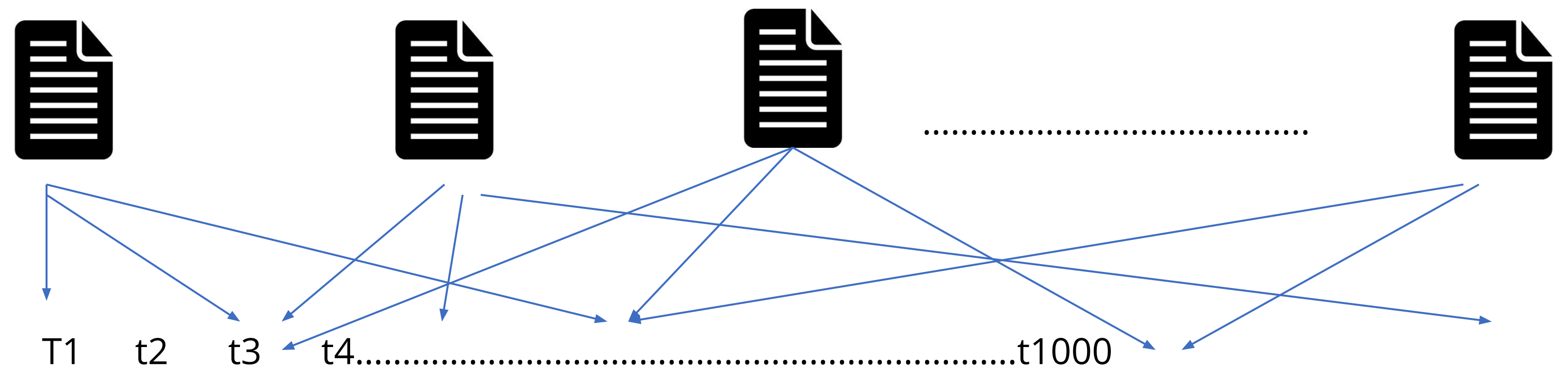
No. of parameters: 3

For 1 document and 3 words, number of parameters are = 1*3=3

Latent Dirichlet Allocation: Example

1000 documents(d)

5000 terms/words (t)



Parameters P(t|d)

For 1000 documents and 5000 words, number of parameters are = 1000*5000=5000000 (50 Lakhs)

Problem:
There are so many parameters to extract information and so, the task is to reduce number of parameters without losing information

Latent Dirichlet Allocation: Example

Solution:
Introduce a layer of topics called the Latent Variable

Topic is a mix of terms that is likely to generate the term.
Example: Finance, Science, Sport, etc.

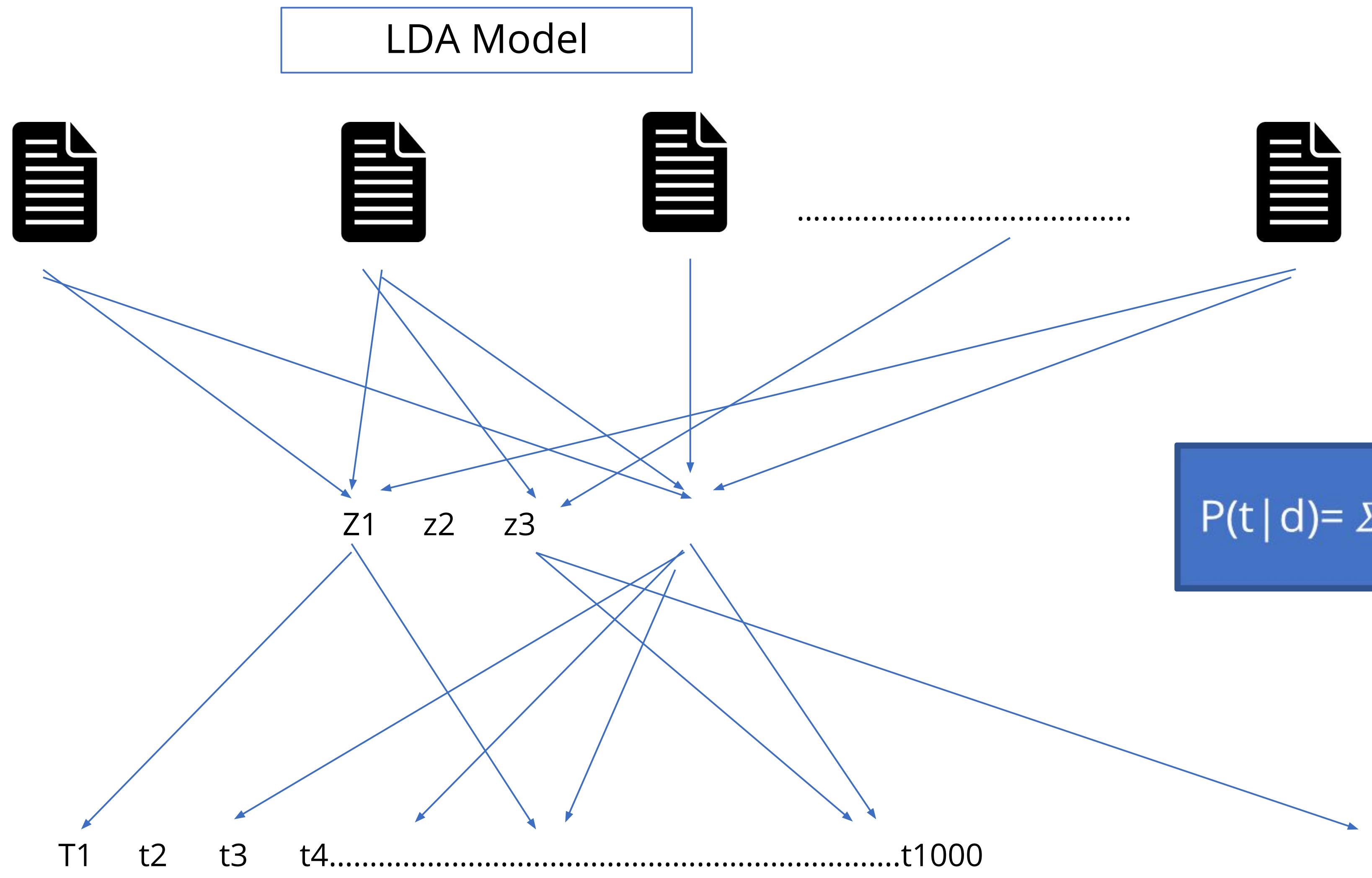
1000 documents(d)

P(z|d) → Probability of topic z given document d

Topics/Latent Variable (z)

P(t|z) → Probability of term t given topic z

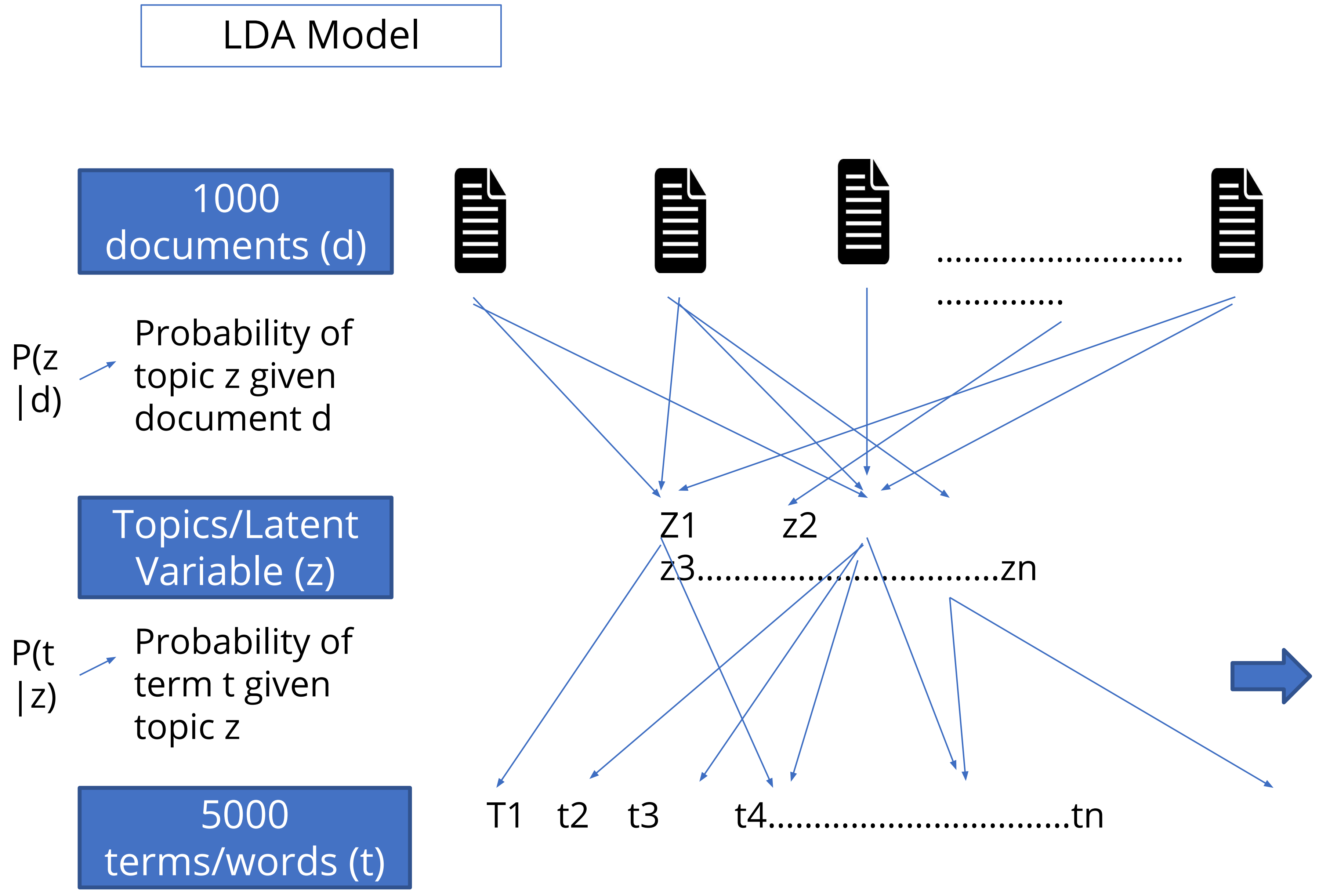
5000 Terms/Words (t)



P(t|d)= ΣP(z|d)P(z|d)

For 1000 documents, 5000 words, 10 topics, the number of parameters are = 1000*10+10*5000=60000

Latent Dirichlet Allocation: Example


LDA Model
1000 documents (d)
....
....
P(z |d)
Probability of topic z given document d
Topics/Latent Variable (z)
Z1 z2 z3..................................zn
P(t |z)
Probability of term t given topic z
5000 terms/words (t)
T1 t2 t3 t4.................................tn

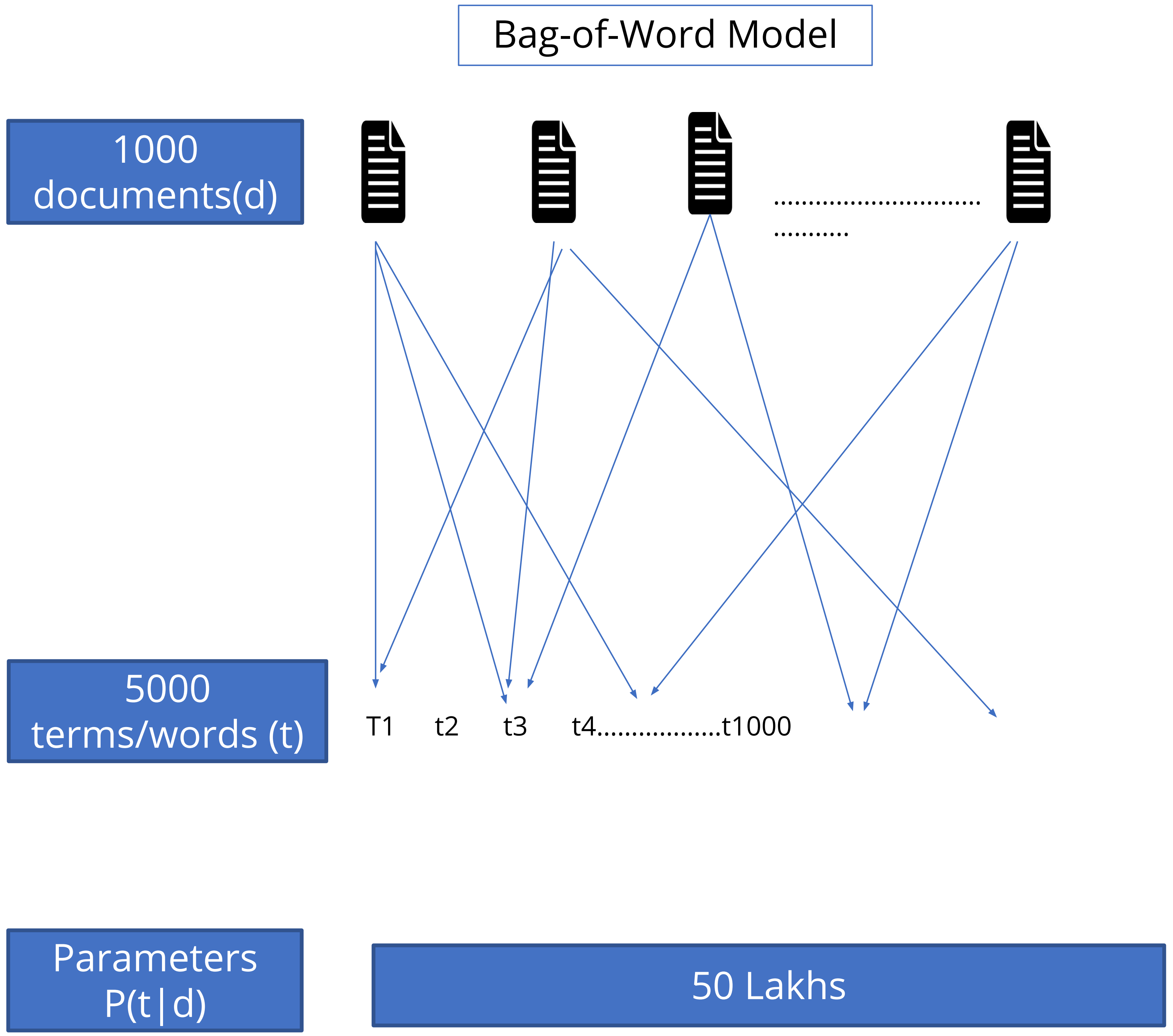
M1

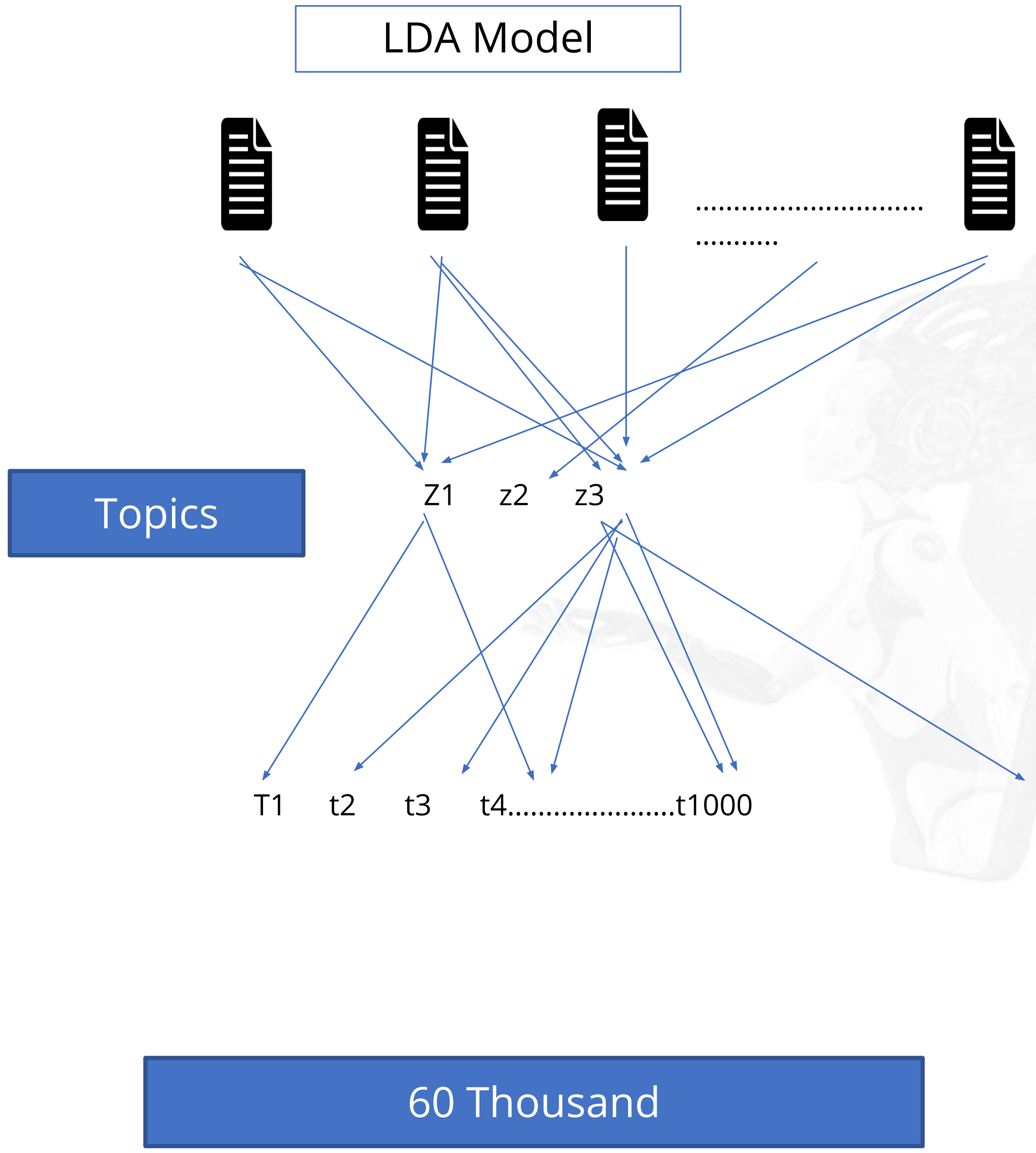
	Z1	Z2	..	zn
d1				
d2				
d3				
d4				
...				
dn				

M2

	t1	t2	t3	t4	t5	....	tn
z1							
z2							
..							
zn							

Latent Dirichlet Allocation: Example





Topic Modeling

Topic Modeling

It is a type of statistical model and has the following advantages:

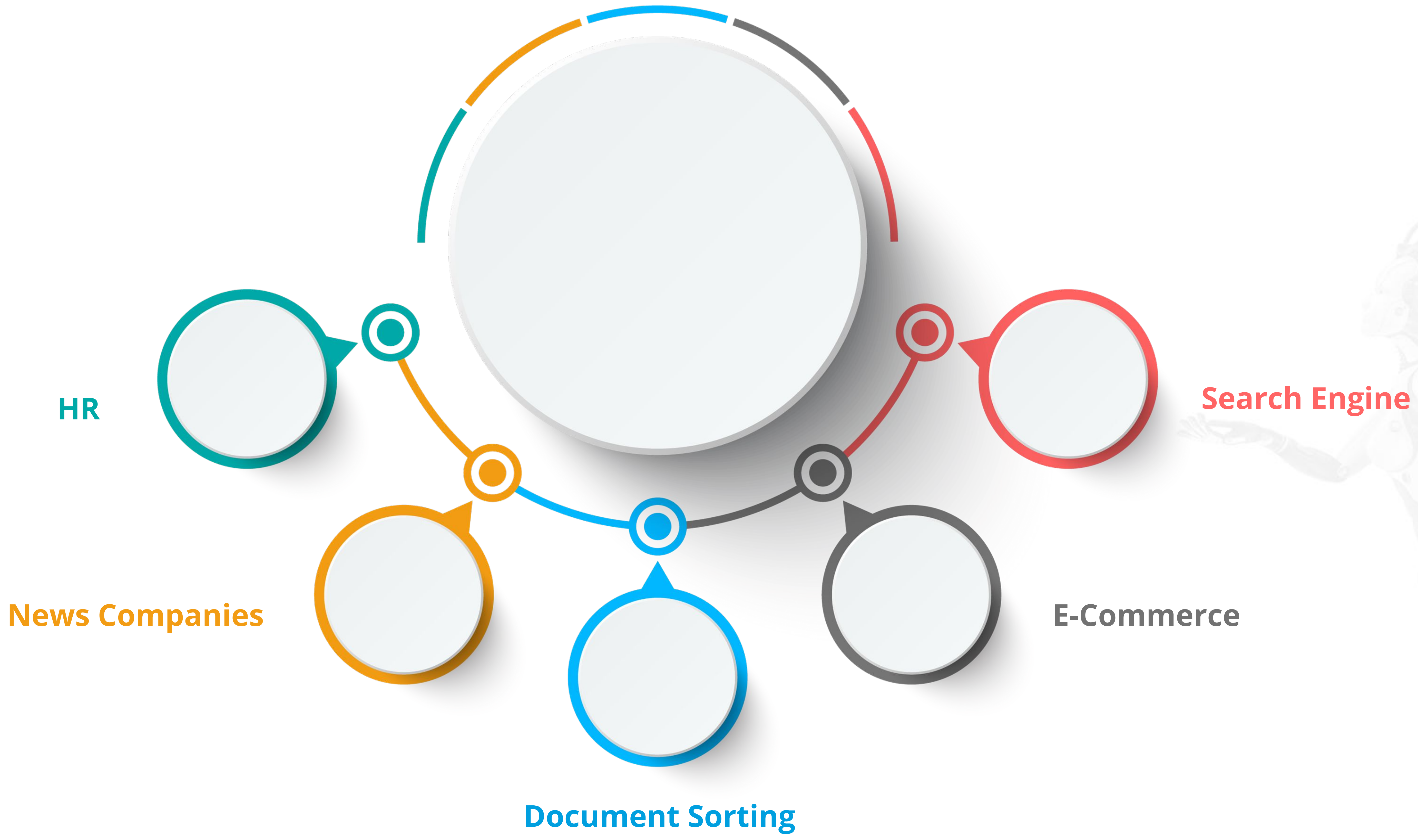
Discovering the abstract topics in a collection of documents

Document clustering

Information retrieval from unstructured text and feature selection

Organizing large blocks of textual data

Topic Modeling: Industry Use Cases



Gensim

Gensim: Introduction

1 Gensim is a free python library which is platform-independent.

2 It is open-source.

3 It is robust and scalable.

4 It analyzes plain-text documents for semantic structure.

5 It is used to retrieve semantically similar documents.

Gensim: Syntax and Library

System Requirement:

Operating system:
macOS / OS X · Linux · Windows

Python version:
Python >=2.7

Dependency:
- NumPy >= 1.11.3
- SciPy >= 0.18.1
- Six >= 1.5.0
- smart_open >= 1.2.1

```
>> import gensim
```

Gensim: Vectorization

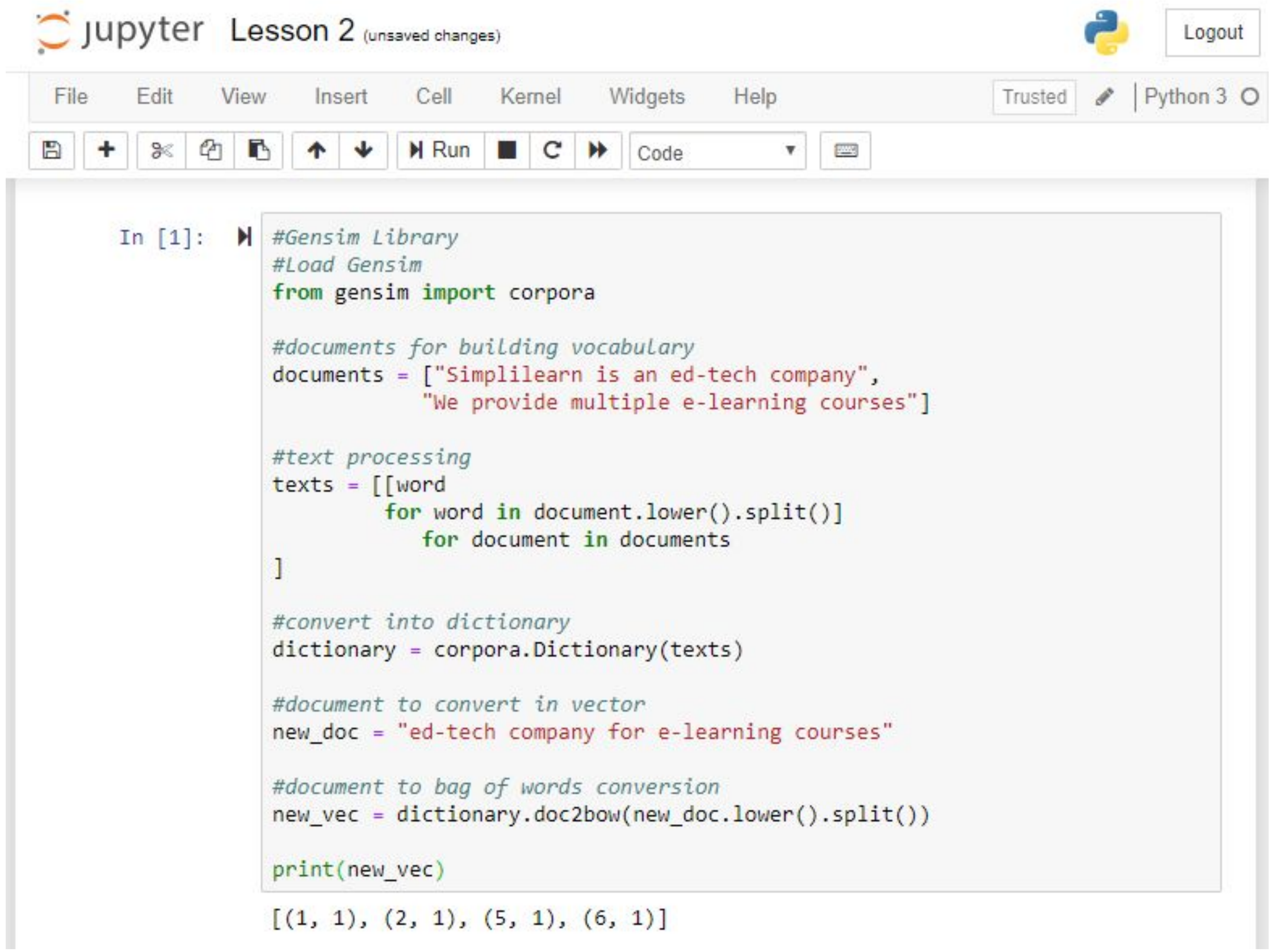
```
#Gensim Library
#Load Gensim
from gensim import corpora

#documents for building vocabulary
documents = ["Simplilearn is an ed-tech company",
            "We provide multiple e-learning courses"]

#text processing
texts = [[word
         for word in document.lower().split()]
            for document in documents
]
#convert into dictionary
dictionary = corpora.Dictionary(texts)
#document to convert in vector
new_doc = "ed-tech company for e-learning courses"
#document to bag of words conversion
new_vec = dictionary.doc2bow(new_doc.lower().split())
print(new_vec)
```

Gensim: Vectorization

Output: [(1, 1), (2, 1), (5, 1), (6, 1)]


jupyter Lesson 2 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [1]:

```
#Gensim Library
#Load Gensim
from gensim import corpora

#documents for building vocabulary
documents = ["Simplilearn is an ed-tech company",
            "We provide multiple e-learning courses"]

#text processing
texts = [[word
         for word in document.lower().split()]
            for document in documents
]

#convert into dictionary
dictionary = corpora.Dictionary(texts)

#document to convert in vector
new_doc = "ed-tech company for e-learning courses"

#document to bag of words conversion
new_vec = dictionary.doc2bow(new_doc.lower().split())

print(new_vec)
```

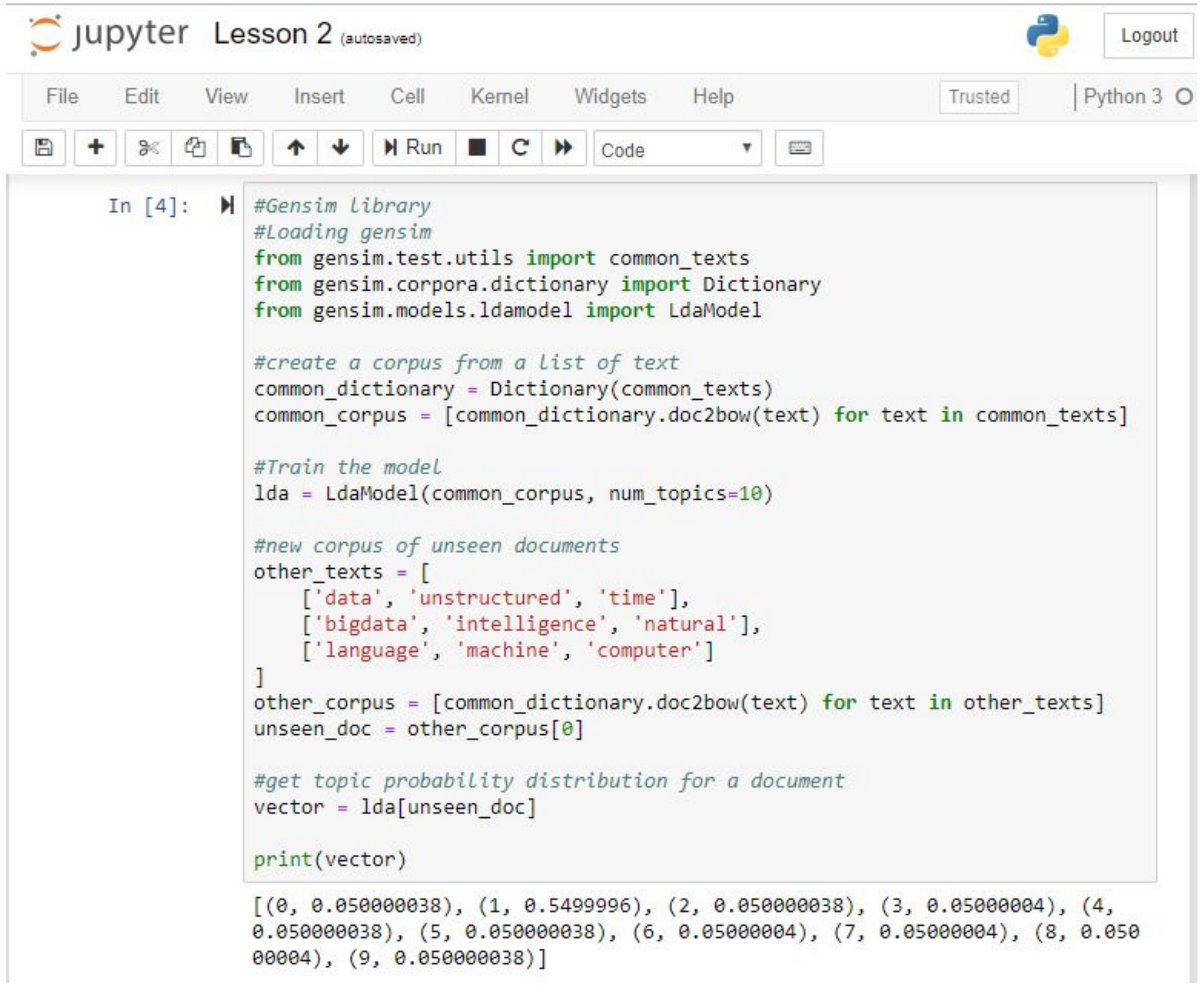
```
[(1, 1), (2, 1), (5, 1), (6, 1)]
```

Gensim: Topic Modeling

```
#Gensim library
#Loading gensim
from gensim.test.utils import common_texts
from gensim.corpora.dictionary import Dictionary
from gensim.models.ldamodel import LdaModel
#create a corpus from a list of text
common_dictionary = Dictionary(common_texts)
common_corpus = [common_dictionary.doc2bow(text) for text in common_texts]
#Train the model
lda = LdaModel(common_corpus, num_topics=10)
#new corpus of unseen documents
other_texts = [
    ['data', 'unstructured', 'time'],
    ['bigdata', 'intelligence', 'natural'],
    ['language', 'machine', 'computer']
]
other_corpus = [common_dictionary.doc2bow(text) for text in other_texts]
unseen_doc = other_corpus[0]
#get topic probability distribution for a document
vector = lda[unseen_doc]
print(vector)
```

Gensim: Topic Modeling

Output:
[(0, 0.050000038), (1, 0.5499996), (2, 0.050000038), (3, 0.05000004), (4, 0.050000038), (5, 0.050000038), (6, 0.05000004), (7, 0.05000004), (8, 0.05000004), (9, 0.050000038)]


jupyter Lesson 2 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

In [4]:
```
#Gensim library
#Loading gensim
from gensim.test.utils import common_texts
from gensim.corpora.dictionary import Dictionary
from gensim.models.ldamodel import LdaModel

#create a corpus from a list of text
common_dictionary = Dictionary(common_texts)
common_corpus = [common_dictionary.doc2bow(text) for text in common_texts]

#Train the model
lda = LdaModel(common_corpus, num_topics=10)

#new corpus of unseen documents
other_texts = [
    ['data', 'unstructured', 'time'],
    ['bigdata', 'intelligence', 'natural'],
    ['language', 'machine', 'computer']
]
other_corpus = [common_dictionary.doc2bow(text) for text in other_texts]
unseen_doc = other_corpus[0]

#get topic probability distribution for a document
vector = lda[unseen_doc]

print(vector)
```

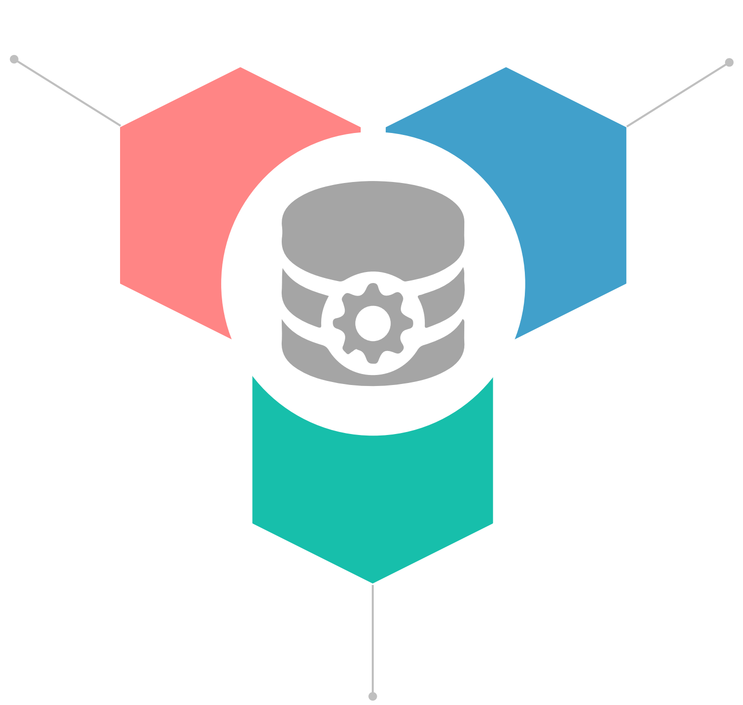
```
[(0, 0.050000038), (1, 0.5499996), (2, 0.050000038), (3, 0.05000004), (4,
0.050000038), (5, 0.050000038), (6, 0.05000004), (7, 0.05000004), (8, 0.050
00004), (9, 0.050000038)]
```

DATA AND
ARTIFICIAL INTELLIGENCE

Word Embedding

Word Embedding

Use the following while working with individual words or phrases:



Text Generation

Machine Translation

Large Vocabulary

Word Embedding

It represents text in the N-dimensional space, in the form of vectors

Vectors are called embeddings

It is the distributed representation

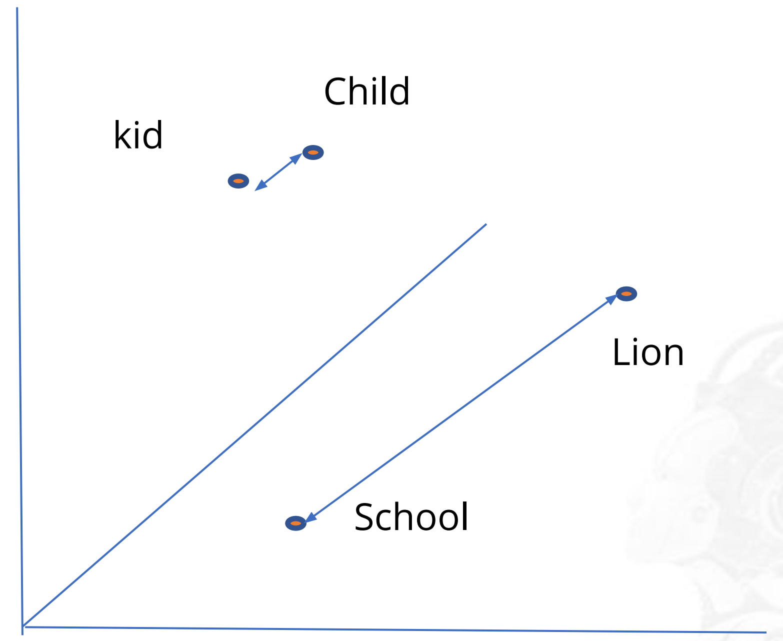
Each word is mapped to one real-valued vector

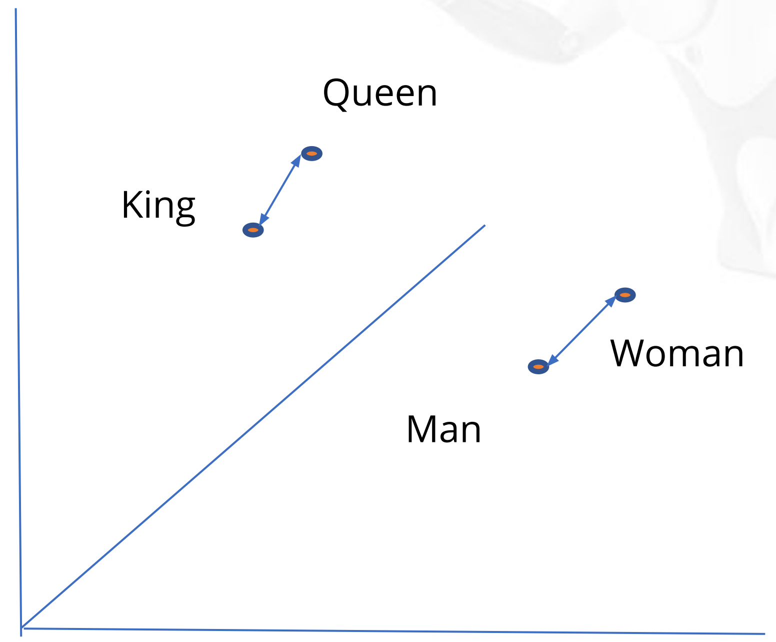
Word embedding techniques:
- Word2vec
- Glove

Applications of word embedding:
- Music or video recommendation system
- Analyzing survey responses

Word Embedding: Overview

- Word embedding represents word in vector form
- Some properties must be exhibited while representing a word in vector form:
  - Similar meaning words should be closer to each other when compared to the words which don't have similar meaning
  - Words having difference in meaning should be kept at the same distance from each other
- This kind of representation helps in finding:
  - Analogy word
  - Synonym
  - Classification of the word: Positive, negative or neutral





Identify Topics from News Items

Problem Statement: Identification of document for a domain or keyword is a tough task. Write a script which will provide the important topics from the news data.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Working of Word Analogies

Problem Statement: Apply word analogies technique using word2vec for identification of new next word.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

ASSISTED PRACTICE

Build Your Own News Search Engine

Objective: Use text feature engineering (TF-IDF) and some rules to make our first search engine for news articles. For any input query, we'll present the five most relevant news articles.

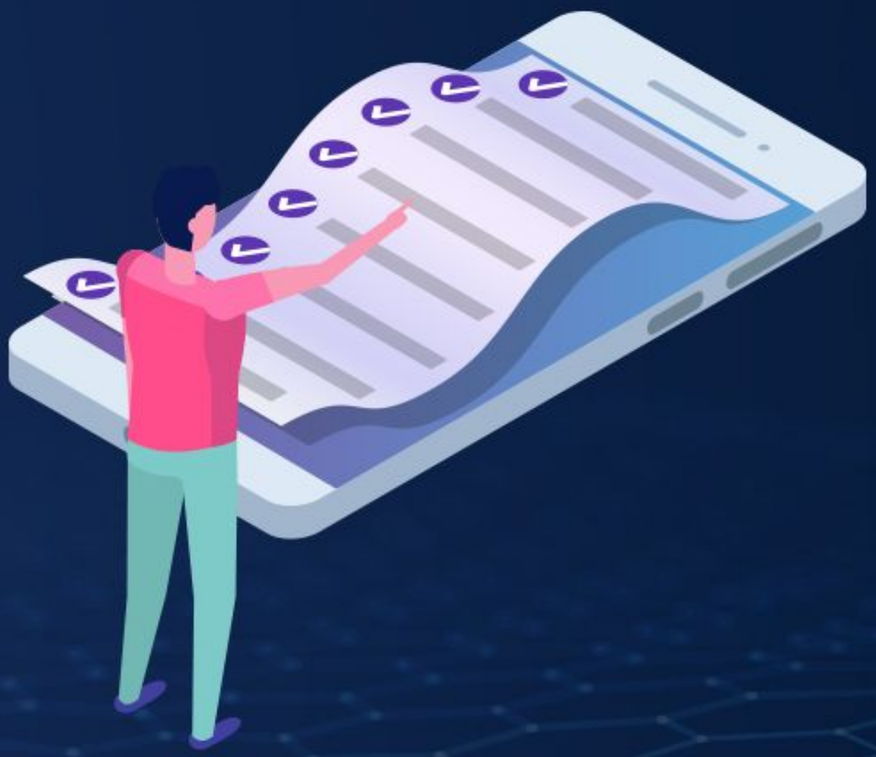
Problem Statement: Reuters Ltd. is an international news agency headquartered in London and is a division of Thomson Reuters. The data was originally collected and labeled by Carnegie Group Inc. and Reuters Ltd. in the course of developing the construe text categorization system. An important step before assessing similarity between documents, or between documents and a search query, is the right representation i.e., correct feature engineering. We'll make a process that provides the most similar news articles to a given text string (search query).

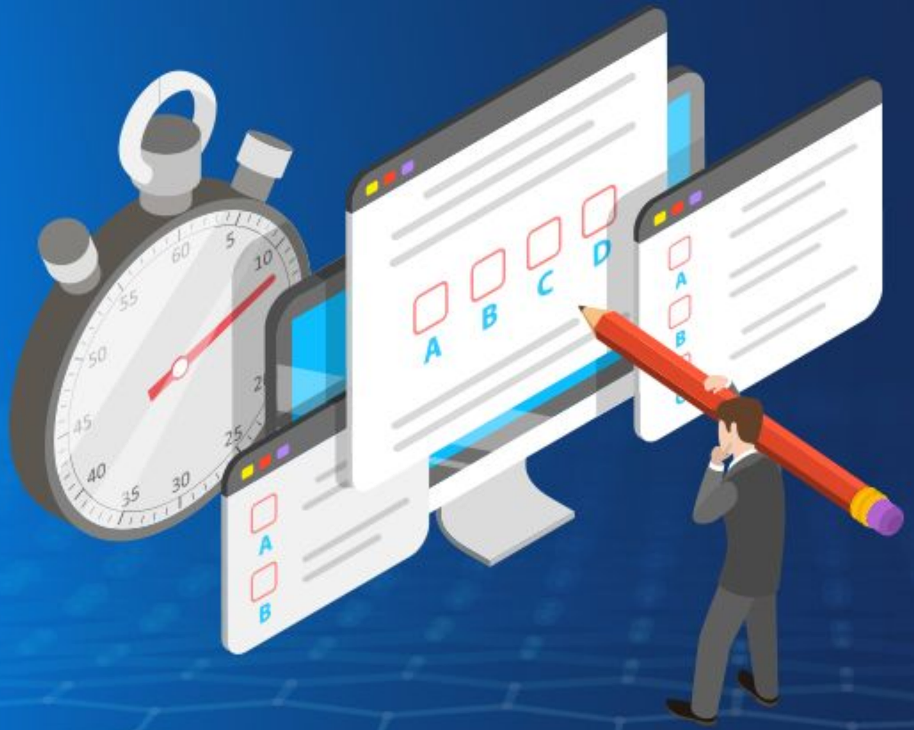
LESSON-END PROJECT

Key Takeaways

You are now able to:

- Explain N-gram
- Demonstrate the different word embedding models
- Perform operations on word analogies
- Demonstrate the working of Bag-of-Words
- Demonstrate the working of top modeling technique





Knowledge Check

Knowledge Check

1

How many bigrams can be generated from the given sentence?
"Simplilearn is a great source to learn machine learning"

a. 7

b. 8

c. 9

d. 10

Knowledge Check

1

How many bigrams can be generated from given sentence?
“Simplilearn is a great source to learn machine learning”

a. 7

b. 8

c. 9

d. 10

The correct answer is **b**

Bigrams: Simplilearn is, is a, a great, great source, source to, to learn, learn machine, machine learning

Knowledge Check

2

The main advantages of document-term matrix are:

a. Feature engineering

b. Understanding the frequency of word

c. Converting text into vectors

d. All of the above

Knowledge Check

2

The main advantages of document-term matrix are:

a. Feature engineering

b. Understanding the frequency of word

c. Converting text into vectors

d. All of the above

The correct answer is **d**

Document-term matrix converts sentences into vectors, and it is achieved by creating matrix of unique words of sentences.

Knowledge Check

3

Highest distance in the Levenshtein approach depicts:

a. More similar words

b. More dissimilar words

c. Cannot decide the distance

d. Depends on the length of words

Knowledge Check

3

Highest distance in the Levenshtein approach depicts:

a. More similar words

b. More dissimilar words

c. Cannot decide the distance

d. Depends on the length of words

The correct answer is **b**

Highest distance in the Levenshtein approach depicts more dissimilar words.

Knowledge Check

4

What is the purpose of topic modeling?

a. Clustering the documents

b. Converting text into vectors

c. Understanding the frequency of word

d. Vectorization

Knowledge Check

4

What is the purpose of topic modeling?

a. Clustering the documents

b. Converting text into vectors

c. Understanding the frequency of word

d. Vectorization

The correct answer is **a**

Topic modeling provides the topic which is used to map the documents.

Knowledge Check

5

Which techniques are used to find the similarity between text?

a. Cosine, Levenshtein, Document-Term Matrix

b. Cosine, Word2vec, Document-Term Matrix

c. POS, Document-Term Matrix, Levenshtein

d. Cosine, Levenshtein, Word2vec, POS

Knowledge Check

5

Which techniques are used to find the similarity between text?

a. Cosine, Levenshtein, Document-Term Matrix

b. Cosine, Word2vec, Document-Term Matrix

c. POS, Document-Term Matrix, Levenshtein

d. Cosine, Levenshtein, Word2vec, POS

The correct answer is **d**

Cosine, Levenshtein, Word2vec, and POS are the techniques used to find the similarity between text.

Thank You