Text Mining and Natural Language Processing

Unit 4: Session 02

By Md. Meftaul Haque Mishu

Objectives

- Understand the basics of Text Mining
- Know popular Text Mining techniques
- Understand the basics of Natural Language Processing (NLP)
- Apply NLP techniques for sentiment analysis

Text Mining

What is Text Mining?

- Process of transforming unstructured text data into structured format.
- Extracting meaningful information from text.

Why Text Mining?

- Extract customer feedback from millions of reviews.
- Extract key information from legal documents.
- Understand political sentiments from social media posts.
- Detect spam and fraud.
- Extract key information from resumes.

Types of Data

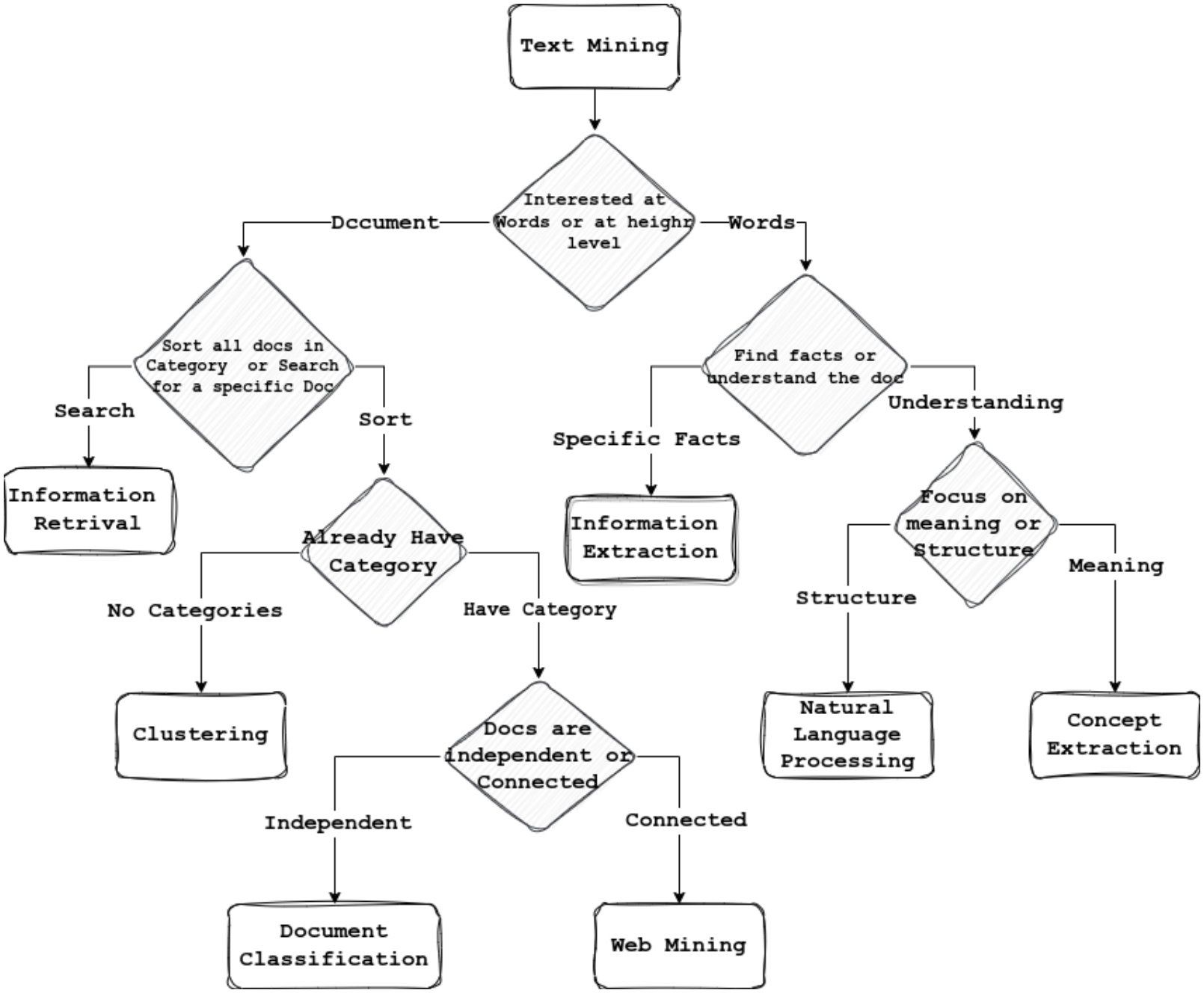
Depending on the database, text data can be organized as follows:

- **Unstructured data**: Data does not have a predefined data format. It can include text from sources, like social media or product reviews, or rich media formats like, video and audio files.
- **Semi-structured data**: Data that is organized in a pre-defined manner but does not conform to the structure of a relational database, such as XML, CSV, JSON files.
- **Structured data**: Text data that is organized in a pre-defined manner and conforms to the structure of a relational database, such as a table in a relational database.

Common Text Mining Techniques

- **Sentiment Analysis**: Extracting the sentiment of a text document.
- **Text Categorization**: Assigning a category to a text document.
- **Named Entity Recognition**: Identifying and classifying named entities in a text document.
- **Topic Modeling**: Identifying topics in a text document.
- **Text Summarization**: Generating a summary of a text document.
- **Text Clustering**: Grouping similar text documents together.

Where it fits?



Before applying any text mining technique, we must start with pre-processing the text data. This practice generally known as Natural Language Processing (NLP).

So, let's start with NLP !

Natural Language Processing

(NLP)

What is NLP?

Giving computers the ability to understand text and spoken words as humans can.

NLP combines computational linguistics— rule-based modeling of human language— with statistical, machine learning, and deep learning models.

NLP Challenges

- **Lexical Ambiguity**: Words have multiple meanings.
- **Syntactic Ambiguity**: Sentences have multiple meanings.
- **Referential Ambiguity**: Pronouns can refer to multiple entities.

Natural Language Understanding (NLU), Natural Language Generation (NLG)

Unstructured Text

Bring groceries for dinner. I would like to have mutton biryani and something sweet to have for the desert.

→ NLU →

(Natural Language Understanding)

--

← NLG ←

(Natural Language Generation)

Structured Text

- Rice
- Biryani masala
- Mutton
- Sweet

(XML, JSON, CSV, Tabular Data)

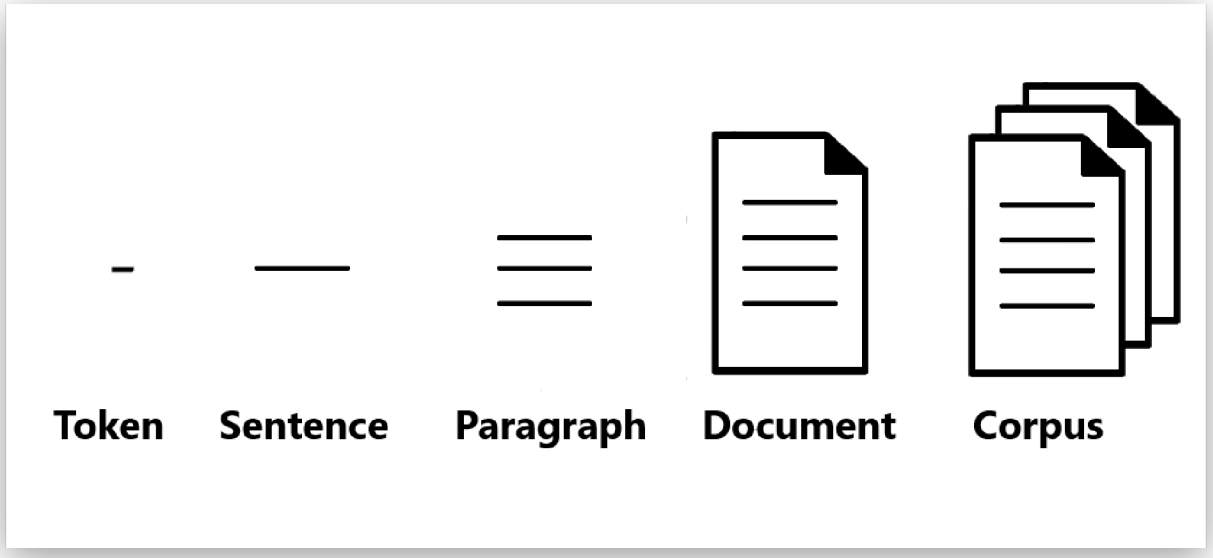
Tools for NLP

- **NLTK**: Natural Language Toolkit
- **SpaCy**: Industrial-Strength Natural Language Processing
- **Gensim**: Topic Modeling for Humans
- **Flair**: A very simple framework for state-of-the-art NLP.

Key NLP Concepts

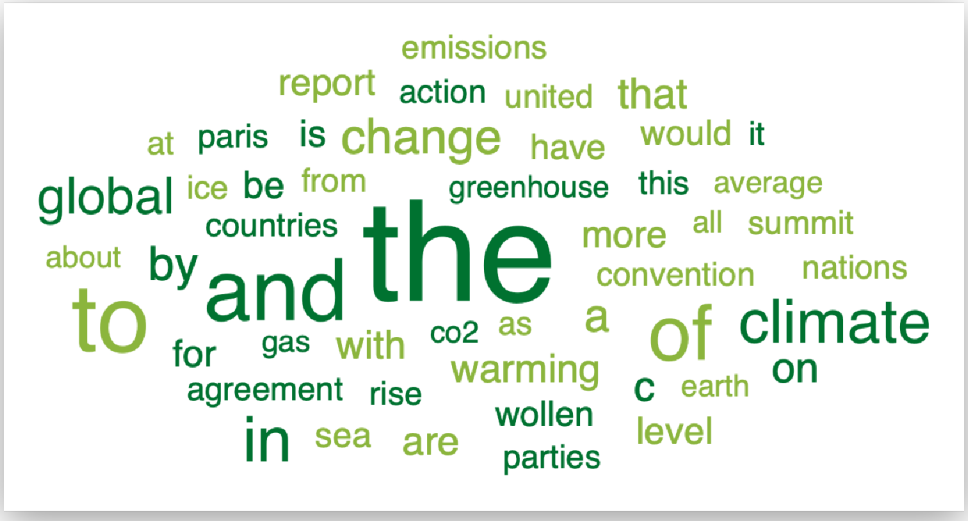
Corpus

- Corpus refers to a large collection of unstructured texts.
- Used for statistical analysis and hypothesis testing.
- The plural of corpus is corpora.



Stop Words

- Commonly used words in a language.
- Stop words are removed from the text before processing.
- Example: a, an, the, is, at, which, on, for, this, that, etc.

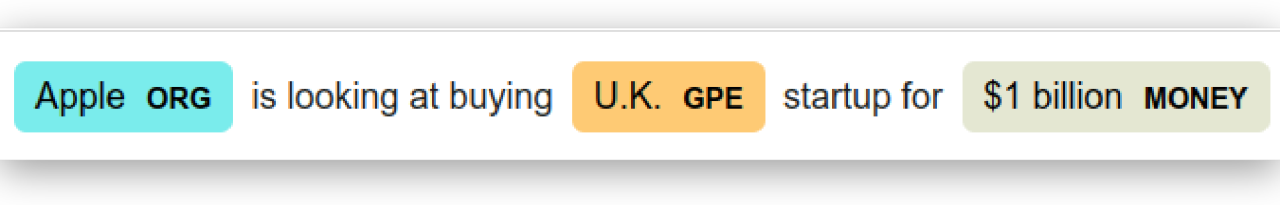


Part-of-Speech (POS) Tagging

- Assigning a part-of-speech tag to each word in a sentence.
- Based on the word's definition and its context.

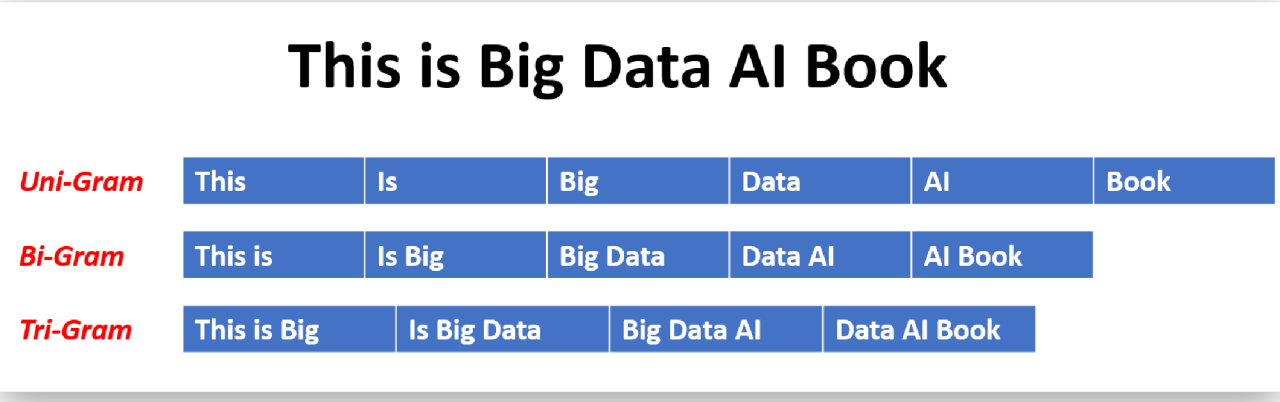
Named Entity Recognition (NER)

- Identifying and classifying named entities in a text document.
- Named entities can be names of people, locations, organizations, etc.



N-grams

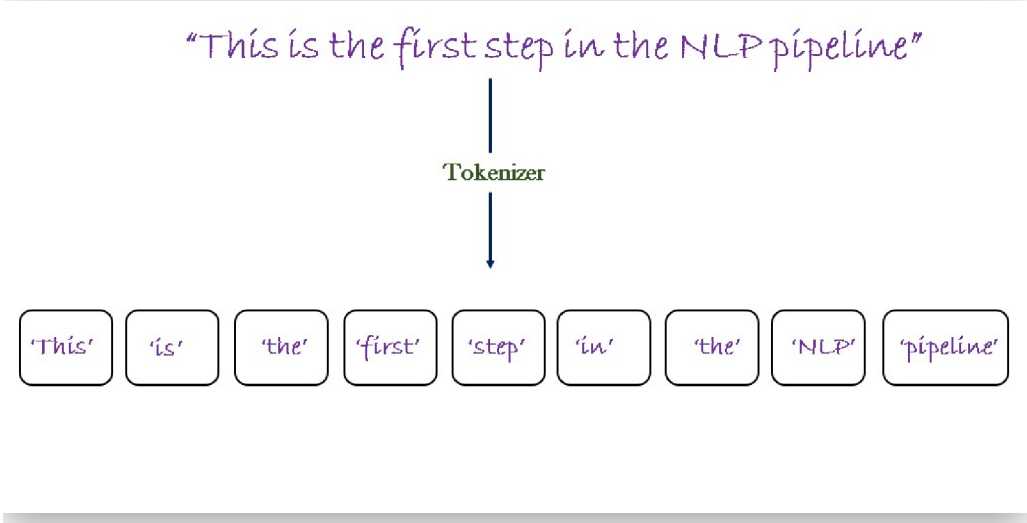
- A contiguous sequence of n items from a given sample of text or speech.
- The items can be phonemes, syllables, letters, words or base pairs according to the application.



Tokenization

- Splitting text into smaller chunks which is called token.
- Tokenizing By Words: Splitting text into words.
- Tokenizing By Sentences: Splitting text into sentences.

- White space tokenizer
- Word tokenizer
- Sentence tokenizer
- Character tokenizer
- N-gram tokenizer
- Regular Expression tokenizer



Normalization

Refers to several different processes used to make text consistent and create a label playing field for the tokens in the text mining process.

- **Case Normalization**: Converting all text to either upper case or lower case.
- **Removing Punctuation**: Removing all punctuation marks from the text.
- **Expanding Contractions**: Expanding contractions to their full form.
- **Convert Numbers to Words**: Converting numbers to their word equivalents.

Stemming

- Removes prefixes or suffixes from words to obtain a common base or root form.
- It uses heuristics and rule-based approaches to simplify words.
- Example: studying → study, dogs, dog's, dogs' → dog

Lemmatization

- Lemmatization involves reducing words to their base or dictionary form, considering the context and meaning of the words.
- It typically uses lexical knowledge and language rules.
- Better than stemming as it uses a dictionary-based approach.
- Example: am, are, is ⟶ be, Better, best ⟶ good

Summary of Key Concepts:

- **Corpus** : A large collection of unstructured texts.
- **Stop Words**: Most commonly used words in a language.
- **POS Tagging**: Assigning a part-of-speech tag to each word in a sentence.
- **NER**: Identifying and classifying named entities in a text document.
- **N-grams**: A contiguous sequence of n items from a given sample of text or speech.
- **Tokenization**: Splitting text into smaller chunks.
- **Normalization**: Making text consistent and creating a level playing field for the tokens.
- **Stemming**: Removing prefixes or suffixes from words to obtain a common base or root form.
- **Lemmatization**: Reducing words to their base or dictionary form, considering the context and meaning of the words.

Convert tokens into numbers

- Bag of Words
- TF-IDF
- Word Embedding

Bag of Words

- **Step 1**: Data Collection - Consider the following three sentences:
  - I love to eat Burgers.
  - I love to eat Fries.
  - I love to eat Pizza.
- **Step 2**: Tokenization - Splitting the sentences into words:
  - [I, love, to, eat, Burgers]
  - [I, love, to, eat, Fries]
  - [I, love, to, eat, Pizza]

- **Step 3**: Vocabulary Creation - Creating a vocabulary of unique words from the corpus:
    - I, love, to, eat, Burgers, Fries, Pizza
- **Step 4**: Vectorization - Converting each sentence into a vector of numbers:

	I	love	to	eat	Burgers	Fries	Pizza
Sentence 1	1	1	1	1	1	0	0
Sentence 2	1	1	1	1	0	1	0
Sentence 3	1	1	1	1	0	0	1

Bag of Words: Pros and Cons

- **Pros**: Simple and easy to understand.
- **Cons**:
  - Does not consider the order of words.
  - Does not consider the context of words.
  - Does not consider the semantics of words.

TF-IDF

Term Frequency - Inverse Document Frequency

TF-IDF

Statistical measure that evaluates how relevant a word is to a document in a collection of documents.

- **Term Frequency (TF)**: Measures how frequently a term occurs in a document.
- **Inverse Document Frequency (IDF)**: Measures how important a term is in a collection of documents.

TF-IDF

- Consider the following three sentences:
  - Document 1: "The cat sat on the mat. The cat was fluffy and white."
  - Document 2: "The dog chased the ball. The dog barked loudly."
  - Document 3: "The horse galloped across the field. The horse was brown and strong."
- **Goal**: Score the documents for a search query "cat".
- **Step 1**: Calculate Term Frequency (TF) for each word in each document.
  - In Document 1, the word "cat" occurs twice. So, TF for "cat" in Document 1 is 2.
  - In Document 2, the word "cat" does not occur. So, TF for "cat" in Document 2 is 0.
  - In Document 3, the word "cat" does not occur. So, TF for "cat" in Document 3 is 0.

TF-IDF

- **Step 2**: Calculate Inverse Document Frequency (IDF) for each word in each document.
  - IDF for "cat" = log (Total number of documents / Number of documents with the word "cat" in it)
  - IDF for "cat" = log (3 / 1) = 0.477
- **Step 3**: Calculate TF-IDF for each word in each document.
  - Multiply TF and IDF for each word in each document.
  - TF-IDF for "cat" in Document 1 = TF * IDF = 2 * 0.477 = 0.954
  - TF-IDF for "cat" in Document 2 = TF * IDF = 0 * 0.477 = 0
  - TF-IDF for "cat" in Document 3 = TF * IDF = 0 * 0.477 = 0

TF-IDF: Pros and Cons

- Pros:
  - Useful for finding important words in a document.
  - Simple and easy to understand.
- Cons:
  - Cannot capture the semantics of words, co-occurrence in different documents.

Word Embedding

Word Embedding

- Word2Vec
- GloVe
- FastText

Word Embedding

- Each word is represented by a vector
- Words with similar meanings or contexts tend to have similar vector representations.

Let's consider the following sentence:

The quick brown fox jumps over the lazy dog.

- Create a vocabulary of unique words from the corpus. **[The, quick, brown, fox, jumps, over, lazy, dog]**
- Goal is to assign number to each word in the vocabulary.
- We can assign random numbers to each word in the vocabulary.

Let's consider the following sentences

```
That's a great idea.
Just lost my phone. Great !.
```

- A word can have different meanings in different contexts.
- Single number representation of a word cannot capture the context of the word.
- We need a vector representation of a word to capture the context of the word.

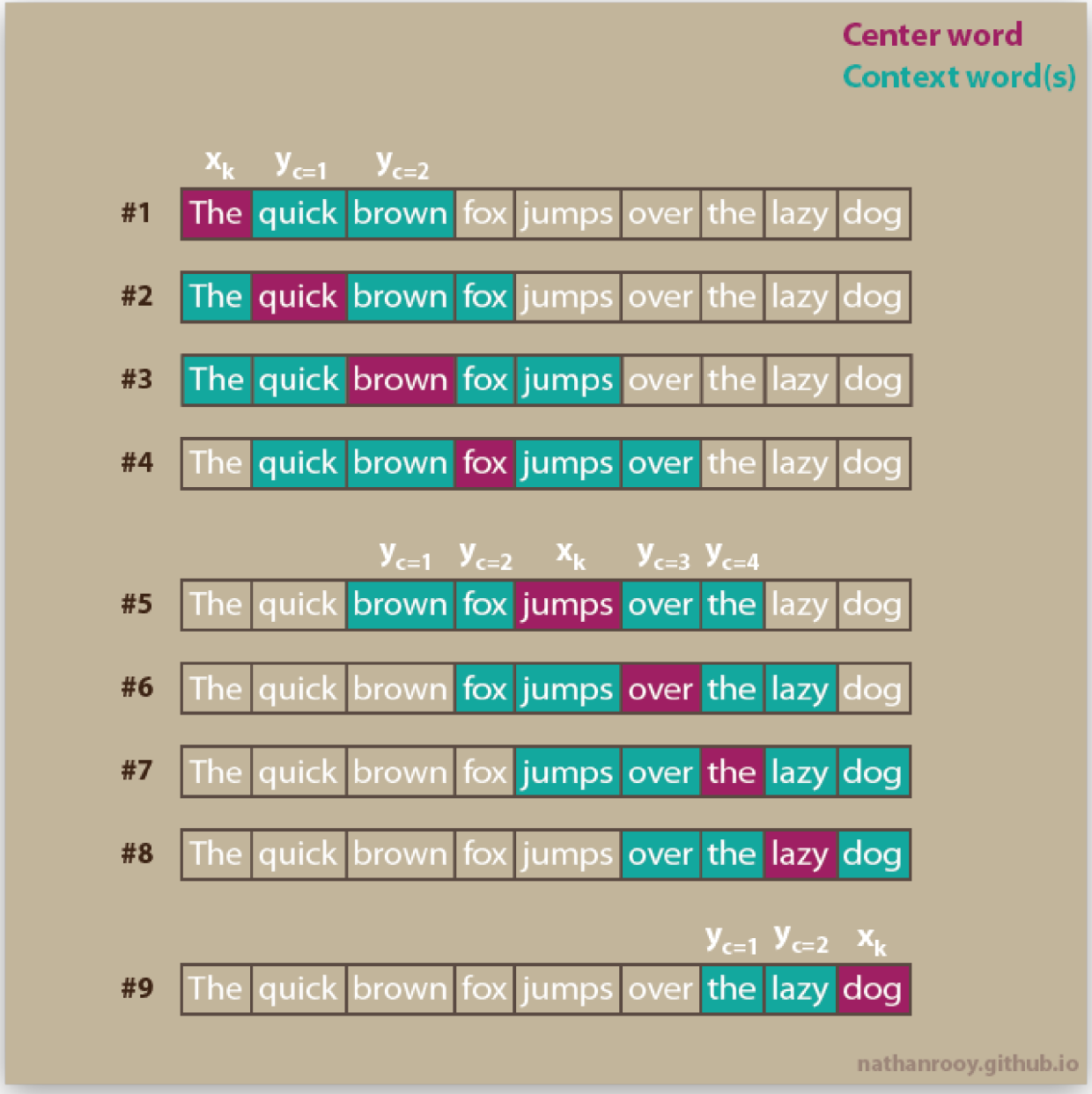
Back to our example sentence:

The quick brown fox jumps over the lazy dog.

- Let's define a window size of 5.
- Let's consider the word "fox" and its context words within the window size of 5.
- The context words are: "quick", "brown", "jumps", "over".

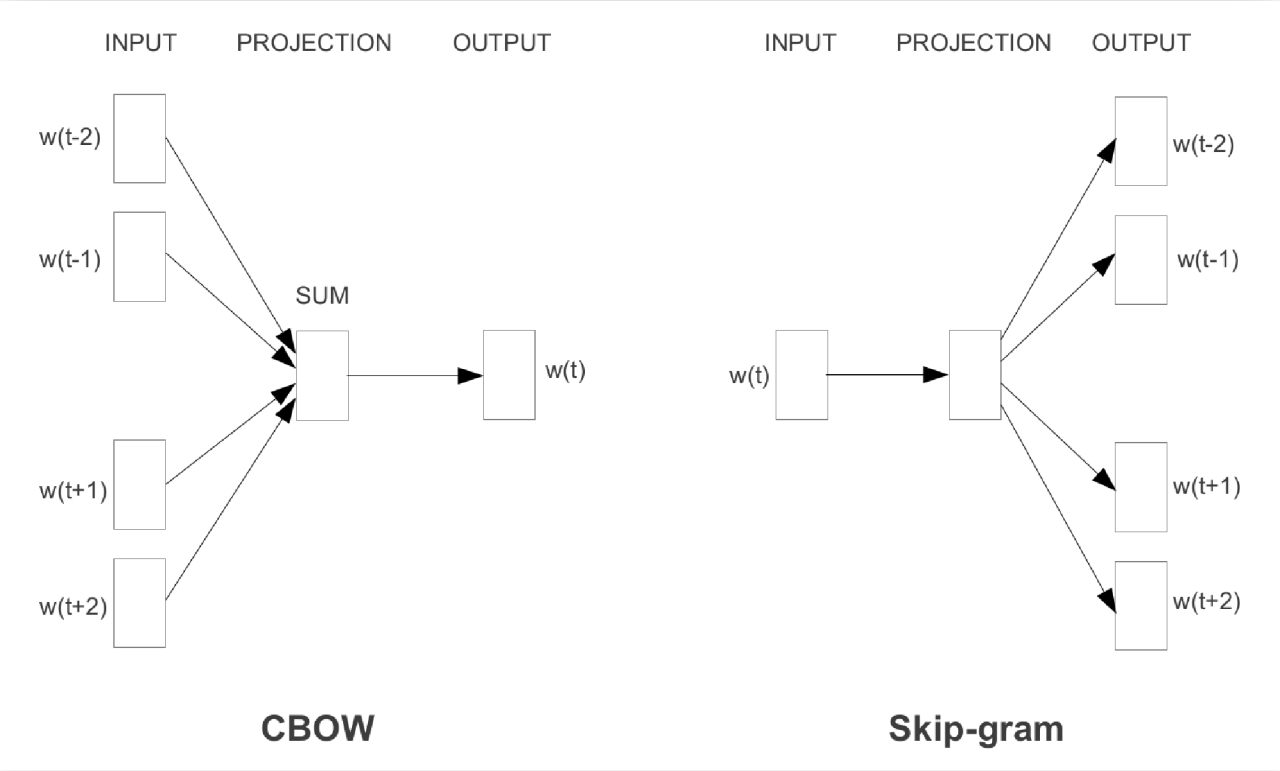
Context Word

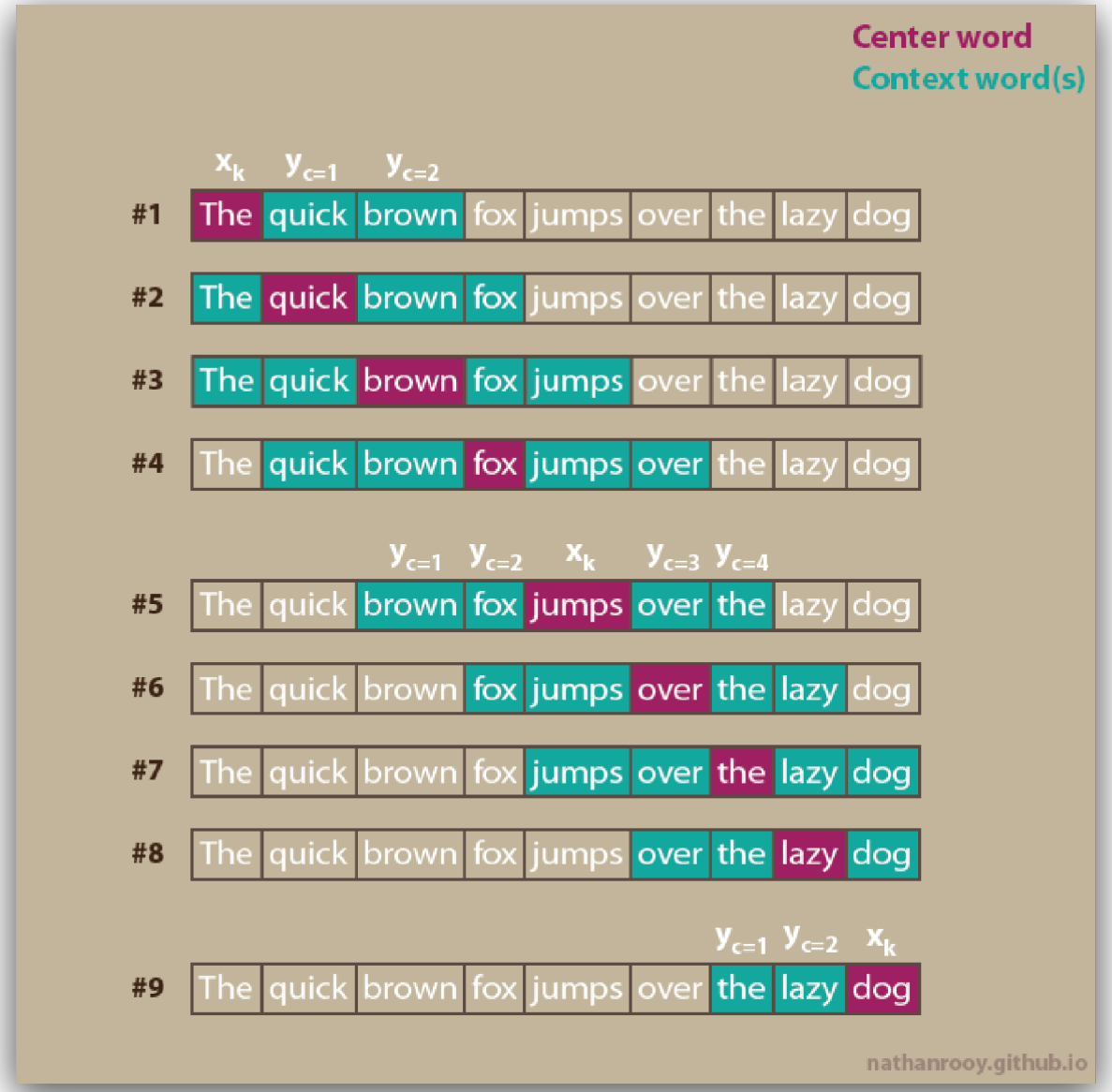
Target Word

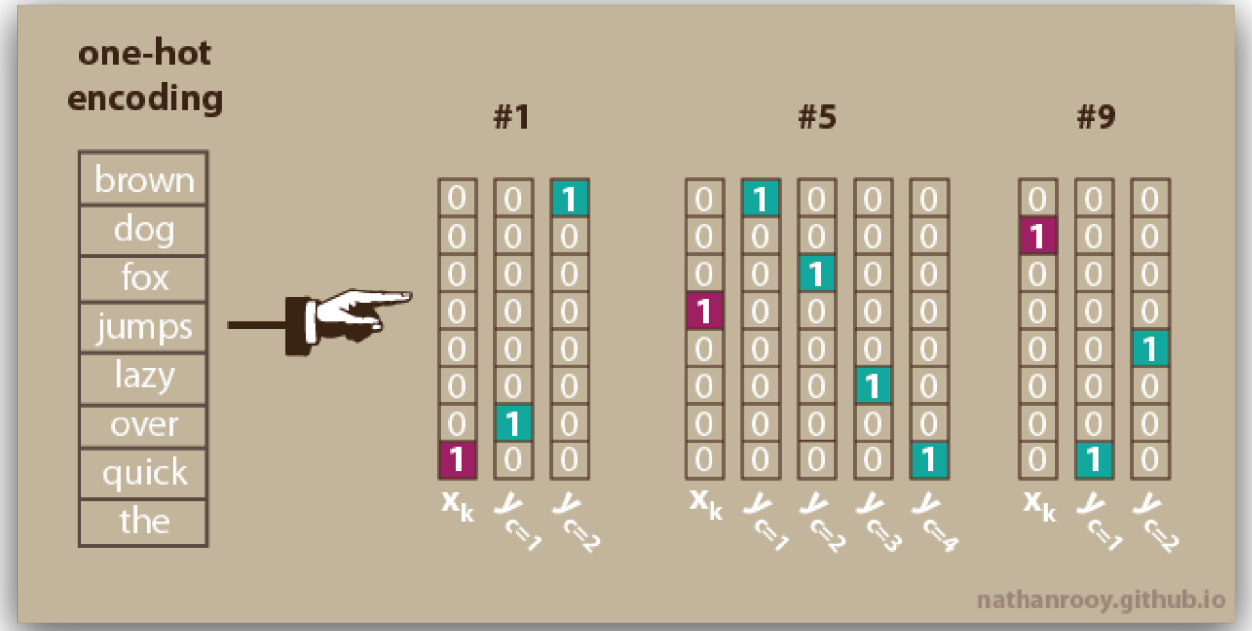


- Predicting the context words from the target word is called **Continuous Bag of Words (CBOW)**.
- Predicting the target word from the context words is called **skip-gram**.

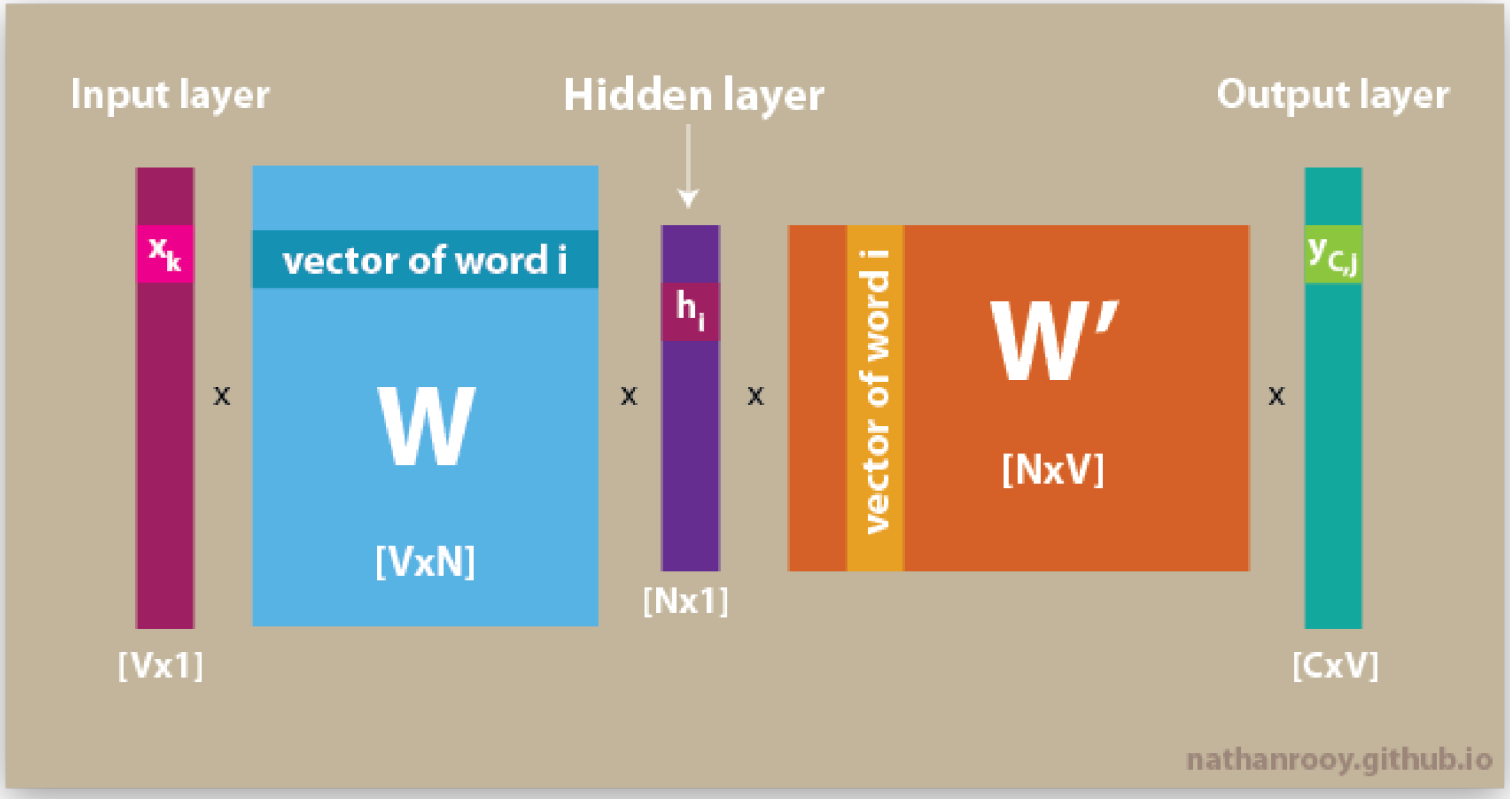
CBOW vs skip-gram




Center word
Context word(s)
x_k $y_{c=1}$ $y_{c=2}$
#1 The quick brown fox jumps over the lazy dog
#2 The quick brown fox jumps over the lazy dog
#3 The quick brown fox jumps over the lazy dog
#4 The quick brown fox jumps over the lazy dog
$y_{c=1}$ $y_{c=2}$ x_k $y_{c=3}$ $y_{c=4}$
#5 The quick brown fox jumps over the lazy dog
#6 The quick brown fox jumps over the lazy dog
#7 The quick brown fox jumps over the lazy dog
#8 The quick brown fox jumps over the lazy dog
$y_{c=1}$ $y_{c=2}$ x_k
#9 The quick brown fox jumps over the lazy dog
nathanrooy.github.io


one-hot encoding
brown
dog
fox
jumps
lazy
over
quick
the
#1
#5
#9
x_k $y_{c=1}$ $y_{c=2}$ $y_{c=3}$ $y_{c=4}$
nathanrooy.github.io

skip-gram network architecture



skip-gram results

Input word	brown	dog	fox	jumps	lazy	over	quick	the
fox	2.45e-01	4.34e-04	4.45e-04	2.53e-01	2.34e-05	2.53e-01	2.45e-01	7.62e-07
lazy	5.81e-05	3.32e-01	2.42e-04	1.11e-05	1.91e-04	3.33e-01	4.51e-04	3.33e-01
dog	1.85e-07	3.17e-04	1.31e-03	1.29e-04	4.98e-01	1.42e-05	4.86e-06	4.99e-01

Explore Word Embedding

https://projector.tensorflow.org/

Coding Time

Resources:

https://medium.com/nlplanet/awesome-nlp-18-high-quality-resources-for-studying-nlp-1b4f7fd87322

https://github.com/keon/awesome-nlp

https://realpython.com/nltk-nlp-python/

https://realpython.com/natural-language-processing-spacy-python/

https://towardsdatascience.com/topic-modeling-on-pycaret-2ce0c65ba3ff

https://www.kaggle.com/code/yclaudel/find-similar-articles-with-tf-idf

https://towardsdatascience.com/basic-concepts-of-natural-language-processing-nlp-models-and-python-implementation-88a589ce1fc0

https://www.kdnuggets.com/2017/02/natural-language-processing-key-terms-explained.html

https://nathanrooy.github.io/posts/2018-03-22/word2vec-from-scratch-with-python-and-numpy/

Thank You

Happy Learning 🚀 !!

contact@meftaul.com