

# Natural Language Processing

( <https://bit.ly/NLPLEE> )

# Table of Contents

1. Introduction to Natural Language Processing: 3
2. Feature Extraction Methods: 48
3. Developing a Text Classifier: 104
4. Collecting Text Data with Web Scraping and APIs: 182
5. Topic Modeling: 211
6. Vector Representation: 242
7. Text Generation and Summarization: 265
8. Sentiment Analysis: 285



# 1. Introduction to Natural Language Processing

# Overview

- In this lesson, you will learn the difference between Natural Language Processing (NLP) and basic text analytics.
- You will implement various preprocessing tasks such as tokenization, lemmatization, stemming, stop word removal, and more.
- By the end of this lesson, you will have a deep understanding of the various phases of an NLP project, from data collection to model deployment.

# Introduction

- Before we can get into NLP in any depth, we first need to understand what natural language is.
- To put it in simple terms, it is a means for us to express our thoughts and ideas.
- To define it more specifically, language is a mutually agreed upon set of protocols involving words/sounds that we use to communicate with each other.

- **Morphological level:** This level deals with understanding word structure and word information.
- **Lexical level:** This level deals with understanding the part of speech of the word.
- **Syntactic level:** This level deals with understanding the syntactic analysis of a sentence, or parsing a sentence.
- **Semantic level:** This level deals with understanding the actual meaning of a sentence.
- **Discourse level:** This level deals with understanding the meaning of a sentence beyond just the sentence level, that is, considering the context.
- **Pragmatic level:** This level deals with using real-world knowledge to understand the sentence.

# History of NLP

- NLP is a field that has emerged from various other fields such as artificial intelligence, linguistics, and data science.
- With the advancement of computing technologies and the increased availability of data, NLP has undergone a huge change.
- Previously, a traditional rule-based system was used for computations, in which you had to explicitly write hardcoded rules.

# History of NLP

- Consider an example. Let's say we have to extract the names of some politicians from a set of political news articles.
- So, if we want to apply rule-based grammar, we must manually craft certain rules based on human understanding of language.
- Some of the rules for extracting a person's name can be that the word should be a proper noun, every word should start with a capital letter, and so on

# Text Analytics and NLP

- Text analytics is the method of extracting meaningful insights and answering questions from text data, such as those to do with the length of sentences, length of words, word count, and finding words from the text. Let's understand this with an example.
- Suppose we are doing a survey using news articles.
- Let's say we have to find the top five countries that contributed the most in the field of space technology in the past 5 years.

# Text Analytics and NLP

- It is important here to note the difference between text analytics and NLP.
- The art of extracting useful insights from any given text data can be referred to as text analytics.
- NLP, on the other hand, helps us in understanding the semantics and the underlying meaning of text, such as the sentiment of a sentence, top keywords in text, and parts of speech for different words.

# Text Analytics and NLP

It can be broadly categorized into two types: Natural Language Understanding (NLU) and Natural Language Generation (NLG). A proper explanation of these terms is provided here:

- NLU
- NLG

# Complete Exercise 1.01: Basic Text Analytics

# Various Steps in NLP

- We've talked about the types of computations that are done with natural language.
- Apart from these basic tasks, you can also design your own tasks as per your requirements.
- In the coming sections, we will discuss the various preprocessing tasks in detail and demonstrate each of them with an exercise.

# Tokenization

- Tokenization refers to the procedure of splitting a sentence into its constituent parts—the words and punctuation that it is made up of.
- It is different from simply splitting the sentence on whitespaces, and instead actually divides the sentence into constituent words, numbers (if any), and punctuation, which may not always be separated by whitespaces

# PoS Tagging

- In NLP, the term PoS refers to parts of speech. Parts-of-Speech (PoS) tagging refers to the process of tagging words within sentences with their respective PoS.
- We extract the PoS of tokens constituting a sentence so that we can filter out the PoS that are of interest and analyze them.

# PoS Tagging

- For example, if we look at the sentence, "The sky is blue," we get four tokens, namely "The," "sky," "is," and "blue", with the help of tokenization.
- Now, using a PoS tagger, we tag the PoS for each word/token. This will look as follows:

```
[('The', 'DT'), ('sky', 'NN'), ('is', 'VBZ'), ('blue', 'JJ')]
```

# Stop Word Removal

- Stop words are the most frequently occurring words in any language and they are just used to support the construction of sentences and do not contribute anything to the semantics of a sentence.
- So, we can remove stop words from any text before an NLP process, as they occur very frequently and their presence doesn't have much impact on the sense of a sentence.

# Complete Exercise 1.02: Tokenization of a Simple Sentence

# Complete Exercise 1.03: PoS Tagging

# Complete Exercise 1.04: Stop Word Removal

# Text Normalization

- There are some words that are spelled, pronounced, and represented differently—for example, words such as Mumbai and Bombay, and US and United States.
- Although they are different, they refer to the same thing.
- There are also different forms of words that need to be converted into base forms.

# Spelling Correction

Spelling correction is one of the most important tasks in any NLP project. It can be time-consuming, but without it, there are high chances of losing out on important information.

Spelling correction is executed in two steps:

1. Identify the misspelled word, which can be done by a simple dictionary lookup. If there is no match found in the language dictionary, it is considered to be misspelled.
2. Replace it or suggest the correctly spelled word. There are a lot of algorithms for this task.

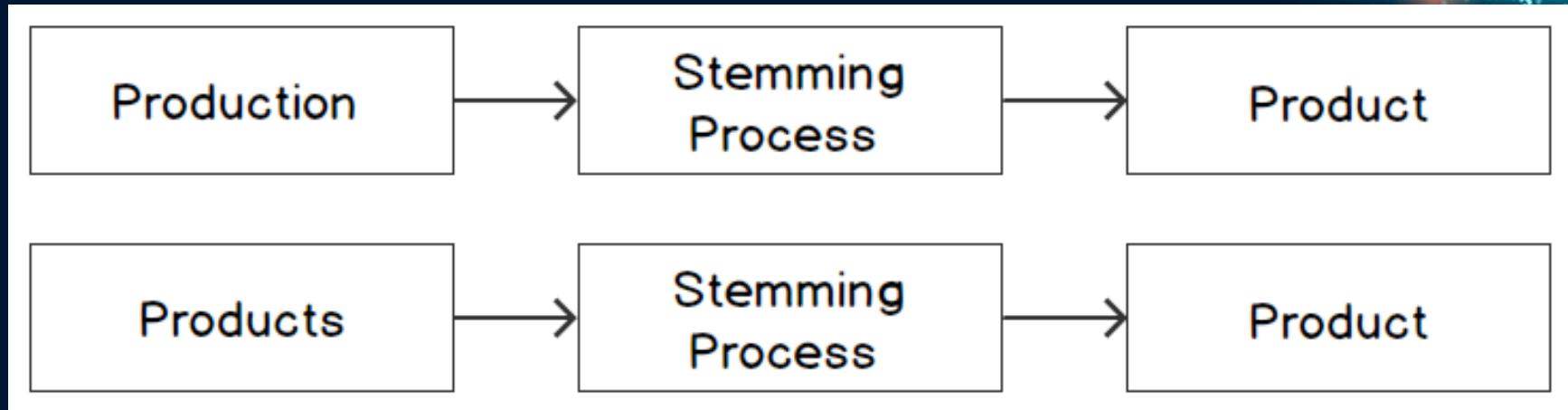
# Complete Exercise 1.05: Text Normalization

# **Complete Exercise 1.06: Spelling Correction of a Word and a Sentence**

# Stemming

- In most languages, words get transformed into various forms when being used in a sentence.
- For example, the word "product" might get transformed into "production" when referring to the process of making something or transformed into "products" in plural form.
- It is necessary to convert these words into their base forms, as they carry the same meaning in any case.

# Stemming



# Complete Exercise 1.07: Using Stemming

# Lemmatization

- Sometimes, the stemming process leads to incorrect results.
- For example, in the last exercise, the word *battling* was transformed to "battl", which is not a word.
- To overcome such problems with stemming, we make use of lemmatization.
- Lemmatization is the process of converting words to their base grammatical form, as in "battling" to "battle," rather than just randomly axing words.

# Complete Exercise 1.08: Extracting the Base Word Using Lemmatization

# Named Entity Recognition (NER)

- NER is the process of extracting important entities, such as person names, place names, and organization names, from some given text.
- These are usually not present in dictionaries. So, we need to treat them differently.
- The main objective of this process is to identify the named entities (such as proper nouns) and map them to categories, which are already defined.

# Named Entity Recognition (NER)

- In the case of search algorithms, let's suppose we have to create a search engine, meant specifically for courses. If we were to submit a given query for all the words, the search would take a lot of time.
- Instead, if we extract the top entities from all the courses using NER and run a search query on the entities rather than all the content, the speed of the system would increase dramatically.

# Complete Exercise 1.09: Treating Named Entities

# Word Sense Disambiguation

- There's a popular saying: "A man is known by the company he keeps." Similarly, a word's meaning depends on its association with other words in a sentence.
- This means two or more words with the same spelling may have different meanings in different contexts. This often leads to ambiguity.
- Word sense disambiguation is the process of mapping a word to the sense that it should carry

# Word Sense Disambiguation

He knows how to **play** Harmonica.

We **play** only soccer.

Please **play** the next song.

What  
does **play**  
mean  
here?

# Word Sense Disambiguation

- For example, suppose we have a sentence such as "We play only soccer" in a given text. Now, we need to find the meaning of the word "play" in this sentence.
- In the Lesk algorithm, each word with ambiguous meaning is saved in background synsets. In this case, the word "play" will be saved with all possible definitions.
- Let's say we have two definitions of the word "play":
  1. Play: Participating in a sport or game
  2. Play: Using a musical instrument

# Sentence Boundary Detection

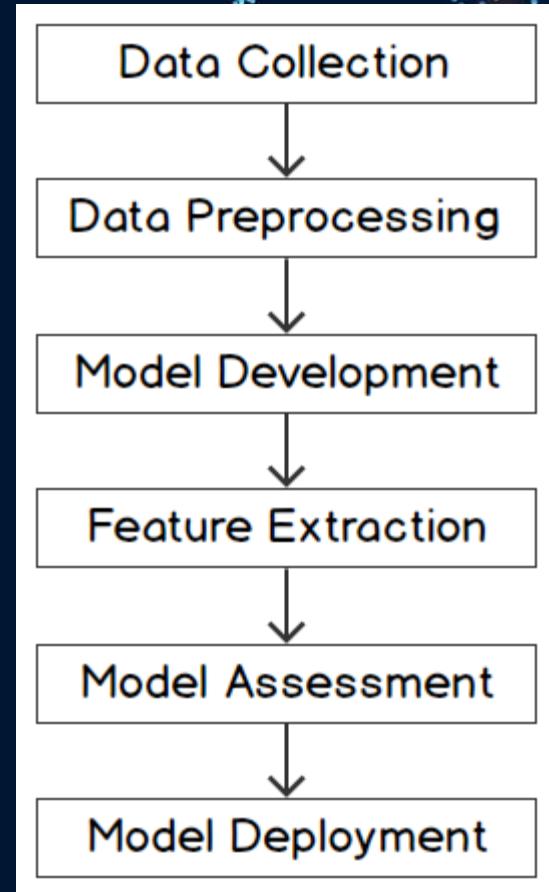
- Sentence boundary detection is the method of detecting where one sentence ends and another begins.
- If you are thinking that this sounds pretty easy, as a period (.) or a question mark (?) denotes the end of a sentence and the beginning of another sentence, then you are wrong.

# **Complete Exercise 1.10: Word Sense Disambiguation**

# Complete Exercise 1.11: Sentence Boundary Detection

# Complete Activity 1.01: Preprocessing of Raw Text

- We can divide an NLP project into several sub-projects or phases.
- These phases are completed in a particular sequence.
- This tends to increase the overall efficiency of the process, as memory usage changes from one phase to the next.
- An NLP project has to go through six major phases, which are outlined in the following figure:



# Data Collection

- This is the initial phase of any NLP project. Our sole purpose is to collect data as per our requirements.
- For this, we may either use existing data, collect data from various online repositories, or create our own dataset by crawling the web.
- In our case, we will collect different email data. We can even get this data from our personal emails as well, to start with.

# Data Preprocessing

It is necessary to clean the collected data to ensure effectiveness and accuracy. In our case, we will follow these preprocessing steps:

1. Converting all the text data to lowercase
2. Stop word removal
3. Text normalization, which will include replacing all numbers with some common term and replacing punctuation with empty strings
4. Stemming and lemmatization

# Feature Extraction

- Computers understand only binary digits: 0 and 1. As such, every instruction we feed into a computer gets transformed into binary digits.
- Similarly, machine learning models tend to understand only numeric data.
- Therefore, it becomes necessary to convert text data into its equivalent numerical form.

# Model Development

- Once the feature set is ready, we need to develop a suitable model that can be trained to gain knowledge from the data.
- These models are generally statistical, machine learning-based, deep learning-based, or reinforcement learning-based.
- In our case, we will build a model that is capable of differentiating between important and unimportant emails.

# Model Assessment

- After developing a model, it is essential to benchmark it. This process of benchmarking is known as model assessment.
- In this step, we will evaluate the performance of our model by comparing it to others.
- This can be done by using different parameters or metrics.
- These parameters include precision, recall, and accuracy.

# Model Deployment

- This is the final stage for most industrial NLP projects.
- In this stage, the models are put into production.
- They are either integrated into an existing system or new products are created by keeping this model as a base.
- In our case, we will deploy our model to production, so that it can classify emails as important and unimportant in real time.

# Summary

- In this lesson, we learned about the basics of NLP and how it differs from text analytics.
- We covered the various preprocessing steps that are included in NLP, such as tokenization, PoS tagging, stemming, lemmatization, and more.
- We also looked at the different phases an NLP project has to pass through, from data collection to model deployment.

# 2. Feature Extraction Methods

# Overview

- In this lesson, you will be able to categorize data based on its content and structure.
- You will be able to describe preprocessing steps in detail and implement them to clean up text data.
- You will learn about feature engineering and calculate the similarity between texts.

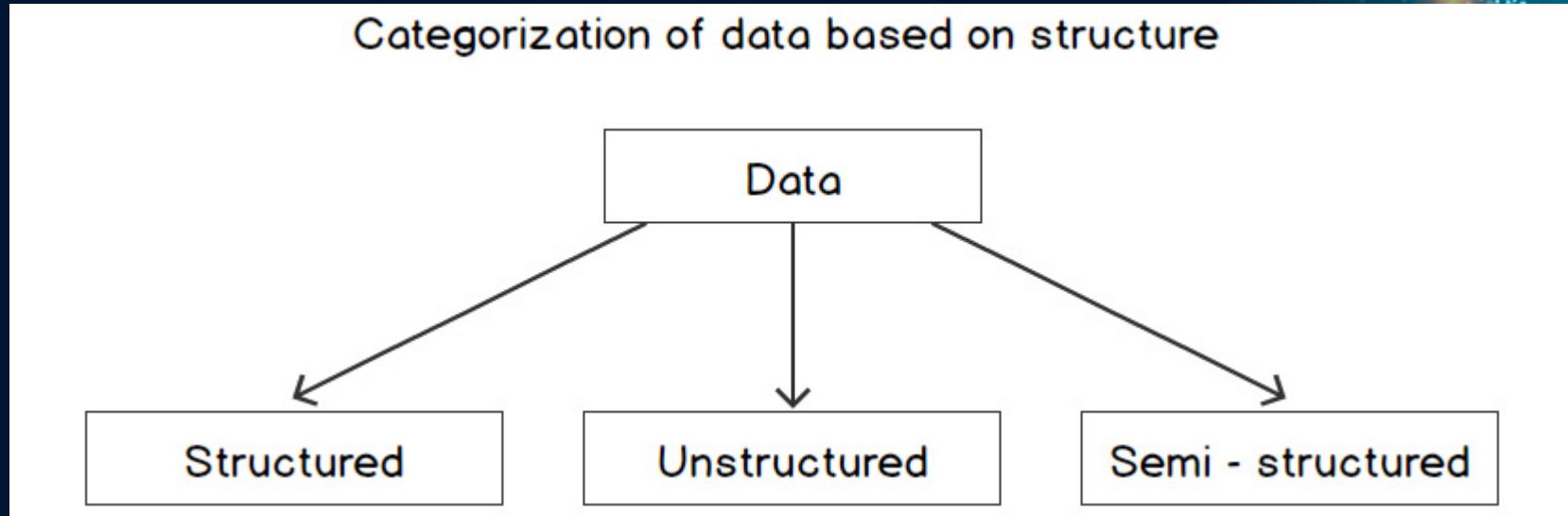
# Introduction

- In the previous lesson, we learned about the concepts of Natural Language Processing (NLP) and text analytics.
- We also took a quick look at various preprocessing steps.
- In this lesson, we will learn how to make text understandable to machine learning algorithms.

# Types of Data

- To deal with data effectively, we need to understand the various forms in which it exists.
- First, let's explore the types of data that exist.
- There are two main ways to categorize data (by structure and by content), as explained in the upcoming sections.

# Categorizing Data Based on Structure



# Structured data:

- This is the most organized form of data.
- It is represented in tabular formats such as Excel files and Comma-Separated Value (CSV) files.
- The following image shows what structured data usually looks like:

Name	Age	Location
Ram	25	Delhi
Shyam	28	Banglore
Jon	35	Kolkata
Madhu	28	Mumbai
Hari	56	Chennai

# Semi-structured data:

- This type of data is not presented in a tabular structure, but it can be transformed into a table.
- Here, information is usually stored between tags following a definite pattern.
- XML and HTML files can be referred to as semi-structured data.
- The following screenshot shows how semi-structured data can appear:

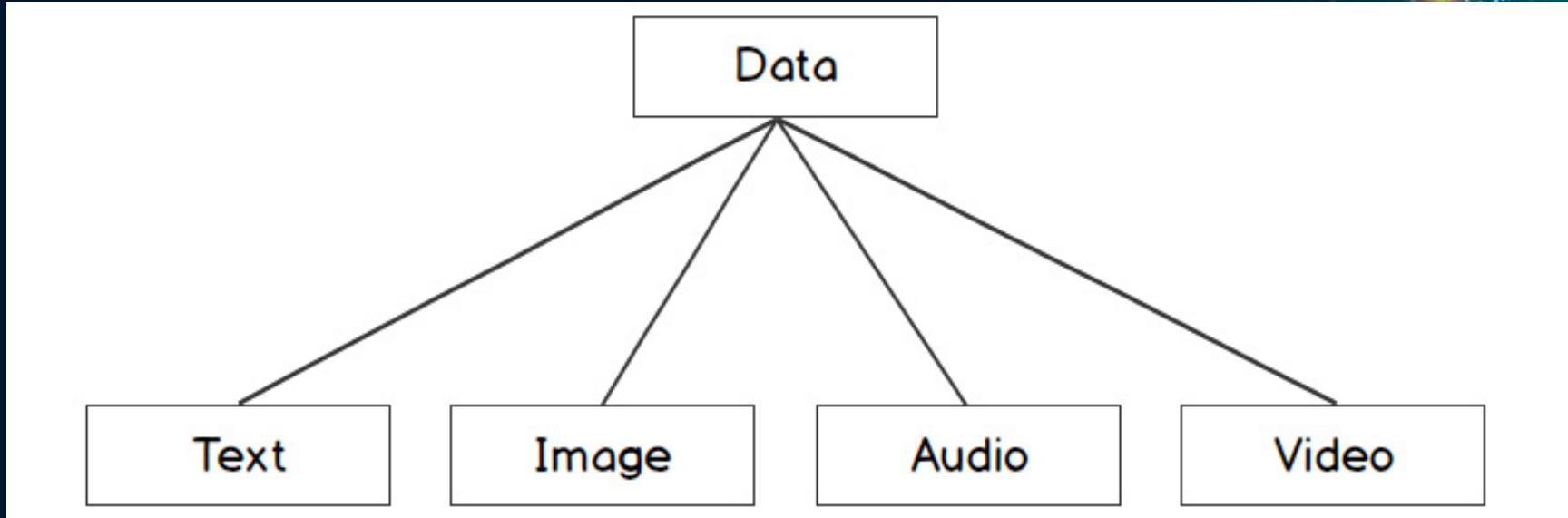
```
<student>
  <name>
    Jagat
  </name>
  <roll_number>
    3
  </roll_number>
  <rank>
    1
  </rank>
  <qualification>
    <qualification1>
      B.Tech
    </qualification1>
    <qualification2>
      M.Tech
    </qualification2>
  </qualification>
</student>
<student>
  <name>
    Jani
  </name>
  <roll_number>
    5
  </roll_number>
  <rank>
    3
  </rank>
  <qualification>
    <qualification1>
      B.A
    </qualification1>
  </qualification>
</student>
```

# Unstructured data:

- This type of data is the most difficult to deal with. Machine learning algorithms would find it difficult to comprehend unstructured data without any loss of information.
- Text corpora and images are examples of unstructured data.
- The following image shows what unstructured data looks like:

We have three employees in Block A named James, Noah and Charlie. Their ages are 34,32 and 45 respectively. Charlie is from New Jersey while as Noah and James come from Waikiki.

# Categorizing Data Based on Content



# Categorizing Data Based on Content

Let's look at each category here:

- **Text data:** This refers to text corpora consisting of written sentences. This type of data can only be read. An example would be the text corpus of a course.
- **Image data:** This refers to pictures that are used to communicate messages. This type of data can only be seen.
- **Audio data:** This refers to voice recordings, music, and so on. This type of data can only be heard.
- **Video data:** A continuous series of images coupled with audio forms a video. This type of data can be seen as well as heard.

# Cleaning Text Data

- The text data that we are going to discuss here is unstructured text data, which consists of written sentences.
- Most of the time, this text data cannot be used as it is for analysis because it contains some noisy elements, that is, elements that do not really contribute much to the meaning of the sentence at all.
- These noisy elements need to be removed because they do not contribute to the meaning and semantics of the text

# Cleaning Text Data

To achieve this, methods such as stopword removal, tokenization, and stemming are used. We will explore them in detail in the upcoming sections. Before we do so, let's get acquainted with some basic NLP libraries that we will be using here:

- Re
- Textblob
- keras

# Tokenization

- Tokenization and word tokenizers were briefly described in lesson 1, Introduction to Natural Language Processing.
- Tokenization is the process of splitting sentences into their constituents; that is, words and punctuation.
- Let's perform a simple exercise to see how this can be done using various packages.

# Complete Exercise 2.01: Text Cleaning and Tokenization

# Complete Exercise 2.02: Extracting n-grams

# Complete Exercise 2.03: Tokenizing Text with Keras and TextBlob

# Types of Tokenizers

There are different types of tokenizers that come in handy for specific tasks. Let's look at the ones provided by nltk one by one:

- Whitespace tokenizer
- Tweet tokenizer
- MWE tokenizer
- Regular expression tokenizer
- WordPunctTokenizer

# Complete Exercise 2.04: Tokenizing Text Using Various Tokenizers

# Stemming

- In many languages, the base forms of words change when they're used in sentences.
- For example, the word "produce" can be written as "production" or "produced" or even "producing," depending on the context.
- The process of converting a word back into its base form is known as stemming.

# RegexpStemmer

- RegexpStemmer uses regular expressions to check whether morphological or structural prefixes or suffixes are present.
- For instance, in many cases, verbs in the present continuous tense (the present tense form ending with "ing") can be restored to their base form simply by removing "ing" from the end; for example, "playing" becomes "play".

# **Complete Exercise 2.05: Converting Words in the Present Continuous Tense into Base Words with RegexpStemmer**

# The Porter Stemmer

- The Porter stemmer is the most common stemmer for dealing with English words.
- It removes various morphological and inflectional endings (such as suffixes, prefixes, and the plural "s") from English words.
- In doing so, it helps us extract the base form of a word from its variations.

# Complete Exercise 2.06: Using the Porter Stemmer

# Lemmatization

- As we saw in the previous section, there is a problem with stemming. It often generates meaningless words.
- Lemmatization deals with such cases by using vocabulary and analyzing the words' morphologies.
- It returns the base forms of words that can be found in dictionaries.

# Complete Exercise 2.07: Performing Lemmatization

# Complete Exercise 2.08: Singularizing and Pluralizing Words

# Language Translation

- You might have used Google Translate before, which gives the exact translation of a word in another language; this is an example of language translation or machine translation.
- In Python, we can use TextBlob to translate text from one language into another.
- TextBlob provides a method called `translate()`, in which you have to pass text in the source language.

# Complete Exercise 2.09: Language Translation

# Stop-Word Removal

- Stop words, such as "am," "the," and "are," occur frequently in text data.
- Although they help us construct sentences properly, we can find the meaning even if we remove them.
- This means that the meaning of text can be inferred even without them.
- So, removing stop words from text is one of the preprocessing steps in NLP tasks.

# Complete Exercise 2.10: Removing Stop Words from Text

# **Complete Activity 2.01: Extracting Top Keywords from the News Article**

# Feature Extraction from Texts

Features can be classified into two different categories:

- **General features:** These features are statistical calculations and do not depend on the content of the text. Some examples of general features could be the number of tokens in the text, the number of characters in the text, and so on.
- **Specific features:** These features are dependent on the inherent meaning of the text and represent the semantics of the text. For example, the frequency of unique words in the text is a specific feature.

# Extracting General Features from Raw Text

- Let's consider these two sentences: "The sky is blue" and "The pillar is yellow".
- Here, the sentences have the same number of words (a general feature)—that is, four.
- But the individual constituent tokens are different.
- Let's complete an exercise to understand this better.

# Complete Exercise 2.11: Extracting General Features from Raw Text

# Complete Exercise 2.12: Extracting General Features from Text

# Bag of Words (BoW)

- In this technique, we convert each sentence into a vector. The length of this vector is equal to the number of unique words in all the documents. This is done in two steps:
  1. The vocabulary or dictionary of all the words is generated.
  2. The document is represented in terms of the presence or absence of all words.

# Complete Exercise 2.13: Creating a Bag of Words

# Zipf's Law

- According to Zipf's law, the number of times a word occurs in a corpus is inversely proportional to its rank in the frequency table.
- In simple terms, if the words in a corpus are arranged in descending order of their frequency of occurrence, then the frequency of the word at the  $i$ th rank will be proportional to  $1/i$ :

*Frequency of a word =  $1/rank\ of\ word\ in\ vocabulary$*

# Complete Exercise 2.14: Zipf's Law

# Term Frequency–Inverse Document Frequency

- Term Frequency-Inverse Document Frequency (TFIDF) is another method of representing text data in a vector format.
- Here, once again, we'll represent each document as a list whose length is equal to the number of unique words/tokens in all documents (corpus), but the vector here not only represents the presence and absence of a word, but also the frequency of the word—both in the current document and the whole corpus.

# Term Frequency–Inverse Document Frequency

- This technique is based on the idea that the rarely occurring words are better representatives of the document than frequently occurring words.
- Hence, this representation gives more weightage to the rarer or less frequent words than frequently occurring words. It does so with the following formula:

$$Tf - idf = \text{term-frequency} * \text{inverse document frequency}$$

# Complete Exercise 2.15: TFIDF Representation

# Finding Text Similarity – Application of Feature Extraction

There are different techniques for finding the similarity between two strings or texts. They are explained one by one here:

- **Cosine similarity:** The cosine similarity is a technique to find the similarity between the two vectors by calculating the cosine of the angle between them.
- As we know, the cosine of a zero-degree angle is 1 (meaning the cosine similarity of two identical vectors is 1), while the cosine of 180 degrees is -1 (meaning the cosine of two opposite vectors is -1).

# Finding Text Similarity – Application of Feature Extraction

- We can use this cosine angle to find the similarity between the vectors from 1 to -1.
- To use this technique in finding text similarity, we convert text into vectors using one of the previously discussed techniques and find the similarity between the vectors of the text. This is calculated as follows:

$$\text{Similarity} = \cos(\theta) = \cos(A, B) = A \cdot B / (|A| * |B|)$$

# Finding Text Similarity – Application of Feature Extraction

- **Jaccard similarity:** This is another technique that's used to calculate the similarity between the two texts, but it only works on BoW vectors.
- The Jaccard similarity is calculated as the ratio of the number of terms that are common between two text documents to the total number of unique terms present in those texts.

# Finding Text Similarity – Application of Feature Extraction

- Text 1: I like detective Byomkesh Bakshi.
- Text 2: Byomkesh Bakshi is not a detective; he is a truth seeker.

The common terms are "Byomkesh," "Bakshi," and "detective."

# Complete Exercise 2.16: Calculating Text Similarity Using Jaccard and Cosine Similarity

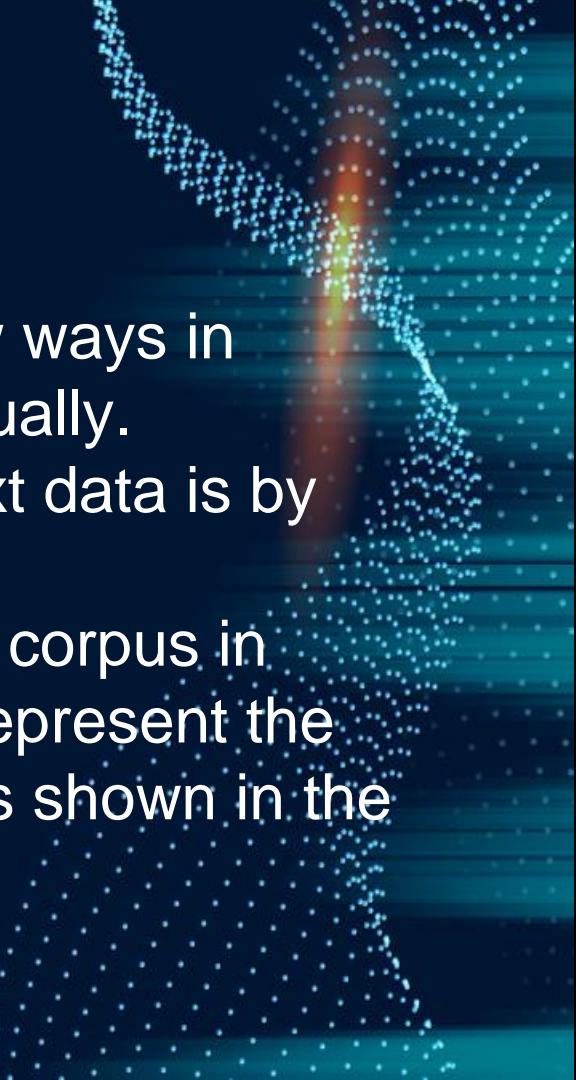
# Word Sense Disambiguation Using the Lesk Algorithm

- The Lesk algorithm is used for resolving word sense disambiguation.
- Suppose we have a sentence such as "On the bank of river Ganga, there lies the scent of spirituality" and another sentence, "I'm going to withdraw some cash from the bank".
- Here, the same word—that is, "bank"—is used in two different contexts.

# **Complete Exercise 2.17: Implementing the Lesk Algorithm Using String Similarity and Text Vectorization**

# Word Clouds

- Unlike numeric data, there are very few ways in which text data can be represented visually.
- The most popular way of visualizing text data is by using word clouds.
- A word cloud is a visualization of a text corpus in which the sizes of the tokens (words) represent the number of times they have occurred, as shown in the next image



of player window  
 many window  
 standard us better  
 read Armenian  
 want actually  
 two good really  
 others though God  
 part day card since  
 law available go  
 mean use point used  
 back said come must  
 called nMAX know one  
 non Ax support found  
 first thing probably  
 done around gun  
 seen still line  
 write driver game writes  
 article issue  
 call book run long  
 key nfor claim  
 issue  
 write  
 G9V G9V going  
 every may seem  
 bit might got  
 program netcom  
 post  
 nOrganization University person  
 nthis teaching thought start  
 na following nDistribution world  
 think  
 well sure even need work  
 application  
 source  
 MR. STEPHANOPOULOS

# Complete Exercise 2.18: Generating Word Clouds

# Other Visualizations

- **Visualizing sentences using a dependency parse tree:** Generally, the phrases constituting a sentence depend on each other. We depict these dependencies by using a tree structure known as a dependency parse tree.
- For instance, the word "helps" in the sentence "God helps those who help themselves" depends on two other words.
- These words are "God" (the one who helps) and "those" (the ones who are helped).
- **Visualizing named entities in a text corpus:** In this case, we extract the named entities from texts and highlight them by using different colors.

# Complete Exercise 2.19: Other Visualizations Dependency Parse Trees and Named Entities

# Complete Activity 2.02: Text Visualization

# Summary

- In this lesson, you have learned about various types of data and ways to deal with unstructured text data.
- Text data is usually extremely noisy and needs to be cleaned and preprocessed, which mainly consists of tokenization, stemming, lemmatization, and stop-word removal.
- After preprocessing, features are extracted from texts using various methods, such as BoW and TFIDF.

# 3. Developing a Text Classifier

# Overview

- This lesson starts with an introduction to the various types of machine learning methods, that is, the supervised and unsupervised methods.
- You will learn about hierarchical clustering and k-means clustering and implement them using various datasets.
- Next, you will explore tree-based methods such as random forest and XGBoost

# Introduction

- In this lesson, you will learn how to use these extracted features to develop machine learning models.
- These models are capable of solving real-world problems, such as detecting whether sentiments carried by texts are positive or negative, predicting whether emails are spam or not, and so on.

# Machine Learning

- Machine learning is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.
- Machine learning algorithms are fed with large amounts of data that they can work on to build a model.
- This model is later used by businesses to generate solutions that help them analyze data and build strategies for the future.

# Unsupervised Learning

- Unsupervised learning is the method by which algorithms learn patterns within data that is not labeled.
- Since labels (supervisors) are absent, it is referred to as unsupervised learning.
- In unsupervised learning, you provide the algorithm with the feature data and it learns patterns from the data on its own.

# Unsupervised Learning

Unsupervised learning is further classified into clustering and association:

- Clustering
- Association

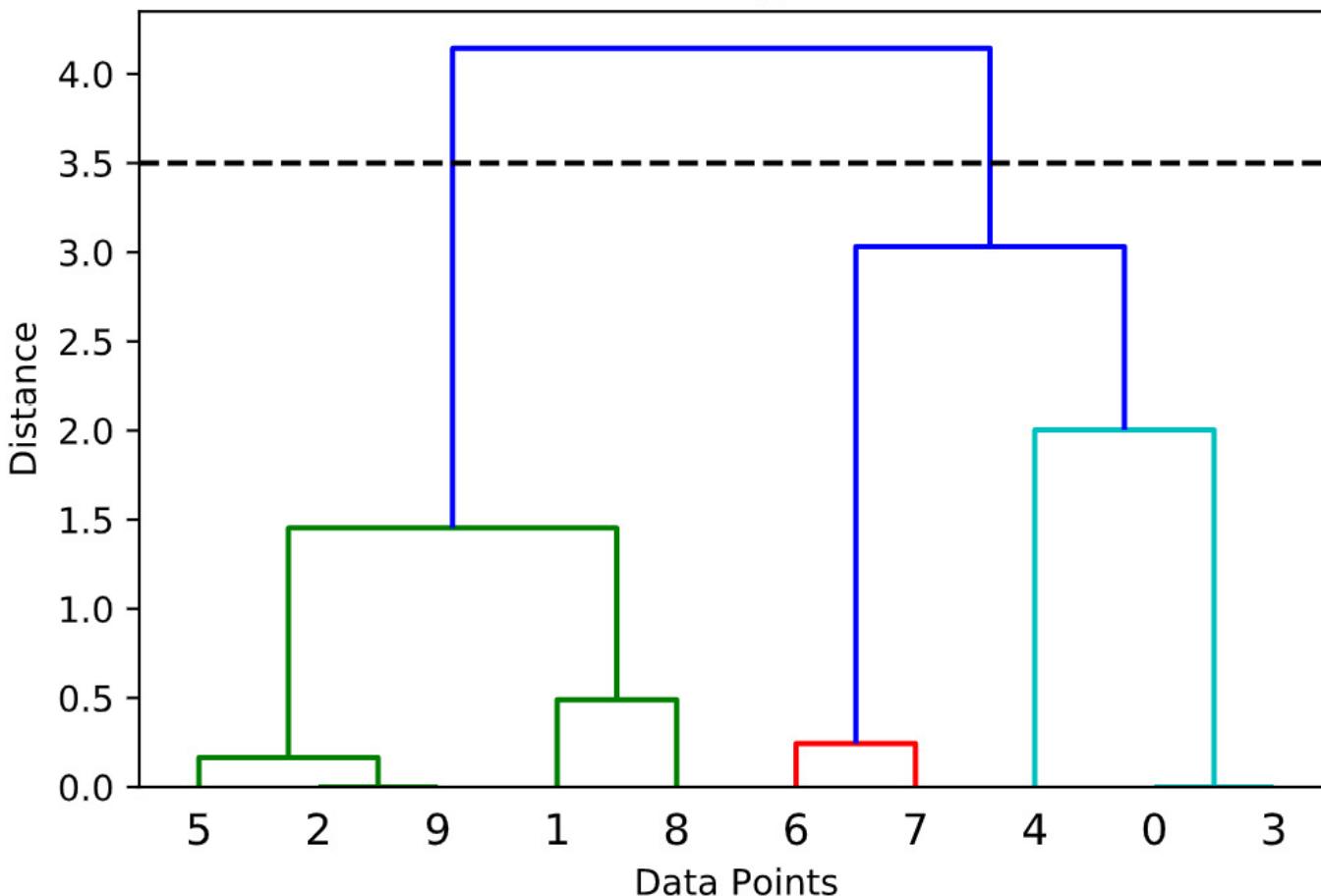
- Let's explore the different types of clustering. In particular, we will be talking about hierarchical and k-means clustering, and the different scenarios in which they should be used.
- However, before we dive into those, it's important to understand the concept of distance metrics, which is what we use to create clusters and identify similar data points.
- The most common distance metric is Euclidean, which is calculated as follows:

$$\text{Euclidean Distance}(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

# Hierarchical Clustering

- Hierarchical clustering algorithms group similar objects together to create a cluster with the help of a dendrogram. In this algorithm, we can vary the number of clusters as per our requirements.
- First, we construct a matrix consisting of distances between each pair of instances (data points).
- After that, we construct a dendrogram (a representation of clusters in the form of a tree) based on the distances between them.

# Dendrogram



# Hierarchical Clustering

In the preceding graph, we can perform a truncation at distance 3.5 to get two clusters or at 2.5 to get three clusters, depending on the requirements. To create a dendrogram using scikit-learn, we can use the following code:

```
import scipy.cluster.hierarchy as sch  
dendrogram = sch.dendrogram(sch.linkage(X,  
method='ward'))  
plt.title('Dendrogram')  
plt.show()
```

# Complete Exercise 3.01: Performing Hierarchical Clustering

# k-means Clustering

- In this algorithm, we segregate the given instances (data points) into "k" number of groups (here, k is a natural number).
- First, we choose k centroids. We assign each instance to its nearest centroid, thereby creating k groups.
- This is the assignment phase, which is followed by the update phase.

# k-means Clustering

- For example, suppose you have 10 documents. You want to group them into three categories based on their attributes, such as the number of words they contain, the number of paragraphs, punctuation, and the tone of the document.
- In this case, we will assume that  $k$  is 3; that is, we want to create these three groups.

# k-means Clustering

- The scikit-learn library can be used to perform k-means in Python using the following code:

```
from sklearn.cluster import KMeans  
kmeans = KMeans(n_clusters=4)  
kmeans.fit(X)  
clusters = kmeans.predict(X)
```

# Complete Exercise 3.02: Implementing k-means Clustering

# Supervised Learning

- Unlike unsupervised learning, supervised learning algorithms need labeled data.
- They learn how to automatically generate labels or predict values by analyzing various features of the data provided.
- For example, say you have already starred important text messages on your phone, and you want to automate the task of going through all your messages daily (considering they are important and marked already).

# Supervised Learning

This is a use case for supervised learning. Here, messages that have been starred previously can be used as labeled data. Using this data, you can create two types of models that are capable of the following:

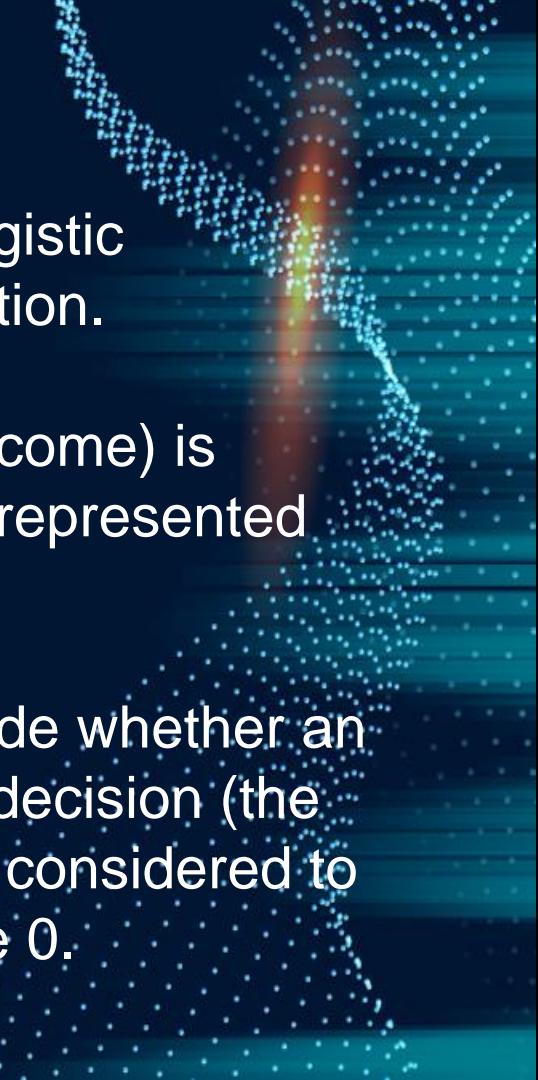
- Classifying whether new messages are important
- Predicting the probability of new messages being important

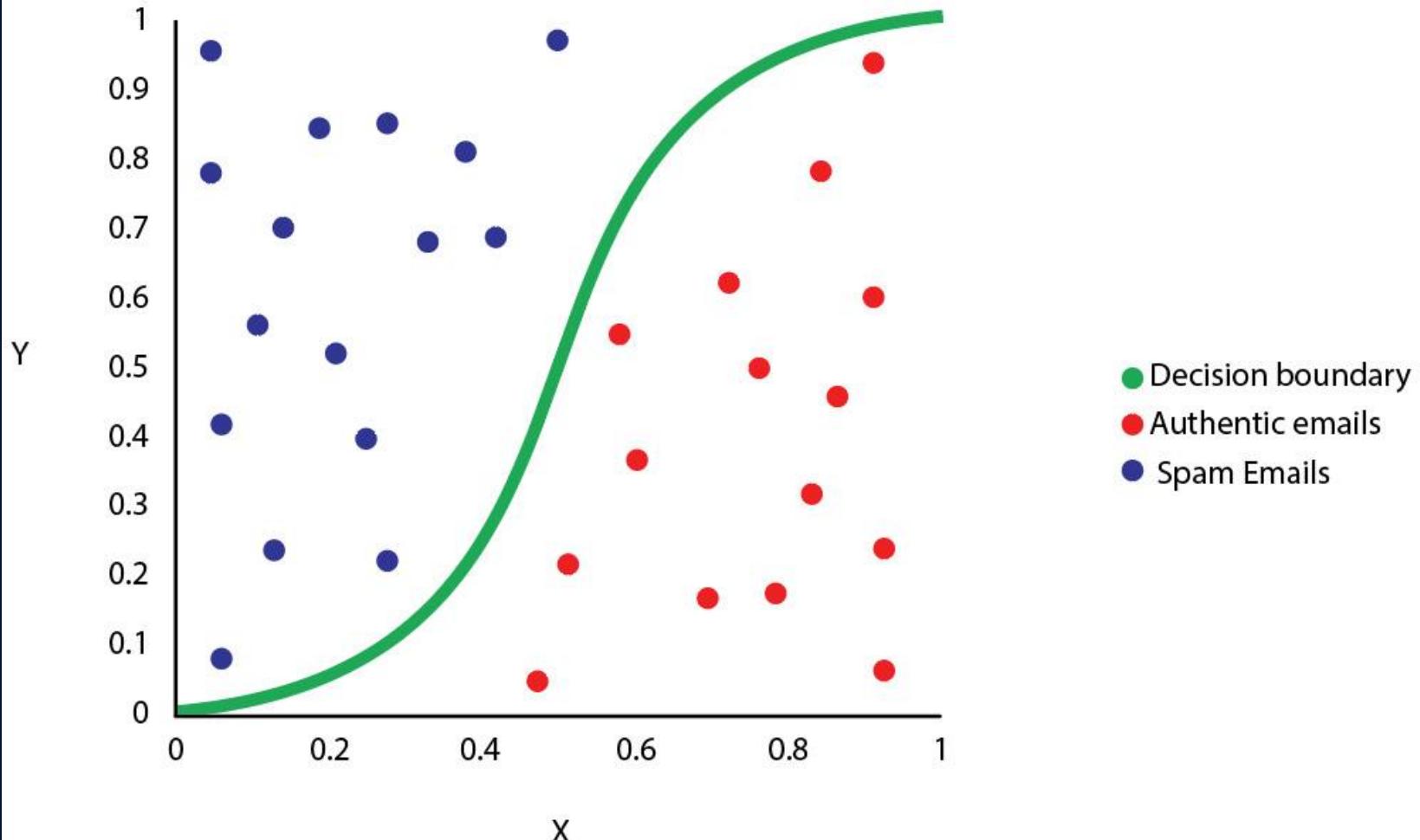
# Classification

- Say you have two types of food, of which type 1 tastes sweet and type 2 tastes salty, and you need to determine how an unknown food will taste using various attributes of the food (such as color, fragrance, shape, and ingredients). This is an instance of classification.
- Here, the two classes are class 1, which tastes sweet, and class 2, which tastes salty.

# Logistic Regression

- Despite having the term "regression" in it, logistic regression is used for probabilistic classification.
- In this case, the dependent variable (the outcome) is binary, which means that the values can be represented by 0 or 1.
- For example, consider that you need to decide whether an email is spam or not. Here, the value of the decision (the dependent variable, or the outcome) can be considered to be 1 if the email is spam; otherwise, it will be 0.





# Logistic Regression

- The scikit-learn library can be used to perform logistic regression in Python using the following code:

```
from sklearn.linear_model import LogisticRegression  
log_reg = LogisticRegression()  
log_reg.fit(X,y)  
predicted_labels = log_reg.predict(X)  
predicted_probability = log_reg.predict_proba(X)[:,1]
```

# Complete Exercise 3.03: Text Classification – Logistic Regression

# Naive Bayes Classifiers

- Just like logistic regression, a Naive Bayes classifier is another kind of probabilistic classifier.
- It is based on Bayes' theorem, which is shown here:

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}$$

# Naive Bayes Classifiers

- Similar to logistic regression, the scikit-learn library can be used to perform naïve Bayes classification and can be implemented in Python using the following code:

```
from sklearn.naive_bayes import GaussianNB  
nb = GaussianNB()  
nb.fit(X,y)  
predicted_labels = nb.predict(X)  
predicted_probability = nb.predict_proba(X)[:,1]
```

# Complete Exercise 3.04: Text Classification – Naive Bayes

# k-nearest Neighbors

- k-nearest neighbors is an algorithm that can be used to solve both regression and classification.
- In this lesson, we will focus on the classification aspect of the algorithm as it is used for NLP applications.
- Consider, for instance, the saying "birds of a feather flock together." This means that people who have similar interests prefer to stay close to each other and form groups.

# k-nearest Neighbors

- We can make use of the scikit-learn library to implement the k-nearest neighbors algorithm in Python using the following code:

```
from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier(n_neighbors=3)  
knn.fit(X,y)  
prediction = knn.predict(X)
```

# Complete Exercise 3.05: Text Classification Using the k-nearest Neighbors Method

# Regression

- To better understand regression, consider a practical example.
- For example, say you have photos of several people, along with a list of their respective ages, and you need to predict the ages of some other people from their photos. This is a use case for regression.

# Linear Regression

- The term "linear" refers to the linearity of parameters. Parameters are the coefficients of predictor variables in the linear regression equation.
- The following formula represents the linear regression equation:

$$y = \beta_0 + \beta_1 X + \epsilon$$

# Linear Regression

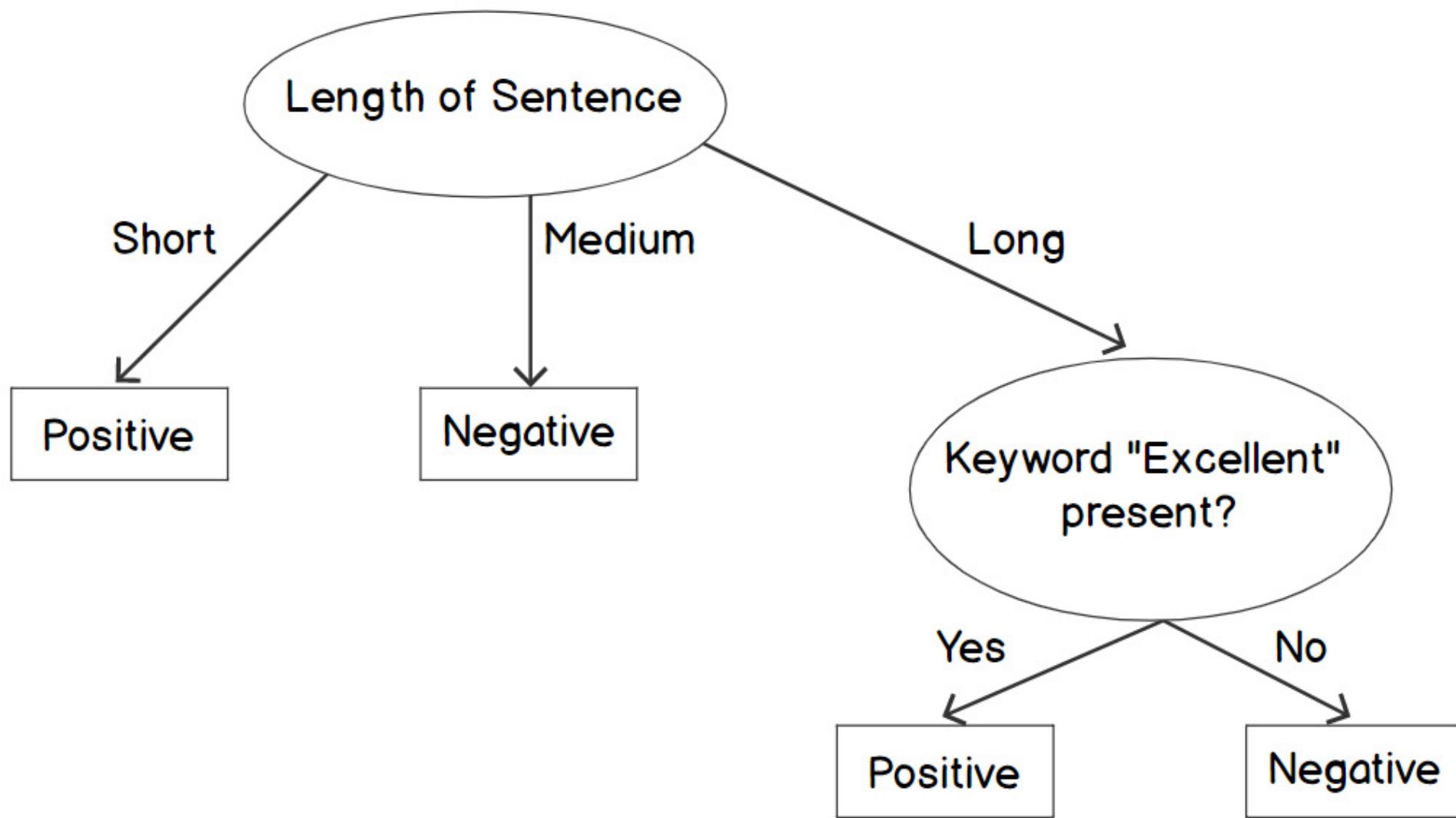
- We can use the scikit-learn library to implement the linear regression algorithm in Python with the following code:

```
from sklearn.linear_model import LinearRegression  
linreg = LinearRegression()  
linreg.fit(X,y)  
coefficient = linreg.coef_  
intercept = linreg.intercept_  
linreg.predict(X)
```

# **Complete Exercise 3.06: Regression Analysis Using Textual Data**

# Tree Methods

- There are several algorithms that have both classification and regression forms.
- Tree-based methods are instances of such cases.
- In the context of machine learning, "tree" refers to a structure that aids decision-making—hence, the term decision tree.



# Tree Methods

- We can make use of the scikit-learn library to implement the decision tree algorithm in Python using the following code:

```
from sklearn import tree  
dtc = tree.DecisionTreeClassifier()  
dtc = dtc.fit(X, y)  
predicted_labels = dtc.predict(X)
```

# Complete Exercise 3.07: Tree-Based Methods – Decision Tree

# Random Forest

- Imagine that you must decide whether to join a university.
- In one scenario, you ask only one person about the quality of the education the university provides.
- In another scenario, you ask several career counselors and academicians about this. Which scenario do you think would help you make a better and the most stable decision? The second one, right?

# Random Forest

- Random forests also aid in understanding the importance of predictor variables and features.
- However, building a random forest often takes a huge amount of time and memory.
- We can make use of the scikit-learn library to implement the random forest algorithm in Python using the following code:

```
from sklearn.ensemble import RandomForestClassifier  
rfc = RandomForestClassifier()  
rfc = rfc.fit(X, y)  
predicted_labels = rfc.predict(X)
```

# Gradient Boosting Machine and Extreme Gradient Boost

- There are various other tree-based algorithms, such as gradient boosting machines (GBM) and extreme gradient boosting (XGBoost). Boosting works by combining rough, less complex, or "weak" models into a single prediction that is more accurate than any single model.
- Iteratively, a subset of the training dataset is ingested into a "weak" algorithm or simple algorithm (such as a decision tree) to generate a weak model.

# Gradient Boosting Machine and Extreme Gradient Boost

The main reasons for the popularity of XGBoost are as follows:

- Ability to automatically deal with missing values
- High-speed execution
- High accuracy, if properly trained
- Support for distributed frameworks such as Hadoop and Spark

# Gradient Boosting Machine and Extreme Gradient Boost

- We can make use of the xgboost library to implement the XGBoost algorithm in Python using the following code:

```
from xgboost import XGBClassifier  
xgb_clf=XGBClassifier()  
xgb_clf = xgb_clf.fit(X, y)  
predicted_labels = rfc.predict(X)
```

# Gradient Boosting Machine and Extreme Gradient Boost

- The trained model can then be used to make predictions using the predict function.
- Here, X represents a DataFrame of independent variables, whereas y represents a DataFrame of dependent variables.
- To get the important features for the trained model, we can use the following code:

```
pd.DataFrame({'word':X.columns,'importance':xgb_clf.feature_importances_})
```

# Complete Exercise 3.08: Tree-Based Methods – Random Forest

# Complete Exercise 3.09: Tree-Based Methods – XGBoost

# Sampling

- Sampling is the process of creating a subset from a given set of instances.
- If you have 1,000 sentences in an article, out of which you choose 100 sentences for analysis, the subset of 100 sentences will be called a sample of the original article. This process is referred to as sampling.

# Sampling

There are different kinds of sampling methods, such as the following:

- Simple random sampling
- Stratified sampling
- Multi-Stage Sampling

# **Complete Exercise 3.10: Sampling (Simple Random, Stratified, and Multi-Stage)**

# Developing a Text Classifier

- A text classifier is a machine learning model that is capable of labeling texts based on their content.
- For instance, a text classifier will help you understand whether a random text statement is sarcastic or not.
- Presently, text classifiers are gaining importance as manually classifying huge amounts of text data is impossible.
- In the next few sections, we will learn about the different parts of text classifiers and implement them in Python.

# Feature Extraction

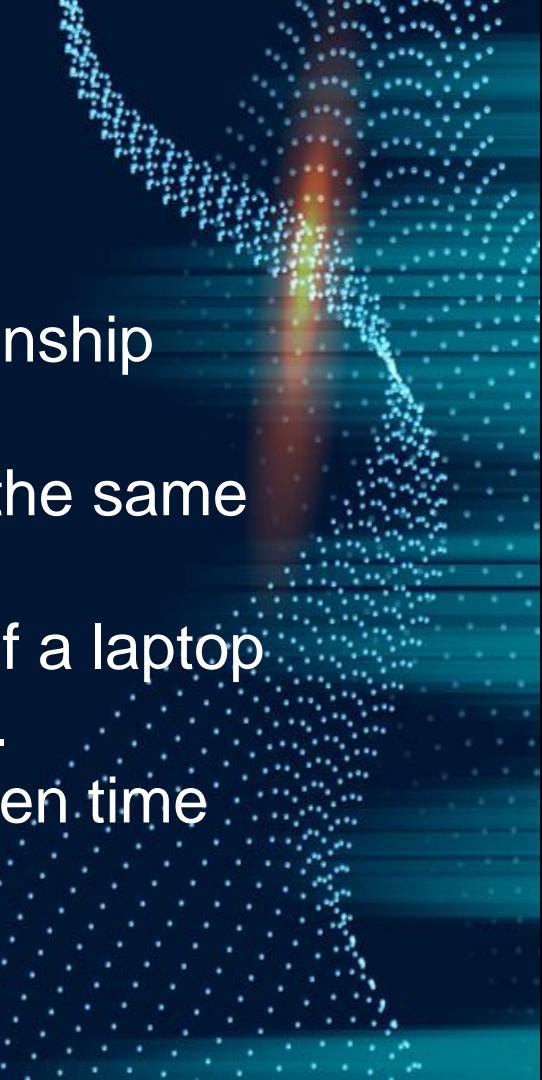
- When dealing with text data, features denote its different attributes.
- Generally, they are numeric representations of the text.
- As we discussed in lesson 2, Feature Extraction Methods, TFIDF representations of texts are one of the most popular ways of extracting features from them.

# Feature Engineering

- Feature engineering is the art of extracting new features from existing ones.
- Extracting novel features, which tend to capture variation in data better, requires sound domain expertise.

# Removing Correlated Features

- Correlation refers to the statistical relationship between two variables.
- Two highly correlated variables provide the same kind of information.
- For example, the remaining battery life of a laptop and its screen time are highly correlated.
- The battery life will decrease as the screen time increases.



# Removing Correlated Features

- Regression models, including logistic regression, are unable to perform well when correlation between features exists.
- Thus, features with correlation beyond a certain threshold need to be removed.
- The most widely used correlation statistic is Pearson correlation, which can be calculated as follows:

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y}$$

# Complete Exercise 3.11: Removing Highly Correlated Features (Tokens)

# Dimensionality Reduction

- There are some optional steps that we can follow on a case-to-case basis.
- For example, sometimes, the TFIDF matrix or Bag-of-Words representation of a text corpus is so big that it doesn't fit in memory.
- In this case, it would be necessary to reduce its dimension—that is, the number of columns in the feature matrix.
- The most popular method for dimension reduction is Principal Component Analysis (PCA).

# Complete Exercise 3.12: Performing Dimensionality Reduction Using Principal Component Analysis

# Deciding on a Model Type

- Once the feature set is ready, it's necessary to decide on the type of model that will be used to deal with the problem. Usually, unsupervised models are chosen when data is not labeled.
- If we have a predefined number of clusters in mind, we go for clustering algorithms such as k-means; otherwise, we opt for hierarchical clustering.
- For labeled data, we generally follow supervised learning methods such as regression and classification.

# Evaluating the Performance of a Model

- Once a model is ready, it is necessary to evaluate its performance.
- This is because, without benchmarking it, we cannot be confident of how well or how badly it is functioning.
- It is not advisable to put a model into production without evaluating its efficiency.
- There are various ways to evaluate a model's performance.

# Confusion Matrix

		Predicted 0	Predicted 1
Actual 0	Actual 0	<b>TN</b>	<b>FP</b>
	Actual 1	<b>FN</b>	<b>TP</b>

# Confusion Matrix

- In the preceding confusion matrix, the top-left cell will have the count of all correctly classified 0 values by the classifier, whereas the top-right cell will have the count of incorrectly classified 0 values, and so on.
- To create a confusion matrix using Python, you can use the following code:

```
from sklearn.metrics import confusion_matrix  
confusion_matrix(actual_values,predicted_values)
```

# Accuracy

- Accuracy is defined as the ratio of correctly classified instances to the total number of instances.
- Whenever accuracy is used for model evaluation, we need to ensure that the data is balanced in terms of classes, meaning it should have an almost equal number of instances of each class.

# Accuracy

- To get the accuracy of the predicted values using Python, you can use the following code:

```
from sklearn.metrics import accuracy_score  
accuracy_score(actual_values,predicted_values)
```

# Precision and Recall

- For a better understanding of precision and recall, let's consider a real-life example.
- If your mother tells you to explore the kitchen of your house, find items that need to be restocked, and bring them back from the market, you will bring P number of items from the market and show them to your mother.
- Out of P items, she finds Q items to be relevant.

# F1 Score

- For a given classification model, the F1 score is the harmonic mean of precision and recall.
- Harmonic mean is a way to find the average while giving equal weight to all numbers:

$$\text{F1 score} = 2 * ((\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}))$$

# F1 Score

- To get the F1 score, precision, and recall values using Python, you can use the following code:

```
from sklearn.metrics import classification_report  
classification_report(actual_values,predicted_values)
```

# Receiver Operating Characteristic (ROC) Curve

- To understand the ROC curve, we need to get acquainted with the True Positive Rate (TPR) and the False Positive Rate (FPR):

$\text{TPR} = \text{True Positive} / (\text{True Positive} + \text{False Negative})$

$\text{FPR} = \text{False Positive} / (\text{False Positive} + \text{True Negative})$

# Receiver Operating Characteristic (ROC) Curve

- The maximum value of AUROC is 1. To create the ROC curve using Python, use the following code:

```
fpr,tpr,threshold=roc_curve(actual_values,\n                            predicted_probabilities)\nprint ('\nArea under ROC curve for validation set:', auc(fpr,tpr))\nfig, ax = plt.subplots(figsize=(6,6))\nax.plot(fpr,tpr,label='Validation set AUC')\nplt.xlabel('False Positive Rate')\nplt.ylabel('True Positive Rate')\nax.legend(loc='best')\nplt.show()
```

# Root Mean Square Error (RMSE)

- This is mainly used for evaluating the accuracy of regression models. We define it as follows

$$\text{RMSE} = \sqrt{\frac{\sum (P_i - O_i)^2}{n}}$$

# Root Mean Square Error (RMSE)

- Here, n is the number of samples,  $P_i$  is the predicted value of the  $i$ th observation, and  $O_i$  is the observed value of the  $i$ th observation.
- To find the RMSE using Python, use the following code:

```
from sklearn.metrics import mean_squared_error  
rmse = sqrt(mean_squared_error(y_actual,  
y_predicted))
```

# Maximum Absolute Percentage Error (MAPE)

- Just like RMSE, this is another way to evaluate a regression model's performance. It is described as follows:

$$\text{MAPE} = \left( \frac{1}{n} \sum \frac{|O_i - P_i|}{|O_i|} \right) * 100 \text{ for all } i \text{ from } 1 \text{ to } n$$

# Maximum Absolute Percentage Error (MAPE)

- Here, n is the number of samples,  $P_i$  is the predicted value (that is, the forecast value) of the  $i$ th observation, and  $O_i$  is the observed value (that is, the actual value) of the  $i$ th observation.
- To find MAPE in Python, use the following code:

```
from sklearn.metrics import mean_absolute_error  
mape = mean_absolute_error(y_actual, y_predicted) * 100
```

# Complete Exercise 3.13: Calculating the RMSE and MAPE of a Dataset

# Complete Activity 3.01: Developing End-to-End Text Classifiers

# Building Pipelines for NLP Projects

- In general, a pipeline refers to a structure that allows a streamlined flow of air, water, or something similar.
- In this context, pipeline has a similar meaning.
- It helps to streamline various stages of an NLP project.
- An NLP project is done in various stages, such as tokenization, stemming, feature extraction (TFIDF matrix generation), and model building.

# Complete Exercise 3.14: Building the Pipeline for an NLP Project

# Saving and Loading Models

- After a model has been built and its performance matches our expectations, we may want to save it for future use.
- This eliminates the time needed for rebuilding it.
- Models can be saved on the hard disk using the joblib and pickle libraries.

# Saving and Loading Models

The pickle module makes use of binary protocols to save and load Python objects. joblib makes use of the pickle library protocols, but it improves on them to provide an efficient replacement to save large Python objects. Both libraries have two main functions that we will make use of to save and load our models:

- **dump:** This function is used to save a Python object to a file on the disk.
- **loads:** This function is used to load a saved Python object from a file on the disk.

# Complete Exercise 3.15: Saving and Loading Models

# Summary

- In this lesson, you learned about the different types of machine learning techniques, such as supervised and unsupervised learning.
- We explored unsupervised algorithms such as hierarchical clustering and k-means clustering, and supervised learning algorithms, such as k-nearest neighbor, the Naive Bayes classifier, and tree-based methods, such as random forest and XGBoost, that can perform both regression and classification.

# 4. Collecting Text Data with Web Scraping and APIs

# Overview

- This lesson introduces you to the concept of web scraping.
- You will first learn how to extract data (such as text, images, lists, and tables) from pages that are written using HTML.
- You will then learn about the various types of semi-structured data used to create web pages (such as JSON and XML) and extract data from them.

# Introduction

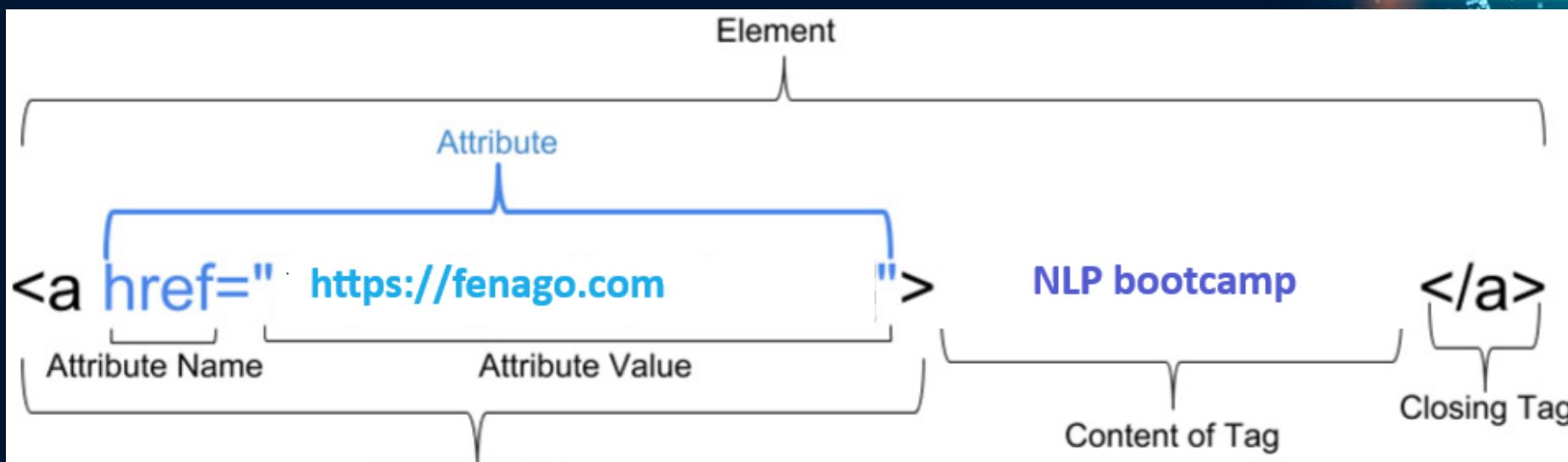
- In the last lesson, we developed a simple classifier using feature extraction methods.
- We also covered different algorithms that fall under supervised and unsupervised learning.
- In this lesson, you will learn how to collect text data by scraping web pages, and then you will learn how to process that data

# Collecting Data by Scraping Web Pages

- The basic building block of any web page is HTML (Hypertext Markup Language)—a markup language that specifies the structure of your content.
- HTML is written using a series of tags, combined with optional content.
- The content encompassed within HTML tags defines the appearance of the web page.
- It can be used to make words bold or italicize them, to add hyperlinks to the text, and even to add images.

# Collecting Data by Scraping Web Pages

- The following figure depicts the contents that are included within an HTML tag



# Collecting Data by Scraping Web Pages

The basic HTML structure and commonly used tags are shown and explained as follows:

```
1 <!DOCTYPE html>
2 ▼ <html>
3 ▼   <head>
4     <meta charset="utf-8">
5 ▼     <title>My Page</title>
6   </head>
7 ▼   <body>
8 ▼     <p>Hello, World!</p>
9   </body>
10  </html>
```

# **Complete Exercise 4.01: Extraction of Tag-Based Information from HTML Files**

# Requesting Content from Web Pages

- Whenever you visit a web page from your web browser, you are actually sending a request to fetch its content.
- This can be done using Python scripts. The Python requests package is widely used to handle all forms of HTTP requests.
- Let's walk through an exercise to get a better understanding of this concept.

# Complete Exercise 4.02: Collecting Online Text Data

# Complete Exercise 4.03: Analyzing the Content of Jupyter Notebooks (in HTML Format)

# Complete Activity 4.01: Extracting Information from an Online HTML Page

# Complete Activity 4.02: Extracting and Analyzing Data Using Regular Expressions

# Dealing with Semi-Structured Data

- We learned about various types of data in lesson 2, Feature Extraction Methods.
- Let's quickly recapitulate what semi-structured data refers to.
- A dataset is said to be semi-structured if it is not in a row-column format but, if required, can be converted into a structured format that has a definite number of rows and columns.

# JSON

- JSON files are used for storing and exchanging data. JSON is human-readable and easy to interpret. Just like text files and CSV files, JSON files are language-independent.
- This means that different programming languages, such as Python, Java, and so on, can work with JSON files effectively.

# JSON

The datatype of values of JSON objects must be any of the following:

- A string
- A number
- Another JSON object
- An array
- A boolean
- Null

```
{  
  "stones": [  
    {  
      "name": "Space Stone",  
      "movies": ["Thor", "Captain America", "The  
        Avengers"]  
    },  
    {  
      "name": "Mind Stone",  
      "movies": ["The Avengers", "The Winter Soldier",  
        "Age of Ultron", "Civil War"]  
    },  
    {  
      "name": "Reality Stone",  
      "movies": ["The Dark World"]  
    },  
    {  
      "name": "Power Stone",  
      "movies": ["Guardians of the Galaxy"]  
    },  
    {  
      "name": "Time Stone",  
      "movies": ["Dr. Strange"]  
    },  
    {  
      "name": "Soul Stone"  
    }  
  ]  
}
```

# Complete Exercise 4.04: Working with JSON Files

# XML

- Just like HTML, XML is another kind of markup language that stores data in between tags. It is human-readable and extensible; that is, we have the liberty to define our own tags.
- Attributes, elements, and tags in the case of XML are similar to those of HTML.
- An XML file may or may not have a declaration. But, if it has a declaration, then that must be the first line of the XML file.

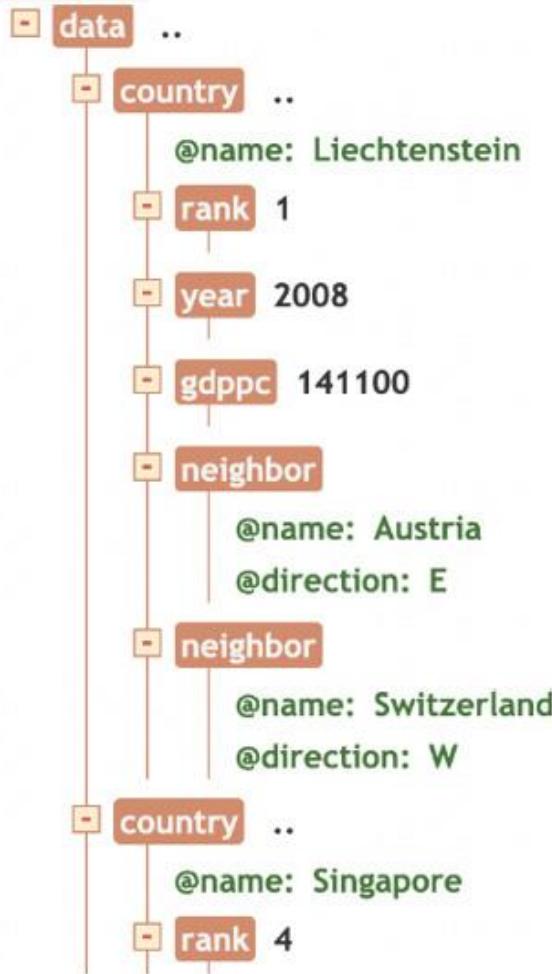
```
<?xml version="1.0"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>
```



```

<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>

```



# Complete Exercise 4.05: Working with an XML File

# Using APIs to Retrieve Real-Time Data

- API stands for Application Programming Interface. To understand what an API is, let's consider a real-life example.
- Suppose you have a socket plug in the wall, and you need to charge your cellphone using it.
- How will you do it? You will have to use a charger/adapter, which will enable you to connect the cellphone to the socket

# Complete Exercise 4.06: Collecting Data Using APIs

# Extracting data from Twitter Using the OAuth API

- Many popular websites, such as Twitter, provide an API that allows access to parts of their services so that people can build software that integrates with the website.
- We'll be focusing mainly on Twitter in this section.
- Twitter's data and services (such as tweets, advertisements, direct messages, and much more) can be accessed via the Twitter API.

# Extracting data from Twitter Using the OAuth API

- You can collect data from Twitter using their Python module, named Tweepy. Tweepy is a Python library for accessing the Twitter API.
- It is great for simple automation and creating Twitter bots.
- It provides abstraction to communicate with Twitter and use its API to ease interactions, which makes this approach more efficient than using the requests library and Twitter API endpoints.

# Extracting data from Twitter Using the OAuth API

- Your Python code should look like this:

```
import tweepy  
consumer_key = 'your consumer key here'  
consumer_secret = 'your consumer secret key here'  
access_token = 'your access token here'  
access_token_secret = 'your access token secret here'  
auth = tweepy.OAuthHandler(consumer_key,  
consumer_secret)  
auth.set_access_token(access_token, access_token_secret)  
api = tweepy.API(auth)
```

# Extracting data from Twitter Using the OAuth API

- To search for a query named randomquery using tweepy, you can use the Cursor object as follows:

```
tweepy.Cursor(api.search, q='randomquery', lang="en")
```

# Complete Activity 4.03: Extracting Data from Twitter

# Summary

- In this lesson, we have learned various ways to collect data by scraping web pages.
- We also successfully scraped data from semi-structured formats such as JSON and XML and explored different methods of retrieving data in real time from a website without authentication.

# 5. Topic Modeling

# Overview

- This lesson introduces topic modeling, which means using unsupervised machine learning to find "topics" within a given set of documents.
- You will explore the most common approaches to topic modeling, which are Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), and the Hierarchical Dirichlet Process (HDP), and learn the differences between them.

# Introduction

- In the previous lesson, we learned about different ways to collect data from local files and online resources.
- In this lesson, we will focus on topic modeling, which is an important area within natural language processing.
- Topic modeling is a simple way to capture the sense of what a document or a collection of documents is about.

# Introduction

- Topic modeling generally uses unsupervised learning algorithms, as opposed to supervised learning algorithms.
- This means that, during training, we do not have to provide labels (that is, topic names corresponding to each document) in order to teach the model.
- This not only helps us discover interesting topics that might exist, but also reduces the manual effort spent in labeling texts

# Topic Discovery

- The main goal of topic modeling is to find a set of topics that can be used to classify a set of documents.
- These topics are implicit because we do not know what they are beforehand, and they are unnamed.
- The number of topics could vary from around 3 to, say, 400 (or even more) topics. Since it is the algorithm that discovers the topics, the number is generally fixed as an input to the algorithm, except in the case of non-parametric models in which the number of topics is inferred from the text.

# Exploratory Data Analysis

- It is recommended to do exploratory data analysis prior to performing any machine learning project.
- This helps you learn about the probability distribution of the items in the dataset.
- We have seen this with word clouds in lesson 2, Feature Extraction Methods.

# Transforming Unstructured Data to Structured Data

- Topic modeling clusters documents based on their topics. Specifically, it is a soft clustering method, as each document gets mapped to multiple topics.
- This is unlike hard clustering, which results in membership of an exemplar or a point of only one cluster.
- Topic models typically give a weight/probability of the document being associated with a topic.

# Bag of Words

- Before we explore modeling algorithms in depth, let's make a few simplifying assumptions.
- Firstly, we treat a document as a bag of words, meaning we ignore the structure and grammar of the document and just use the count of each word in the document to infer patterns in the variation of word counts.
- Ignoring the structure, sequences, and grammar allows us to use algorithms that rely on counts and probability to make the inferences.

# Topic-Modeling Algorithms

Topic-modeling algorithms operate on the following assumptions:

- Topics contain a set of words.
- Documents are made up of a set of topics.

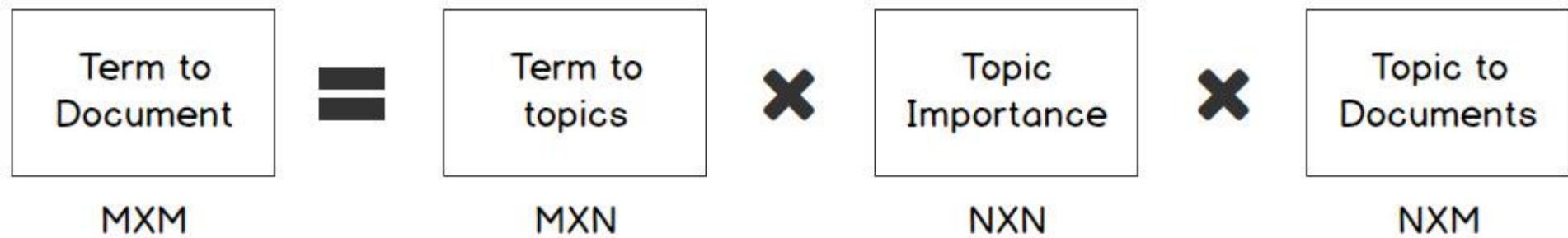
# Latent Semantic Analysis (LSA)

- We will start by looking at LSA. LSA actually predates the World Wide Web. It was first described in 1988.
- LSA is also known by an alternative name, Latent Semantic Indexing (LSI), particularly when it is used for semantic searches of document indices.
- The goal of LSA is to uncover the latent topics that underlie documents and words.

# LSA – How It Works

Term to Document	Doc-1	Doc-1	Doc-3
Water			
Dog			
Willow			
Cart			
Pill			
Stone			

# LSA – How It Works



# LSA – How It Works

- As we can see in this diagram, the rectangular matrix is separated into the product of other matrices.
- The process takes a matrix, M, and splits it, as shown in the following formula:

$$M = U\Sigma V^T$$

# Key Input Parameters for LSA Topic Modeling

- We will be using the gensim library to perform LSA topic modeling.
- The key input parameters for gensim are corpus, the number of topics, and id2word.
- Here, the corpus is specified in the form of a list of documents in which each document is a list of tokens.
- The id2word parameter refers to a dictionary that is used to convert the corpus from a textual representation to a numeric representation such that each word corresponds to a unique number

# Key Input Parameters for LSA Topic Modeling

- We will be using spaCy v2.1.3. After installing spaCy v2.1.3 we will need to download the English language model using the following code, so that we can load this model (since there are models for many different languages).

```
python -m spacy download en_core_web_sm
```

# **Complete Exercise 5.01: Analyzing Wikipedia World Cup Articles with Latent Semantic Analysis**

# Dirichlet Process and Dirichlet Distribution

- A Dirichlet process is a distribution over a distribution. It can be represented as  $DP(\alpha, G)$  where  $G$  is the base distribution and  $\alpha$  is the concentration parameter that defines how close  $DP(\alpha, G)$  is to the base distribution  $G$ .
- It is for this reason that the Dirichlet process is a versatile way to represent various probability distributions. It is used for the HDP topic-modeling algorithm.

# Latent Dirichlet Allocation (LDA)

- Instead of using matrix factorization, like we did for LSA, it is possible to consider a generative model called LDA.
- LDA is considered an advancement over probabilistic LSA. Probabilistic LSA is prone to overfitting as it does not probabilistically model the distribution of the documents.

# LDA – How It Works

	<b>Cat</b>	<b>Dog</b>	<b>Hippo</b>
Document 1	10	0	0
Document 2	0	10	0
Document 3	0	0	10
Document 4	10	10	10

# LDA – How It Works

Words vs Topics

	Topic 1	Topic 2	Topic 3
Cat	0.00	0.00	0.99
Dog	0.99	0.00	0.00
Hippo	0.00	0.99	0.00

Documents vs Topics

	Topic 1	Topic 2	Topic 3
Document 1	0.030	0.030	0.939
Document 2	0.939	0.030	0.030
Document 3	0.030	0.939	0.030
Document 4	0.33	0.33	0.33

# LDA – How It Works

The parameters that we use for tomotopy are as follows:

- corpus: This refers to text that we want to analyze.
- Number of topics: This is the number of topics that the corpus contains.
- iter: This refers to the number of iterations that the model considers the corpus.
- $\alpha$ : This is associated with document generation.
- $\eta$ : This is associated with topic generation.
- seed: This helps with fixing the initial randomization.

# Measuring the Predictive Power of a Generative Topic Model

- The predictive power of a generative topic model can be measured by analyzing the distribution of the generated corpus.
- Perplexity is a measure of how close the distribution of the words in the generated corpus is to reality.
- Log perplexity is a more convenient measure for this closeness. The formula for log perplexity is as

$$\log \text{perplexity} = -\frac{1}{n} \sum_{k=0}^n \log P(w)$$

# Complete Exercise 5.02: Finding Topics in Canadian Open Data Inventory Using the LDA Model

# Complete Activity 5.01: Topic-Modeling Jeopardy Questions

# Hierarchical Dirichlet Process (HDP)

- HDP is a non-parametric variant of LDA. It is called "non-parametric" since the number of topics is inferred from the data, and this parameter isn't provided by us.
- This means that this parameter is learned and can increase (that is, it is theoretically unbounded).
- The tomotopy HDP implementation can infer between 1 and 32,767 topics. gensim's HDP implementation seems to fix the number of topics at 150 topics.

# **Complete Exercise 5.03: Topics in Around the World in Eighty Days**

# **Complete Exercise 5.04: Topics in The Life and Adventures of Robinson Crusoe by Daniel Defoe**

# Practical Challenges

- The selection of the number of topics and topic-modeling algorithms, the number of iterations, and the evaluation of the topic model are the main challenges faced by a practitioner.
- Having prior knowledge about the domain can greatly help in choosing the number of topics.

# State-of-the-Art Topic Modeling

- There is no known benchmark for quantitatively identifying the state-of-the-art topic-modeling algorithm.
- It necessarily involves human participation whenever interpretable topics are required.
- In cases where the interpretation of topics is not necessary, the topic model needs to be evaluated by downstream tasks.

# Complete Activity 5.02: Comparing Different Topic Models

# Summary

- In this lesson, we discussed topic modeling in detail.
- Without delving into advanced statistics, we reviewed various topic-modeling algorithms (such as LSA, LDA, and HDP) and how they can be used for topic modeling on a given dataset.
- We explored the challenges involved in topic modeling, how experimentation can help address those challenges, and, finally, broadly discussed the current state-of-the-art approaches to topic modeling.

# 6. Vector Representation

# Overview

- This lesson introduces you to the various ways in which text can be represented in the form of vectors.
- You will start by learning why this is important, and the different types of vector representation.
- You will then perform one-hot encoding on words, using the preprocessing package provided by scikit-learn, and character-level encoding, both manually and using the powerful Keras library.

# Introduction

- The previous lessons laid a firm foundation for NLP. But now we will go deeper into a key topic—one that gives us surprising insights into how language processing works and how some of the key advances in human-computer interaction are facilitated.
- At the heart of NLP is the simple trick of representing text as numbers.
- This helps software algorithms perform the sophisticated computations that are required to understand the meaning of the text.

# What Is a Vector?

- The basic mathematical definition of a vector is an object that has both magnitude and direction.
- In our definition, it is mostly compared with a scalar, which can be defined as an object that has only magnitude.
- Vectors are also defined as an element in vector space—for example, a point in space with the coordinates ( $x=4$ ,  $y=5$ ,  $z=6$ ) is a vector.

# What Is a Vector?

<b>Weather</b>	<b>Place 1</b>	<b>Place 1</b>
Temperature	25	32
Humidity	50	60
Precipitation	1	0
Wind speed	18	7
Air pressure	1200.0	1019.0

# What Is a Vector?

So, we can represent the weather of these places in vector form as follows:

- Vector for place 1: [25, 50, 1, 18, 1200.0]
- Vector for place 2: [32, 60, 0, 7, 1019.0]

These techniques can be broadly classified into two categories:

- Frequency-based embeddings
- Learned word embeddings

# Frequency-Based Embeddings

Frequency-based embedding is a technique in which the text is represented in vector form by considering the frequency of the word in a corpus. The techniques that come under this category are the following:

- Bag of words
- TFIDF
- Term frequency-based technique

# Complete Exercise 6.01: Word-Level One-Hot Encoding

# Character-Level One-Hot Encoding

- In character-level one-hot encoding, we assign a numeric value to all the possible characters.
- We can use alpha-numeric characters and punctuation as well.
- Then, we represent each character by an array of size equal to all the characters in the document.
- This array contains zero at all the indices, other than the index assigned with the character.

# Complete Exercise 6.02: Character One-Hot Encoding – Manual

# Complete Exercise 6.03: Character-Level One-Hot Encoding with Keras

# Learned Word Embeddings

The vector representations discussed in the preceding section have some serious disadvantages, as discussed here:

- Sparsity and large size
- Context

# Word2Vec

- Word2Vec is a prediction-based algorithm that represents a word by a vector of a fixed size.
- This is a form of unsupervised learning algorithm, which means that we need not to provide manually annotated data; we just feed the raw text.
- It will train a model in such a way that each word is represented in terms of its context throughout the training data.

# Word2Vec

This algorithm has two variations, as follows:

- **Continuous Bag of Words (CBoW):** This model tends to predict the probability of a word given the context. The learning problem here is to predict the word given a fixed-window context—that is, a fixed set of continuous words in text.
- **Skip-Gram model:** This model is the reverse of the CBoW model, as it tends to predict the context of a word.

# Word2Vec

In order to perform basic text cleaning, before it's processed, we will make use of the `textcleaner` class from `gensim`. Some of the most useful functions available in `textcleaner` that we will be using are as follows:

- **`split_sentences()`**: As the name suggests, this function splits the text and gets a list of sentences from the text.
- **`simple_preprocess()`**: This function converts a document into a list consisting of lowercase tokens.

# Complete Exercise 6.04: Training Word Vectors

# Using Pre-Trained Word Vectors

- For a machine learning model, the more data you have, the better the model you get.
- But training the model on large amounts of data is intensively resource-consuming in terms of both time and memory.
- So, we usually train a Word2Vec model on a large amount of data and retain the model for future use.
- There are also a lot of pre-trained models publicly available have been trained on huge datasets such as Wikipedia articles.

# Complete Exercise 6.05: Using Pre-Trained Word Vectors

# Document Vectors

- Word vectors and word embeddings represent words. But if we wanted to represent a whole document, we'd need to use document vectors.
- Note that when we refer to a document, we are referring to a collection of words that have some meaning to a user.
- A document can be a single sentence or a group of sentences.

# Uses of Document Vectors

Some of the uses of document vectors are as follows:

- **Similarity:** We can use document vectors to compare texts for similarity. For example, legal AI software can use document vectors to find similar legal cases.
- **Recommendations:** For example, online magazines can recommend similar articles based on those that users have already read.
- **Predictions:** Document vectors can be used as input into machine learning algorithms to build predictive models.

# Complete Exercise 6.06: Converting News Headlines to Document Vectors

# **Complete Activity 6.01: Finding Similar News Article Using Document Vectors**

# Summary

- In this lesson, we learned about the motivations behind converting human language in the form of text into vectors.
- This helps machine learning algorithms to execute mathematical functions on the text, detect patterns in language, and gain an understanding of the meaning of the text.

# 7. Text Generation and Summarization

# Overview

- This lesson begins with the concept of text generation using Markov chains, before moving on to two types of text summarization—namely, abstractive and extractive summarization.
- You will then explore the TextRank algorithm and use it with different datasets.

# Introduction

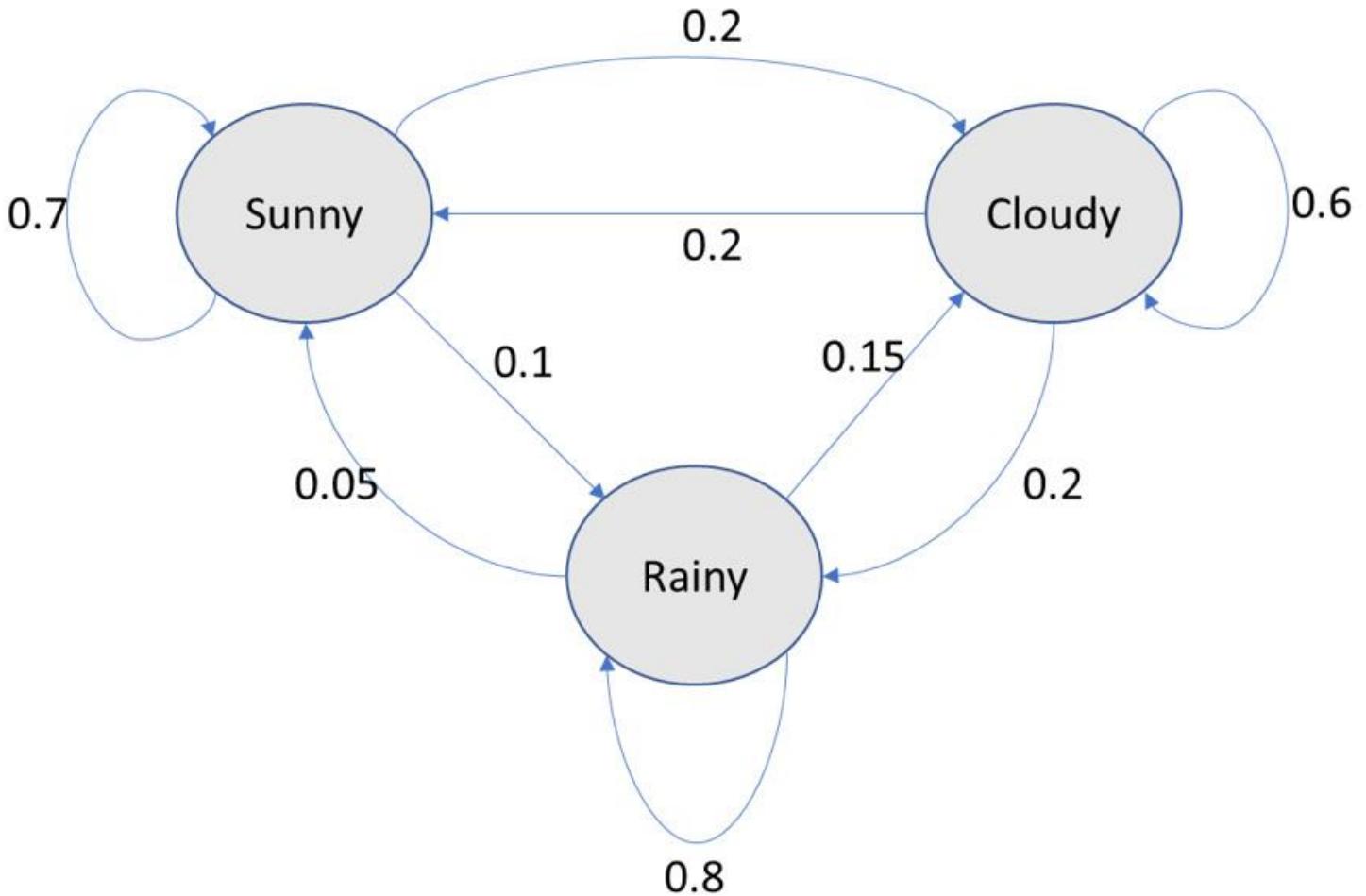
- The ability to express thoughts in words (sentence generation), the ability to replace a piece of text with different but equivalent text (paraphrasing), and the ability to find the most important parts of a piece of text (summarization) are all key elements of using language.
- Although sentence generation, paraphrasing, and summarization are challenging tasks in NLP, there have been great strides recently that have made them considerably more accessible

# Generating Text with Markov Chains

- An idea is expressed using the words of a language. As ideas are not tangible, it is useful to look at text generation in order to gauge whether a machine can think on its own.
- The utility of text generation is currently limited to an auto-complete functionality, besides a few negative use cases that we will discuss later in this section.

# Markov Chains

- A state space defines all possible states that can exist.
- A Markov chain consists of a state space and a specific type of successor function.
- For example, in the case of the simplified state space to describe the weather, the states could be Sunny, Cloudy, or Rainy.
- The successor function describes how a system in its current state can move to a different state or even continue in the same state



# **Complete Exercise 7.01: Text Generation Using a Random Walk over a Markov Chain**

# Text Summarization

- Automated text summarization is the process of using NLP tools to produce concise versions of text that preserve the key information present in the original content.
- Good summaries can communicate the content with less text by retaining the key information while filtering out other information and noise (or useless text, if any).
- A shorter text may often take less time to read, and thus summarization facilitates more efficient use of time.

# TextRank

- TextRank is a graph-based algorithm (developed by Rada Mihalcea and Paul Tarau) used to find the key sentences in a piece of text.
- As we already know, in graph theory, a graph has nodes and edges.
- In the TextRank algorithm, we estimate the importance of each sentence and create a summary with the sentences that have the highest importance.

# TextRank

The TextRank algorithm works as follows:

- Represent a unit of text (say, a sentence) as a node.
- Each node is given an arbitrary importance score.

# TextRank

- Each edge has a weight that corresponds to the similarity between two nodes (for instance, the sentences  $S_x$  and  $S_y$ ).
- The weight could be the number of common words (say,  $w_k$ ) in the two sentences divided by the sum of the number of words in the two sentences. This can be represented as follows:

$$\text{Similarity}(S_x, S_y) = \frac{|(W_k : W_k \in S_x \ \& \ W_k \in S_y)|}{\log |S_x| + \log |S_y|}$$

- For each node, we compute a new importance score, which is a function of the importance score of the neighboring nodes and the edge weights ( $w_{ji}$ ) between them.
- Specifically, the function ( $f$ ) could be the edge-weighted average score of all the neighboring nodes that are directed toward that node that is adjusted by all the outward edge weights ( $w_{jk}$ ) and the damping factor ( $d$ ). This can be represented as follows:

$$f(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} f(V_j)$$

# Key Input Parameters for TextRank

We'll be using the gensim library to implement TextRank. The following are the parameters required for this:

- **text:** This is the input text.
- **ratio:** This is the required ratio of the number of sentences in the summary to the number of sentences in the input text.

# **Complete Exercise 7.02: Performing Summarization Using TextRank**

# Complete Exercise 7.03: Summarizing a Children's Fairy Tale Using TextRank

# **Complete Activity 7.01: Summarizing Complaints in the Consumer Financial Protection Bureau Dataset**

# Recent Developments in Text Generation and Summarization

- Alan Turing (for whom the equivalent of the Nobel Prize in Computer Science is named) proposed a test for artificial intelligence in 1950.
- This test, known as the Turing Test, says that if humans ask questions and cannot distinguish between text responses generated by a machine and a human, then that machine can be deemed to be intelligent.

# Recent Developments in Text Generation and Summarization

- BERT is a transformer that learns the sequential structure of a text in both directions, that is, from left to right and from right to left.
- This is achieved by randomly masking the words while the model is trained, much like how children are often taught a language by using fill-in-the-blanks exercises.
- Such is the generalized learning of BERT that it can be used even for translation-related tasks, even though it has not been specifically taught translation as a task.

# Practical Challenges in Extractive Summarization

- Given the rapid pace of development in NLP, it is even more important to use compatible versions of the libraries that we use.
- Evaluation of a document's suitability for extractive summarization can be undertaken manually. Often, we would like to summarize multiple pieces of text, all of which could be short in length.
- The TextRank algorithm will not work well in such cases.

# Summary

- In this lesson, we learned about text generation using Markov chains and extractive summarization using the TextRank algorithm.
- We also explored both the power and limitations of various advanced approaches.

# 8. Sentiment Analysis

# Overview

- This lesson introduces you to one of the most exciting applications of natural language processing—that is, sentiment analysis.
- You will explore the various tools used to perform sentiment analysis, such as popular NLP libraries and deep learning frameworks.
- You will then perform sentiment analysis on given text data using the powerful textblob library.

# Introduction

- In the previous lesson, we looked at text generation, paraphrasing, and summarization, all of which can be immensely useful in helping us focus on only the essential and meaningful parts of the text corpus.
- This, in turn, helps us to further refine the results of our NLP project.
- In this lesson, we will look at sentiment analysis, which, as the name suggests, is the area of NLP that involves teaching computers how to identify the sentiment behind written content or parsed audio—that is, audio converted to text.

# Why Is Sentiment Analysis Required?

- In machine learning projects, we try to build applications that work similarly to a human being.
- We measure success in part by seeing how close our application is to matching human-level performance.
- Generally, machine learning programs cannot exceed human-level performance by a significant margin—especially if our training data source is human-generated.

# The Growth of Sentiment Analysis

- The field of sentiment analysis is driven by a few main factors.
- Firstly, it's driven by the rapid growth in online content that's used by companies to understand and respond to how people feel.
- Secondly, since sentiment drives human decisions, businesses that understand their customers' sentiments have a major advantage in predicting and shaping purchasing decisions.

# The Monetization of Emotion

- The growth of the internet and internet services has enabled new business models to work with human connection, communication, and sentiment.
- In January 2020, Facebook had about 61.3% of the social media traffic and has been one of the most successful social media platforms at connecting people across the world and providing features that enable users to express their thoughts and post memorable moments from their life online

# Types of Sentiments

There are various sentiments that we can try to detect in language sources. Let's discuss a few of them in detail.

## Emotion

- Sentiment analysis is often used to detect the emotional state of a person.
- It checks whether the person is happy or sad, or content or discontent. Businesses often use it to improve customer satisfaction.
- For example, let's look at the following statement:  
**"I thought I would have enjoyed the movie, but it left me feeling that it could have been better."**

# Key Ideas and Terms

Let's look at some of the key ideas and terms that are used in sentiment analysis.

- Classification
- Supervised Learning
- Polarity
- Intensity

# Applications of Sentiment Analysis

There are various applications of sentiment analysis.

- Financial Market Sentiment
- Product Satisfaction
- Social Media Sentiment
- Brand Monitoring
- Customer Interaction

# Tools Used for Sentiment Analysis

There are a lot of tools capable of analyzing sentiment. Each tool has its advantages and disadvantages. We will look at each of them in detail.

## **NLP Services from Major Cloud Providers**

- Online sentiment analysis is carried out by all major cloud services providers, such as Amazon, Microsoft, Google, and IBM.
- You can usually find sentiment analysis as a part of their text analysis services or general machine learning services.
- Online services offer the convenience of packaging all the necessary algorithms behind the provider's API.

# Online Marketplaces

- Recently, AI marketplaces have emerged that offer different algorithms from third parties.
- Online marketplaces differ from cloud providers.
- An online marketplace allows third-party developers to deploy sentiment analysis services on their platform.

# Python NLP Libraries

- There are a few NLP libraries that need to be integrated into your project instead of being called upon as services.
- These are called dedicated NLP libraries and they usually include many NLP algorithms from academic research.
- Sophisticated NLP libraries used across the industry are spaCy, gensim, and AllenNLP.

# Deep Learning Frameworks

- Deep learning libraries such as PyTorch and TensorFlow are meant to be used to build complex models for a wide range of applications, not limited to just NLP.
- These libraries provide you with more advanced algorithms and mathematical functions, helping you develop powerful and complex models.

# The textblob library

- textblob is a Python library used for NLP, as we've seen in the previous lessons.
- It has a simple API and is probably the easiest way to begin with sentiment analysis. textblob is built on top of the NLTK library but is much easier to use.
- In the following sections, we will do an exercise and an activity to get a better understanding of how we can use textblob for sentiment analysis.

# **Complete Exercise 8.01: Basic Sentiment Analysis Using the textblob Library**

# Complete Activity 8.01: Tweet Sentiment Analysis Using the textblob library

# Understanding Data for Sentiment Analysis

- Sentiment analysis is a type of text classification. Sentiment analysis models are usually trained using supervised datasets.
- Supervised datasets are a kind of dataset that is labeled with the target variable, usually a column that specifies the sentiment value in the text. This is the value we want to predict in the unseen text.

# Complete Exercise 8.02: Loading Data for Sentiment Analysis

# Training Sentiment Models

- The end product of any sentiment analysis project is a sentiment model.
- This is an object containing a stored representation of the data on which it was trained.
- Such a model has the ability to predict sentiment values for text that it has not seen before.

# **Complete Activity 8.02: Training a Sentiment Model Using TFIDF and Logistic Regression**

# Summary

- We started our journey into NLP with basic text analytics and text preprocessing techniques, such as tokenization, stemming, lemmatization, and lowercase conversion, to name a few.
- We then explored ways in which we can represent our text data in numerical form so that it can be understood by machines in order to implement various algorithms.