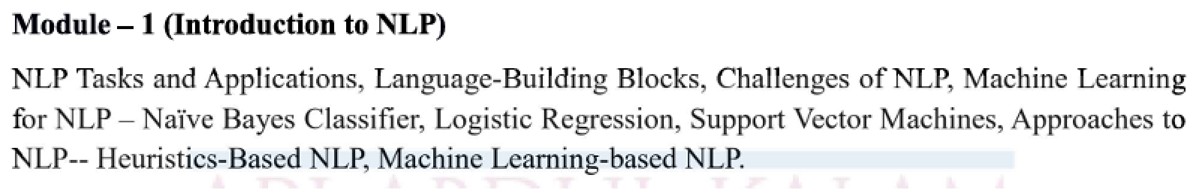
Concepts of Natural language Processing
Module 1



Module – 1 (Introduction to NLP)

NLP Tasks and Applications, Language-Building Blocks, Challenges of NLP, Machine Learning for NLP – Naïve Bayes Classifier, Logistic Regression, Support Vector Machines, Approaches to NLP-- Heuristics-Based NLP, Machine Learning-based NLP.

Natural language processing (NLP) is a field at the intersection of computer science, artificial intelligence, and linguistics. It concerns building systems that can process and understand human language. The past decades breakthroughs have resulted in NLP being increasingly used in a range of diverse domains such as retail, healthcare, finance, law, marketing, human resources, and many more.

NLP Tasks

Language modeling

This is the task of predicting what the next word in a sentence will be based on the history of previous words. The goal of this task is to learn the probability of a sequence of words appearing in a given language. Language modeling is useful for building solutions for a wide variety of problems, such as speech recognition, optical character recognition, handwriting recognition, machine translation, and spelling correction

Text classification

This is the task of bucketing the text into a known set of categories based on its content. Text classification is by far the most popular task in NLP and is used in a variety of tools, from email spam identification to sentiment analysis.

Information extraction

As the name indicates, this is the task of extracting relevant information from text, such as calendar events from emails or the names of people mentioned in a social media post.

Information retrieval

This is the task of finding documents relevant to a user query from a large collection. Applications like Google Search are well-known use cases of information retrieval.

Conversational agent

This is the task of building dialogue systems that can converse in human languages. Alexa, Siri, etc., are some common applications of this task.

Text summarization

This task aims to create short summaries of longer documents while retaining the core content and preserving the overall meaning of the text.

Question answering

This is the task of building a system that can automatically answer questions posed in natural language.

Machine translation

This is the task of converting a piece of text from one language to another. Tools like Google Translate are common applications of this task.

Topic modeling
This is the task of uncovering the topical structure of a large collection of documents. Topic modeling is a common text-mining tool and is used in a wide range of domains, from literature to bioinformatics.

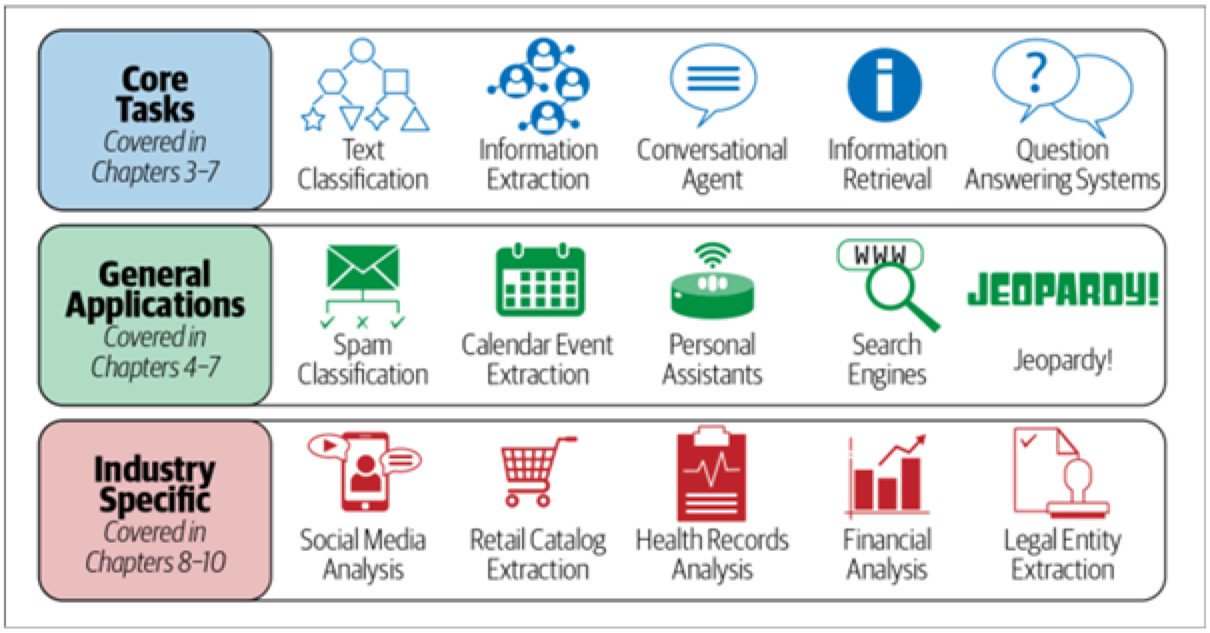


Figure 1-1. NLP tasks and applications

What Is Language?

Language is a structured system of communication that involves complex combinations of its constituent components, such as characters, words, sentences, etc. Linguistics is the systematic study of language.

Human language is composed of four major building blocks:

- Phonemes
- Morphemes and Lexemes
- Syntax
- Context

NLP applications need knowledge of different levels of these building blocks, starting from the basic sounds of language (phonemes) to texts with some meaningful expressions (context).

Figure 1-3 shows these building blocks of language, what they encompass, and a few NLP applications we introduced earlier that require this knowledge. Some of the terms listed here that were not introduced earlier in this chapter (e.g., parsing, word embeddings, etc.) will be introduced later in these first three chapters.

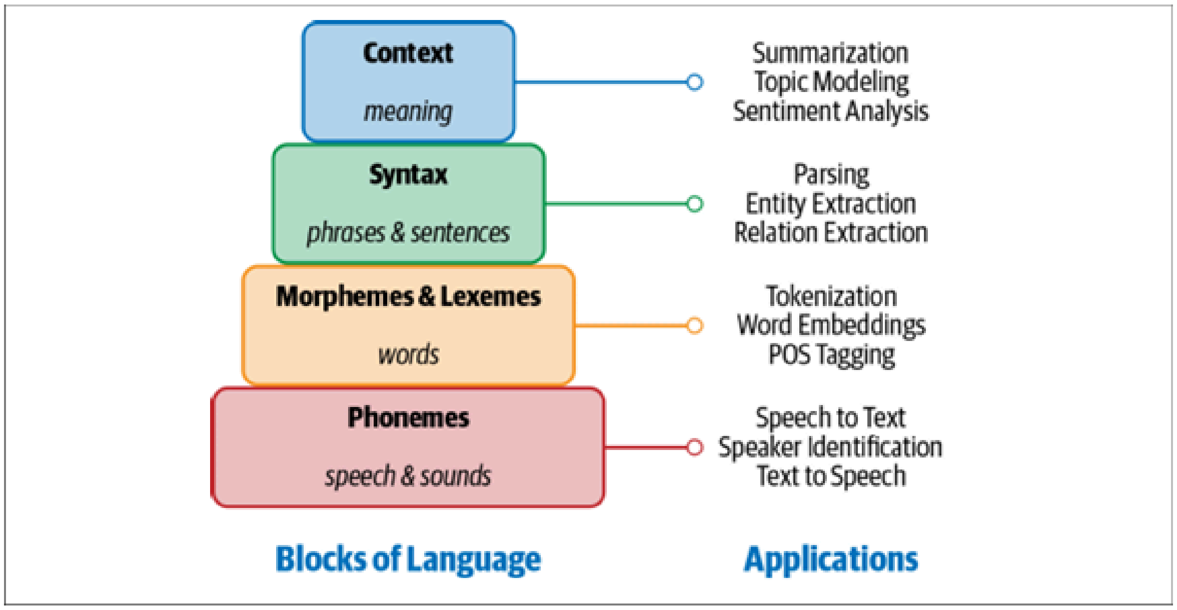


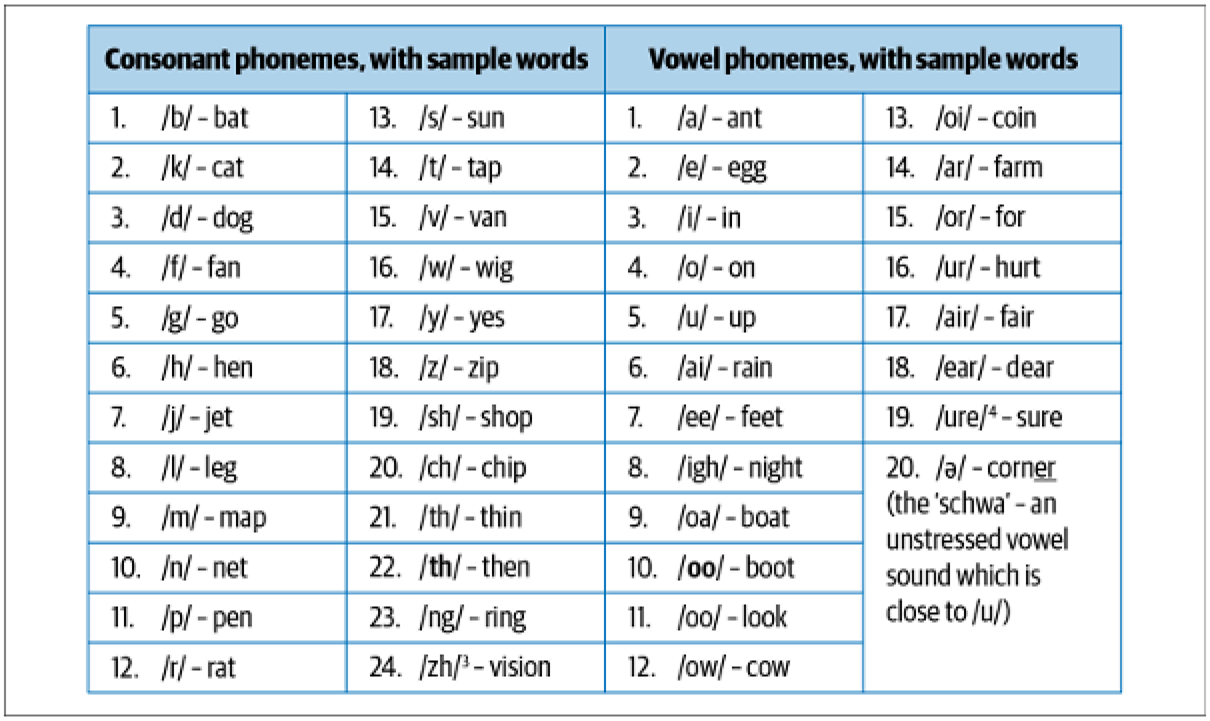
Figure 1-3. Building blocks of language and their applications

Building Blocks of Language

❖ **Phonemes**

Phonemes are the smallest units of sound in a language. They may not have any meaning by themselves but can induce meanings when uttered in combination with other phonemes.

For example, standard English has 44 phonemes, which are either single letters or a combination of letters



Consonant phonemes, with sample words		Vowel phonemes, with sample words	
1. /b/ - bat	13. /s/ - sun	1. /a/ - ant	13. /oi/ - coin
2. /k/ - cat	14. /t/ - tap	2. /e/ - egg	14. /ar/ - farm
3. /d/ - dog	15. /v/ - van	3. /i/ - in	15. /or/ - for
4. /f/ - fan	16. /w/ - wig	4. /o/ - on	16. /ur/ - hurt
5. /g/ - go	17. /y/ - yes	5. /u/ - up	17. /air/ - fair
6. /h/ - hen	18. /z/ - zip	6. /ai/ - rain	18. /ear/ - dear
7. /j/ - jet	19. /sh/ - shop	7. /ee/ - feet	19. /ure/[4] - sure
8. /l/ - leg	20. /ch/ - chip	8. /igh/ - night	20. /ə/ - corner (the 'schwa' - an unstressed vowel sound which is close to /u/)
9. /m/ - map	21. /th/ - thin	9. /oa/ - boat	
10. /n/ - net	22. /**th**/ - then	10. /**oo**/ - boot	
11. /p/ - pen	23. /ng/ - ring	11. /oo/ - look	
12. /r/ - rat	24. /zh/[3] - vision	12. /ow/ - cow	

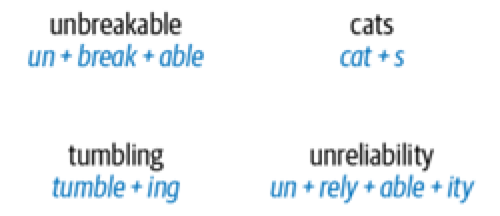
Figure 1-4. Phonemes and examples

Phonemes are particularly important in applications involving speech understanding, such as speech recognition, speech-to-text transcription, and text-to-speech conversion

❖ **Morphemes and lexemes**

A morpheme is the smallest unit of language that has a meaning. It is formed by a combination of phonemes. Not all morphemes are words, but all prefixes and suffixes are morphemes.

For example, in the word “multimedia,” “multi-” is not a word but a prefix that changes the meaning when put together with “media.” “Multi-” is a morpheme.



For words like “cats” and “unbreakable,” their morphemes are just constituents of the full word, whereas for words like “tumbling” and “unreliability,” there is some variation when breaking the words down into their morphemes.

Lexemes are the structural variations of morphemes related to one another by meaning.

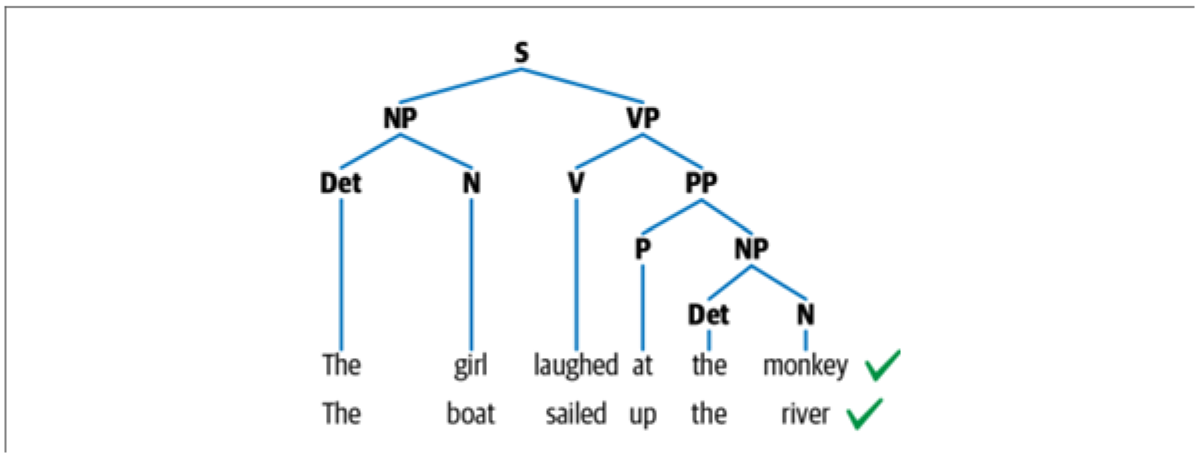
For example, “run” and “running” belong to the same lexeme form. Morphological analysis, which analyzes the structure of words by studying its morphemes and lexemes, is a foundational block for many NLP tasks, such as tokenization, stemming, learning word embeddings, and part-of-speech tagging.

A **lexicon** refers to a dictionary or a collection of words along with their associated meanings, properties, or metadata. It can include:

1. **Word Forms:** Different inflections of a word (e.g., "run," "running," "ran").
2. **Parts of Speech:** The grammatical category of words (e.g., "run" as a noun or verb).
3. **Synonyms/Antonyms:** Words with similar or opposite meanings.
4. **Domain-Specific Terms:** For example, a medical lexicon might contain words like "cardiology," "pacemaker," and "angioplasty."

❖ **Syntax**

Syntax is a set of rules to construct grammatically correct sentences out of words and phrases in a language. Syntactic structure in linguistics is represented in many different ways. A common approach to representing sentences is a parse tree.


S
NP
VP
Det
N
V
PP
P
NP
Det
N
The girl laughed at the monkey ✓
The boat sailed up the river ✓

This has a hierarchical structure of language, with words at the lowest level, followed by part-of-speech tags, followed by phrases, and ending with a sentence at the highest level.

N stands for noun | V for verb | P for preposition | Noun phrase by NP |Verb phrase by VP

The two noun phrases are “The girl” and “The boat,” while the two verb phrases are “laughed at the monkey” and “sailed up the river.” The syntactic structure is guided by a set of grammar rules for the language (e.g., the sentence comprises an NP and a VP), and this in turn guides some of the fundamental tasks of language processing, such as parsing.

Parsing is the NLP task of constructing such trees automatically.

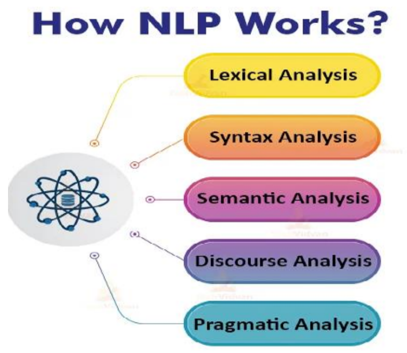
Entity extraction and relation extraction are some of the NLP tasks that build on this knowledge of parsing

Note that the parse structure described above is specific to English. The syntax of one language can be very different from that of another language, and the language-processing approaches needed for that language will change accordingly.

- **Context**

Context is how various parts in a language come together to convey a particular meaning. Context includes long-term references, world knowledge, and common sense along with the literal meaning of words and phrases. The meaning of a sentence can change based on the context, as words and phrases can sometimes have multiple meanings. Generally, context is composed from semantics and pragmatics. Semantics is the direct meaning of the words and sentences without external context. Pragmatics adds world knowledge and external context of the conversation to enable us to infer implied meaning. Complex NLP tasks such as sarcasm detection, summarization, and topic modeling are some of tasks that use context heavily. Linguistics is the study of language and hence is a vast area in itself, and we only introduced some basic ideas to illustrate the role of linguistic knowledge in NLP. Different tasks in NLP require varying degrees of knowledge about these building blocks of language.

How NLP works



Five main Component of Natural Language processing in AI are:

1. Morphological and Lexical Analysis
2. Syntactic Analysis
3. Semantic Analysis
4. Discourse Integration
5. Pragmatic Analysis

1. Morphological and Lexical Analysis

Lexical analysis is a vocabulary that includes its words and expressions. It depicts analyzing, identifying and describing the structure of words. It includes dividing a text into paragraphs, words and the sentences. Individual words are analyzed into their components, and nonword tokens such as punctuations are separated from the words.

2. Syntax Analysis

The words are commonly accepted as being the smallest units of syntax. The syntax refers to the principles and rules that govern the sentence structure of any individual languages. Syntax focus about the proper ordering of words which can affect its meaning. This involves analysis of the words in a sentence by following the grammatical structure of the sentence. The words are transformed into the structure to show hows the word are related to each other.

3. Semantic Analysis

Semantic Analysis is a structure created by the syntactic analyzer which assigns meanings. This component transfers linear sequences of words into structures. It shows how the words are associated with each other. Semantics focuses only on the literal meaning of words, phrases, and sentences. This only abstracts the dictionary meaning or the real meaning from the given context. The structures assigned by the syntactic analyzer always have assigned meaning E.g.. “colorless green idea.” This would be rejected by the Semantic analysis as colorless Here; green doesn’t make any sense.

4. Discourse Integration

It means a sense of the context. The meaning of any single sentence depends upon that sentences. It also considers the meaning of the following sentence. For example, the word “that” in the sentence “He wanted that” depends upon the prior discourse context.

5. Pragmatic Analysis

Pragmatic Analysis deals with the overall communicative and social content and its effect on interpretation. It means abstracting or deriving the meaningful use of language in situations. In this analysis, the main focus is always on what was said and reinterpreted on what is meant. Pragmatic analysis helps users to discover this intended effect by applying a set of rules that characterize cooperative dialogues. E.g., "close the window?" should be interpreted as a request instead of an order.

Natural Language Understanding(NLU) is an area of artificial intelligence to process input data provided by the user in natural language say text data or speech data. It is a way that enables interaction between a computer and a human in a way like humans do using natural languages like English, French, Hindi etc.

Natural Language Generation(NLG) is a sub-component of Natural language processing that helps in generating the output in a natural language based on the input provided by the user. This component responds to the user in the same language in which the input was provided, say the user asks something in English then the system will return the output in English.

NLP = NLU+NLG

Challenges of NLP

The ambiguity and creativity of human language are just two of the characteristics that make NLP a demanding area to work in.

- **Ambiguity**

Ambiguity means uncertainty of meaning. Most human languages are inherently ambiguous.

Consider the following sentence: "I made her duck."

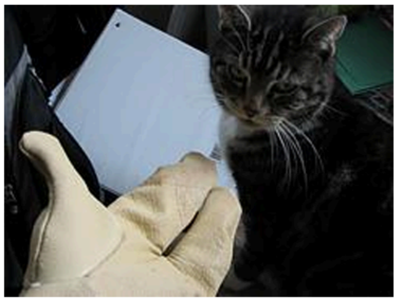
This sentence has multiple meanings.

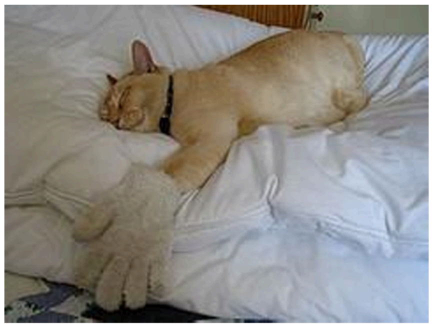
The first one is: I cooked a duck for her.
The second meaning is: I made her bend down to avoid an object.

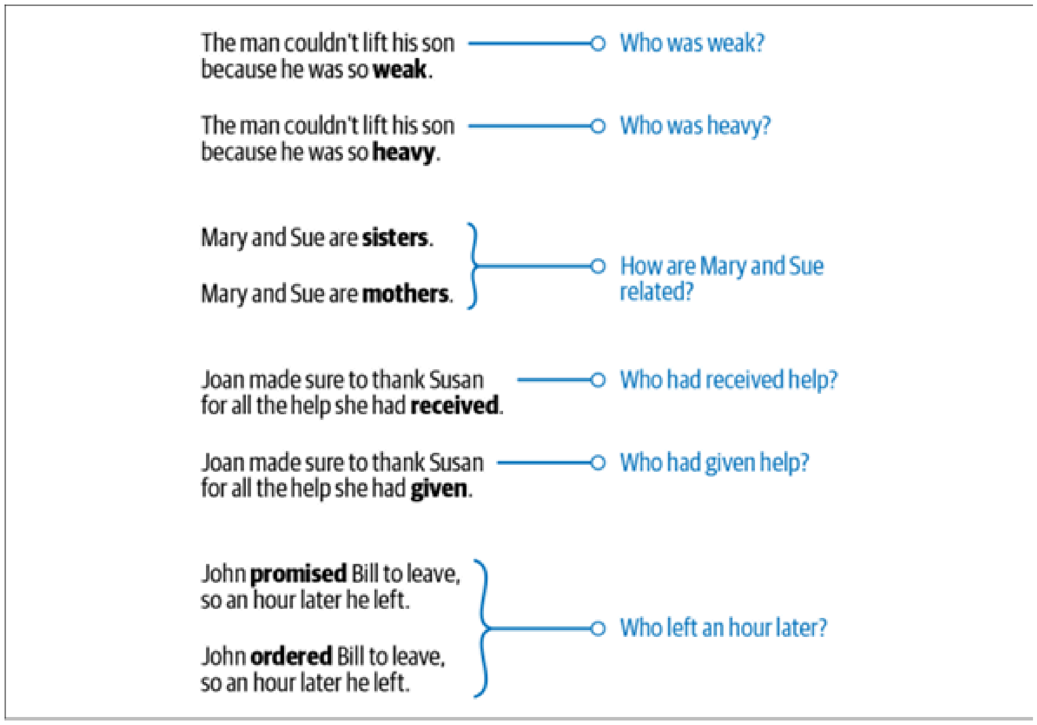
Here, the ambiguity comes from the use of the word "made." Which of the two meanings applies depends on the context in which the sentence appears. If the sentence appears in a story about a mother and a child, then the first meaning will probably apply. But if the sentence appears in a book about sports, then the second meaning will likely apply. The example we saw is a direct sentence.

When it comes to figurative language—i.e., idioms—the ambiguity only increases. For example, “He is as good as John Doe.”
Try to answer, “How good is he?”
The answer depends on how good John Doe is.

“Get the cat with the gloves.”







The man couldn't lift his son because he was so **weak**. — Who was weak?

The man couldn't lift his son because he was so **heavy**. — Who was heavy?

Mary and Sue are **sisters**.
Mary and Sue are **mothers**. — How are Mary and Sue related?

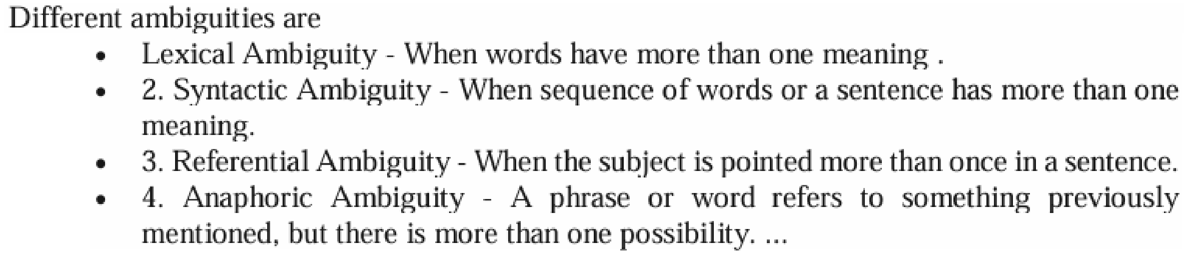
Joan made sure to thank Susan for all the help she had **received**. — Who had received help?

Joan made sure to thank Susan for all the help she had **given**. — Who had given help?

John **promised** Bill to leave, so an hour later he left.
John **ordered** Bill to leave, so an hour later he left. — Who left an hour later?

These examples are easily disambiguated by a human but are not solvable using most NLP techniques.
Consider the pairs of sentences in the figure and the questions associated with them. With some thought, how the answer changes should be apparent based on a single word variation.

As another experiment, consider taking an off-the-shelf NLP system like Google Translate and try various examples to see how such ambiguities affect (or dont affect) the output of the system.



Different ambiguities are

- Lexical Ambiguity - When words have more than one meaning .
- 2. Syntactic Ambiguity - When sequence of words or a sentence has more than one meaning.
- 3. Referential Ambiguity - When the subject is pointed more than once in a sentence.
- 4. Anaphoric Ambiguity - A phrase or word refers to something previously mentioned, but there is more than one possibility. ...

➔ **Common knowledge**

A key aspect of any human language is "common knowledge." It is the set of all facts that most humans are aware of. In any conversation, it is assumed that these facts are known, hence they are not explicitly mentioned, but they do have a bearing on the meaning of the sentence. For example, consider two sentences: "man bit dog" and "dog bit man." We all know that the first sentence is unlikely to happen, while the second one is very possible. Why do we say so? Because we all "know" that it is very unlikely that a human will bite a dog. Further, dogs are known to bite humans. This knowledge is required for us to say that the first sentence is unlikely to happen while the second one is possible. Note that this common knowledge was not mentioned in either sentence. Humans use common knowledge all the time to understand and process any language. In the above example, the two sentences are syntactically very similar, but a computer would find it very difficult to differentiate between the two, as it lacks the common knowledge humans have. One of the key challenges in NLP is how to encode all the things that are common knowledge to humans in a computational model.

➔ **Creativity**

Language is not just rule driven; there is also a creative aspect to it. Various styles, dialects, genres, and variations are used in any language. Poems are a great example of creativity in language. Making machines understand creativity is a hard problem not just in NLP, but in AI in general.

➔ **Diversity across languages**

For most languages in the world, there is no direct mapping between the vocabularies of any two languages. This makes porting an NLP solution from one language to another hard. A solution that works for one language might not work at all for another language. This means that one either builds a solution that is language agnostic or that one needs to build separate solutions for each language. While the first one is conceptually very hard, the other is laborious and time intensive. All these issues make NLP a challenging—yet rewarding—domain to work in. Before looking into how some of these challenges are tackled in NLP, we should know the common approaches to solving NLP problems. Let's start with an overview of how machine learning and deep learning are connected to NLP before delving deeper into different approaches to NLP.

Approaches to NLP

The different approaches used to solve NLP problems commonly fall into three categories:

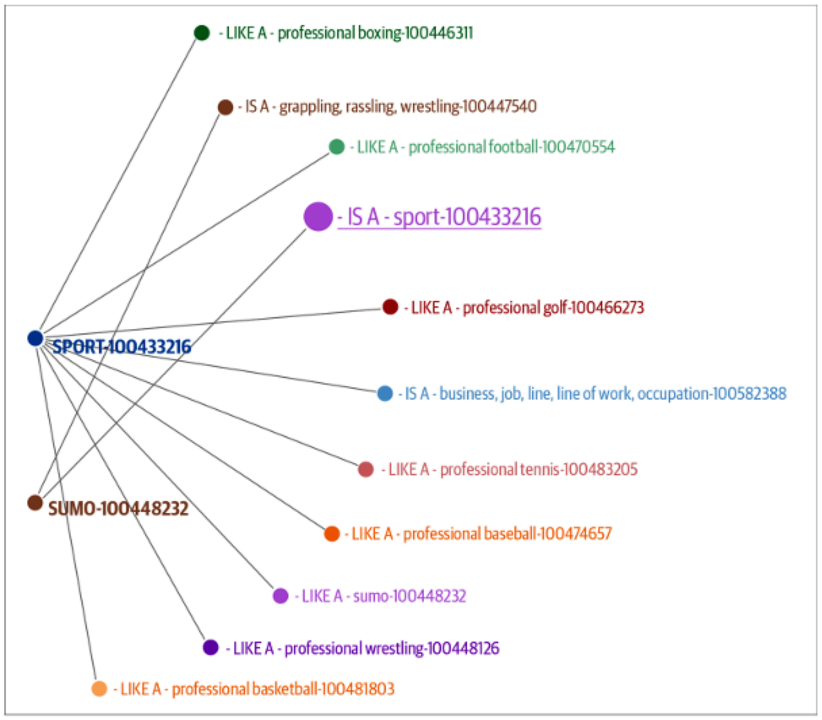
- Heuristics
- Machine learning
- Deep learning.

Heuristics-Based NLP
Similar to other early AI systems, early attempts at designing NLP systems were based on building rules for the task at hand. This required that the developers had some expertise in the domain to formulate rules that could be incorporated into a program. Such systems also required resources like dictionaries and thesauruses, typically compiled and digitized over a period of time. An example of designing rules to solve an NLP problem using such resources is lexicon-based sentiment analysis. It uses counts of positive and negative words in the text to deduce the sentiment of the text.

Besides dictionaries and thesauruses, more elaborate knowledge bases have been built to aid NLP in general and rule-based NLP in particular.

One example is Wordnet, which is a database of words and the semantic relationships between them. Some examples of such relationships are synonyms, hyponyms, and meronyms. Synonyms refer to words with similar meanings.
Hyponyms capture is-type-of relationships. For example, baseball, sumo wrestling, and tennis are all hyponyms of sports.
Meronyms capture is-part-of relationships. For example, hands and legs are meronyms of the body. All this information becomes useful when building rule-based systems around language.

[Additional : What is Rule-Based System? - These systems rely on if-then rules - predefined rules, patterns, or heuristics are used to analyze, extract, or process information- Eg: System to detect fake dates - Rules: The months should between 1 and 12, etc - i.e. we are not executing the model till the end to give the answer , there are predefined rules]


- LIKE A - professional boxing-100446311
- IS A - grappling, rassling, wrestling-100447540
- LIKE A - professional football-100470554
- IS A - sport-100433216
- LIKE A - professional golf-100466273
SPORT-100433216
- IS A - business, job, line, line of work, occupation-100582388
- LIKE A - professional tennis-100483205
SUMO-100448232
- LIKE A - professional baseball-100474657
- LIKE A - sumo-100448232
- LIKE A - professional wrestling-100448126
- LIKE A - professional basketball-100481803
Example depiction of such relationships between words using Wordnet.

More recently, common sense world knowledge has also been incorporated into knowledge bases like Open Mind Common Sense, which also aids such rule-based systems. While what we've seen so far are largely lexical resources based on word level information, rule-based systems go beyond words and can incorporate other forms of information, too.
Some of them are introduced below.
Regular expressions (regex) are a great tool for text analysis and building rule-based systems. A regex is a set of characters or a pattern that is used to match and find substrings in text.

For example, a regex like ^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.] +)\.([a-zA-Z]{2,5})$ is used to find all email IDs in a piece of text.
Regexes are a great way to incorporate domain knowledge in your NLP system.

For example, given a customer complaint that comes via chat or email, we want to build a system to automatically identify the product the complaint is about. There is a range of product codes that map to certain brand names. We can use regexes to match these easily.

Regexes are used for deterministic matches—meaning its either a match or its not. Probabilistic regexes is a sub-branch that addresses this limitation by including a probability of a match.

Context-free grammar (CFG) is a type of formal grammar that is used to model natural languages. CFGs can be used to capture more complex and hierarchical information that a regex might not. The Earley parser allows parsing of all kinds of CFGs. To model more complex rules, grammar languages like JAPE (Java Annotation Patterns Engine) can be used. JAPE has features from both regexes as well as CFGs and can be used for rule-based NLP systems like GATE (General Architecture for Text Engineering). GATE is used for building text extraction for closed and well-defined domains where accuracy and completeness of coverage is more important. As an example, JAPE and GATE were used to extract information on pacemaker implantation procedures from clinical reports

Rules and heuristics play a role across the entire life cycle of NLP projects even now. At one end, they're a great way to build first versions of NLP systems. Put simply, rules and heuristics help you quickly build the first version of the model and get a better understanding of the problem at hand. Rules and heuristics can also be useful as features for machine learning–based NLP systems. At the other end of the spectrum of the project life cycle, rules and heuristics are used to plug the gaps in the system. Any NLP system built using statistical, machine learning, or deep learning techniques will make mistakes. Some mistakes can be too expensive—for example, a healthcare system that looks into all the medical records of a patient and wrongly decides to not advise a critical test. This mistake could even cost a life. Rules and heuristics are a great way to plug such gaps in production systems. Now lets turn our attention to machine learn ing techniques used for NLP.

Machine Learning for NLP

Machine learning techniques are applied to textual data just as they are used on other forms of data, such as images, speech, and structured data. Supervised machine learning techniques such as classification and regression methods are heavily used for various NLP tasks. As an example, an NLP classification task would be to classify news articles into a set of news topics like sports or politics. On the other hand, regression techniques, which give a numeric prediction, can be used to estimate the price of a stock based on processing the social media discussion about that stock. Similarly, unsupervised clustering algorithms can be used to club together text documents. Any machine learning approach for NLP, supervised or unsupervised, can be described as consisting of three common steps: extracting features from text, using the feature representation to learn a model, and evaluating and improving the model.

Naive Bayes

Naïve Bayes classifier is a supervised machine learning algorithm, which is used for classification tasks, like text classification.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

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

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

For the news classification example, one way to represent the text numerically is by using the count of domain-specific words, such as sport-specific or politics-specific words, present in the text. We assume that these word counts are not correlated to one another. If the assumption holds, we can use Naive Bayes to classify news articles. While this is a strong assumption to make in many cases, Naive Bayes is commonly used as a starting algorithm for text classification. This is primarily because it is simple to understand and very fast to train and run.

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

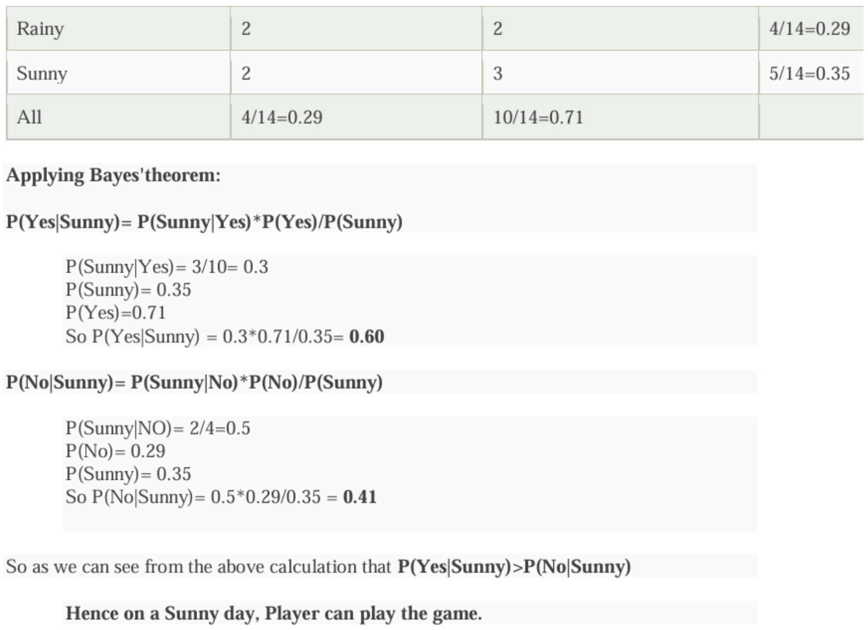
	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35



Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3
P(Sunny)= 0.35
P(Yes)=0.71
So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5
P(No)= 0.29
P(Sunny)= 0.35
So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

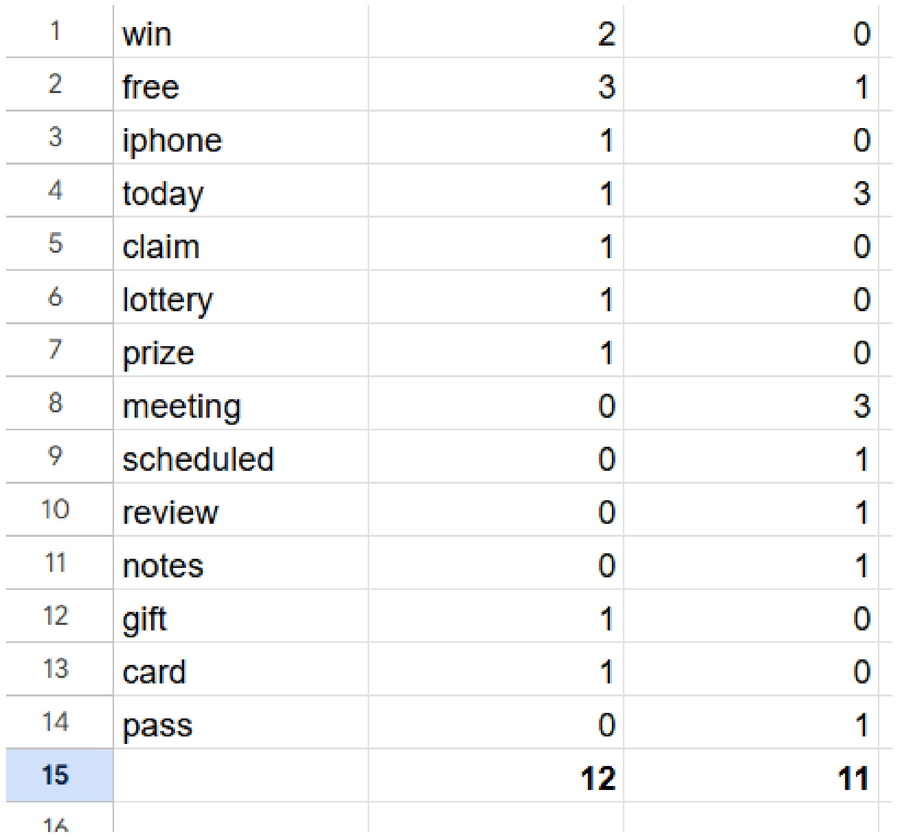
Example Question:
Predict whether ‘Win a free gift today!’ is spam or not using Navie Bayes.

Email Text	Label
Win a free iPhone today	Spam
Claim your free lottery prize	Spam
Meeting is scheduled for today	Not Spam
Review the meeting notes today	Not Spam
Win a free gift card now	Spam
Free meeting pass for today	Not Spam

Step 1: Remove the stop words and get the unique words

[‘win’, ’free’, ‘iphone’,’today’, ‘claim’, ‘lottery’, ‘prize’,’meeting’, ‘scheduled’, ’review’, ’notes’, ’gift’, ‘card’, ‘pass’] => 14 words

Step 2: Get the Bag of words



Sl no	Words	Spam	Not Spam
1	win	2	0
2	free	3	1
3	iphone	1	0
4	today	1	3
5	claim	1	0
6	lottery	1	0
7	prize	1	0
8	meeting	0	3
9	scheduled	0	1
10	review	0	1
11	notes	0	1
12	gift	1	0
13	card	1	0
14	pass	0	1
15		**12**	**11**
16			

NB Classifier:

$$h_{\mathrm{NB}}(\mathbf{x}) = \underset{y \in \mathcal{Y}}{\text{maximize}} \quad P(y \mid \mathbf{x}) = \underset{y \in \mathcal{Y}}{\text{maximize}} \quad \prod_{i=1}^{d} P(x^{(i)} \mid y)\, P(y)$$

Step 3: Remove the stop words and get the unique words from the given question

X = [‘win’, ’free’, ’gift’, ’today’]

Step 4: Calculate the probabilities

P(Spam)= Number of Spam sentences / Total number of sentences = 3 / 6

P(Not Spam)= Number of Not Spam sentences / Total number of sentences = 3 / 6

P(Spam | X) = P(win|spam) x P(free | spam) x P(gift | spam) x P(today| spam) x P(Spam)

You can see some values for X in th BoW table is ZERO => We need to apply laplace smoothing/correction

The correction is only done for X where BoW is zero ; it is done to all the words in X

We add ONE to the numerator and add the number of unique words to the denominator of the probrobality

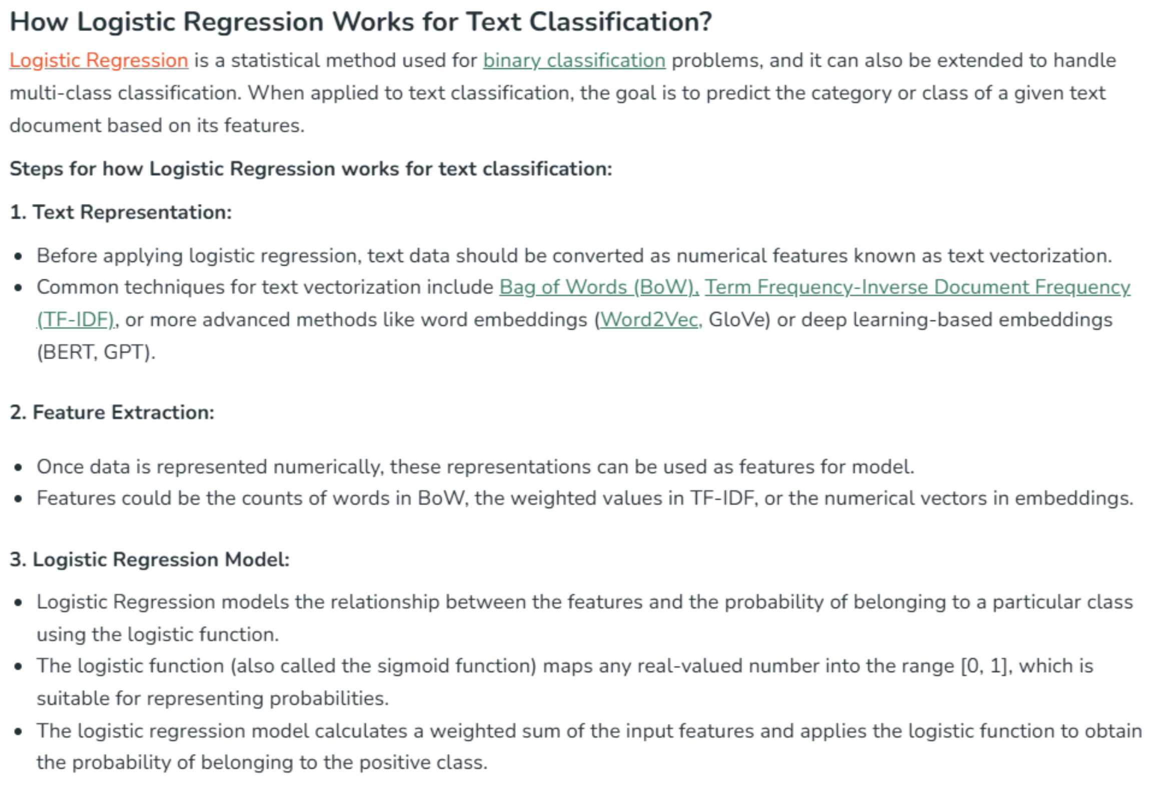
P(Spam | X) = P(win|spam) x P(free | spam) x P(gift | spam) x P(today| spam) x P(Spam)
= (2+1) / (12+14) x (3+1) /(12+14) x (1+1) /(12+14) x (1+1)/(12+14) x 3/6
= 3/26 x 4/26 x 2/26 x 2/26 x 3/6 = 0.000052

P(Not Spam | X) = P(win|not spam) x P(free | not spam) x P(gift | not spam) x P(today| not spam) x P(Not Spam)
= (0+1) / (11+14) x (1+1) /(11+14) x (0+1) /(11+14) x (3+1)/(11+14) x 3/6
= 1/25 x 2/25 x 1/25 x 4/25 x 3/6 = 0.0000102

P(Spam) > P(Not Spam)

X='Win a free gift today' => Spam

Logistic Regression



How Logistic Regression Works for Text Classification?

Logistic Regression is a statistical method used for binary classification problems, and it can also be extended to handle multi-class classification. When applied to text classification, the goal is to predict the category or class of a given text document based on its features.

Steps for how Logistic Regression works for text classification:

1. Text Representation:

- Before applying logistic regression, text data should be converted as numerical features known as text vectorization.
- Common techniques for text vectorization include Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), or more advanced methods like word embeddings (Word2Vec, GloVe) or deep learning-based embeddings (BERT, GPT).

2. Feature Extraction:

- Once data is represented numerically, these representations can be used as features for model.
- Features could be the counts of words in BoW, the weighted values in TF-IDF, or the numerical vectors in embeddings.

3. Logistic Regression Model:

- Logistic Regression models the relationship between the features and the probability of belonging to a particular class using the logistic function.
- The logistic function (also called the sigmoid function) maps any real-valued number into the range [0, 1], which is suitable for representing probabilities.
- The logistic regression model calculates a weighted sum of the input features and applies the logistic function to obtain the probability of belonging to the positive class.

Support vector machine

The support vector machine (SVM) is another popular classification algorithm. The goal in any classification approach is to learn a decision boundary that acts as a separation between different categories of text (e.g., politics versus sports in our news classification example).

This decision boundary can be linear or nonlinear (e.g., a circle). An SVM can learn both a linear and nonlinear decision boundary to separate data points belonging to different classes. A linear decision boundary learns to represent the data in a way that the class differences become apparent.

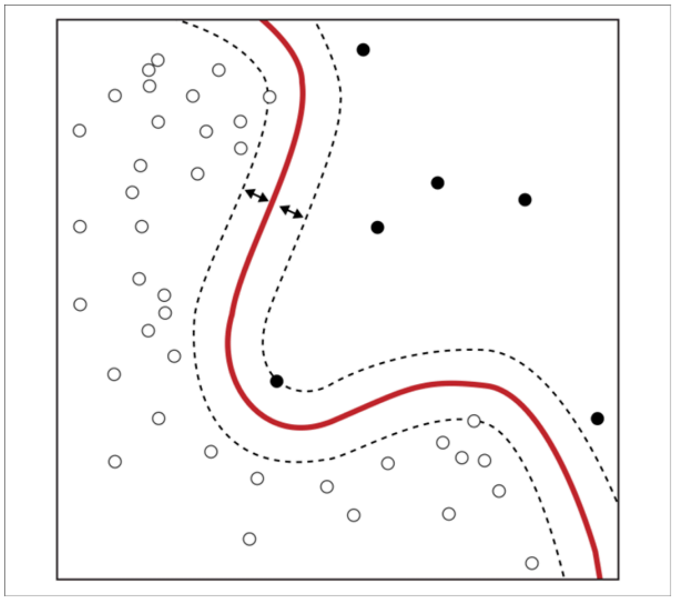
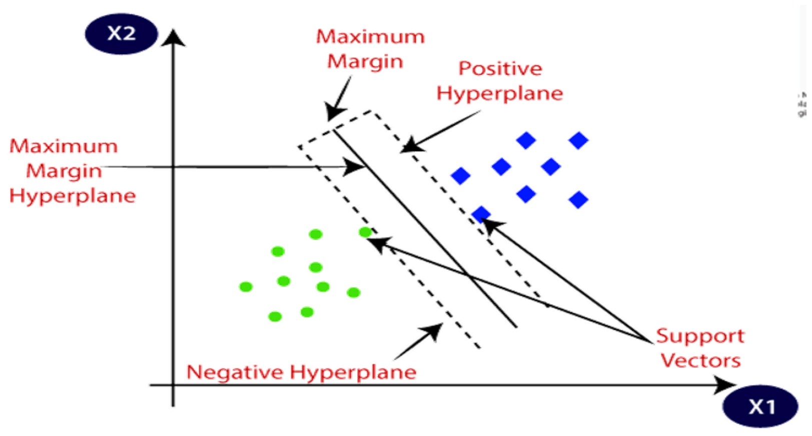


Figure 1-11. A two-dimensional feature representation of an SVM

For two dimensional feature representations, where the black and white points belong to different classes (e.g., sports and politics news groups). An SVM learns an optimal decision boundary so that the distance between points across classes is at its maximum. The biggest strength of SVMs is their robustness to variation and noise in the data. A major weakness is the time taken to train and the inability to scale when there are large amounts of training data.



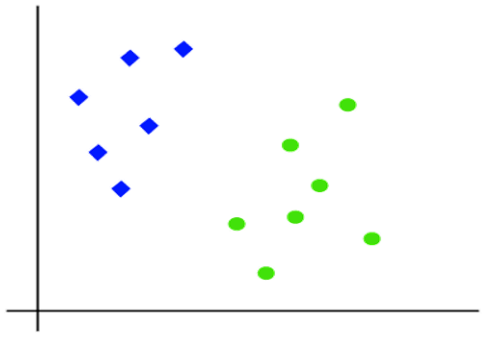
SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

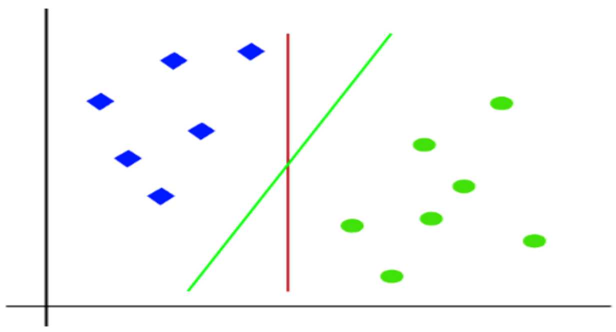
Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

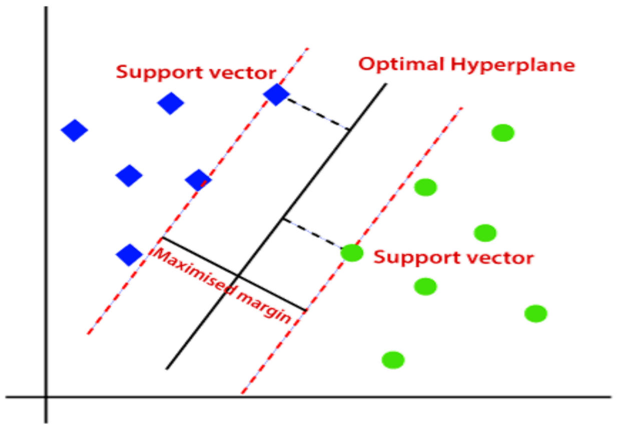
Support Vectors: The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

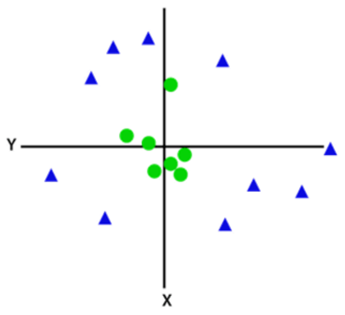


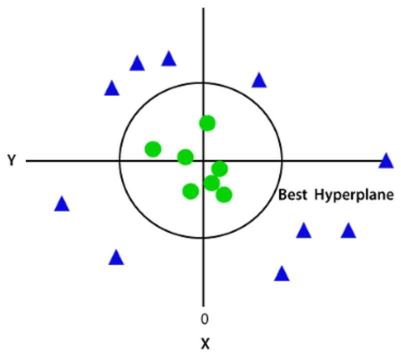


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.

Non-Linear SVM

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line.





Non-Linear SVM is another working principle on SVM that is used for data that cannot be separated linearly because of its high dimensions. Non-linear classification is carried out using the **kernel** concept. The kernel concept in the non-linear case plays a role in determining the classification limits used as a model.

Non-Linear SVM applies the function of the kernel concept to a space that has high dimensions. What is meant by high dimension is that the dataset has more than two features to classify. For example, non-linear classification cases, namely factors that affect human health, consist of age factors, dietary factors, exercise factors, heredity, disease history and stress levels.

The accuracy of the model generated by the process in the SVM algorithm is very dependent on the parameters and **kernel functions** used. In the use of kernel functions in non-linear SVM is something that needs to be considered because the performance of SVM depends on the choice of kernel function.

Non-linier SVM is implemented in practice using a kernel, so it can separate data with the kernel function it called kernel trick**.**

The Kernel Trick

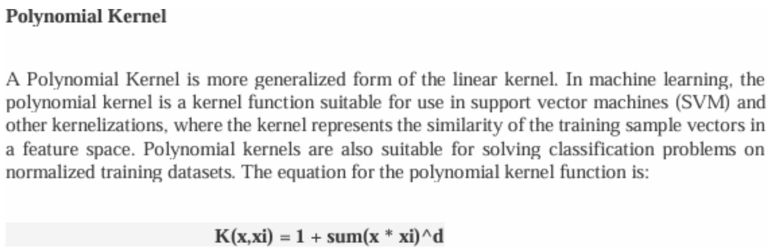
SVM can work well in non-linear data cases using kernel trick. The function of the kernel trick is to map the low-dimensional input space and tranforms into a higher dimensional space.

- **Radial Basis Function Kernel (RBF)**

The RBF kernel is the most widely used kernel concept to solve the problem of classifying datasets that cannot be separated linearly. This kernel is known to have good performance with certain parameters, and the results of the training have a small error value compared to other kernels. The equation formula for the RBF kernel function is:

K(x,xi) = exp(-gamma * sum((x – xi^2))

In the RBF kernel function equation, |xi-x | is the Euclidean Distance between x1 and x2 in two different feature spaces and σ (sigma) is the RBF kernel parameter that determines the kernel weight.

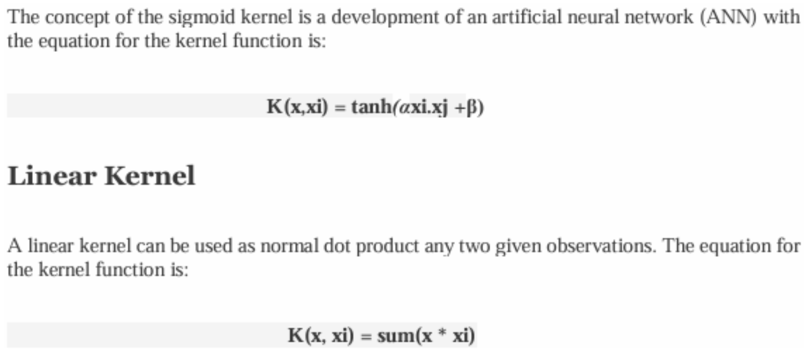


Polynomial Kernel

A Polynomial Kernel is more generalized form of the linear kernel. In machine learning, the polynomial kernel is a kernel function suitable for use in support vector machines (SVM) and other kernelizations, where the kernel represents the similarity of the training sample vectors in a feature space. Polynomial kernels are also suitable for solving classification problems on normalized training datasets. The equation for the polynomial kernel function is:

$$K(x,xi) = 1 + \text{sum}(x * xi)^d$$

Sigmoid Kernel



The concept of the sigmoid kernel is a development of an artificial neural network (ANN) with the equation for the kernel function is:

$$K(x,xi) = \tanh(\alpha xi.xj + \beta)$$

Linear Kernel

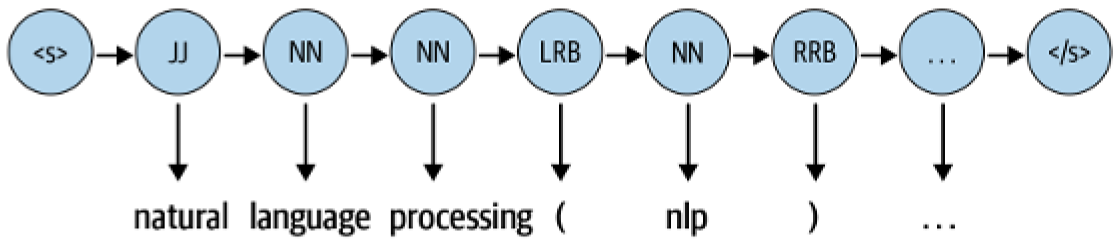
A linear kernel can be used as normal dot product any two given observations. The equation for the kernel function is:

$$K(x, xi) = \text{sum}(x * xi)$$

Hidden Markov Model

The hidden Markov model (HMM) is a statistical model that assumes there is an underlying, unobservable process with hidden states that generates the data—i.e., we can only observe the data once it is generated. An HMM then tries to model the hidden states from this data.

For example, consider the NLP task of part-of-speech (POS) tagging, which deals with assigning part-of-speech tags to sentences. HMMs are used for POS tagging of text data. Here, we assume that the text is generated according to an underlying grammar, which is hidden underneath the text. The hidden states are parts of speech that inherently define the structure of the sentence following the language grammar, but we only observe the words that are governed by these latent states. Along with this, HMMs also make the Markov assumption, which means that each hidden state is dependent on the previous state(s). Human language is sequential in nature, and the current word in a sentence depends on what occurred before it. Hence, HMMs with these two assumptions are a powerful tool for modeling textual data.



We can see an example of an HMM that learns parts of speech from a given sentence. Parts of speech like JJ (adjective) and NN (noun) are hidden states, while the sentence “natural language processing (nlp)…” is directly observed.

Conditional random fields
The conditional random field (CRF) is another algorithm that is used for sequential data. Conceptually, a CRF essentially performs a classification task on each element in the sequence.

Imagine the same example of POS tagging, where a CRF can tag word by word by classifying them to one of the parts of speech from the pool of all POS tags. Since it takes the sequential input and the context of tags into consideration, it becomes more expressive than the usual classification methods and generally performs better. CRFs outperform HMMs for tasks such as POS tagging, which rely on the sequential nature of language.

Deep Learning

Language is inherently complex and unstructured. Therefore, we need models with better representation and learning capability to understand and solve language tasks. Here are a few popular deep neural network architectures that have become the status quo in NLP.

- **Recurrent neural networks**

Language is inherently sequential, and recurrent neural networks (RNNs) are designed to process sequential data effectively. They maintain a temporal memory, updating stored information with each time step as they process inputs, making them powerful for tasks like text classification, named entity recognition, and machine translation. RNNs can also generate text by predicting the next word or character based on preceding context. However, standard RNNs struggle with long-term dependencies, as their memory is limited when handling long input sequences.

Long Short-Term Memory (LSTM) Networks

LSTMs are a type of RNN developed to address the memory limitations of traditional RNNs. They selectively retain relevant context while discarding irrelevant information, enabling them to handle longer sequences more effectively. LSTMs have become widely adopted for NLP tasks, often replacing standard RNNs. Another RNN variant, gated recurrent units (GRUs), is commonly used in tasks like language generation due to their efficiency and simplicity.

Convolutional Neural Networks (CNNs) in NLP

Although CNNs are popular in computer vision, they have also shown success in text classification tasks. By representing sentences as 2D matrices of word vectors, CNNs can process text data similarly to images, using context windows to capture groups of words. This makes CNNs effective for tasks like sentiment analysis. CNNs use convolution and pooling layers to extract features from the text, which are then processed by fully connected layers for classification tasks.

Transformers and Their Applications

Transformers have become the state-of-the-art architecture in NLP, surpassing traditional models in almost all major tasks. Unlike RNNs, transformers use self-attention mechanisms to consider all contextual words simultaneously, enabling nuanced representations of words based on context. Large pre-trained transformer models, like BERT, are trained on massive datasets in an unsupervised manner and fine-tuned for specific downstream tasks, such as text classification or question answering. These models leverage transfer learning to achieve high performance across a variety of NLP applications.

Autoencoders for Text Representation

Autoencoders are unsupervised models used to learn compressed vector representations of inputs. They map input data to a lower-dimensional representation (encoding) and then reconstruct the input from this compressed representation. Variants like LSTM autoencoders are better suited for sequential data like text. Autoencoders are often used for creating feature representations for downstream tasks.

Why Deep Learning Is Not Yet the Silver Bullet for NLP

Over the last few years, DL has made amazing advances in NLP. For example, in text classification, LSTM- and CNN-based models have surpassed the performance of standard machine learning techniques such as Naive Bayes and SVM for many classification tasks. Similarly, LSTMs have performed better in sequence-labeling tasks like entity extraction as compared to CRF models. Recently, powerful transformer models have become state of the art in most of these NLP tasks, ranging from classification to sequence labeling. A huge trend right now is to leverage large (in terms of number of parameters) transformer models, train them on huge datasets for generic NLP tasks like language models, then adapt them to smaller downstream tasks. This approach (known as *transfer learning*) has also been successful in other domains, such as computer vision and speech.

Despite such tremendous success, DL is still not the silver bullet for all NLP tasks when it comes to industrial applications. Some of the key reasons for this are as follows:

Overfitting on small datasets
: DL models tend to have more parameters than traditional ML models, which means they possess more expressivity. This also comes with a curse. Occam's razor [32] suggests that a simpler solution is always preferable given that all other conditions are equal. Many times, in the development phase, sufficient training data is not available to train a complex network. In such cases, a simpler model should be preferred over a DL model. DL models overfit on small datasets and subsequently lead to poor generalization capability, which in turn leads to poor performance in production.

Few-shot learning and synthetic data generation
: In disciplines like computer vision, DL has made significant strides in few-shot learning (i.e., learning from very few training examples) [33] and in models that can generate superior-quality images [34]. Both of these advances have made training DL-based vision models on small amounts of data feasible. Therefore, DL has achieved much wider adoption for solving problems in industrial settings. We have not yet seen similar DL techniques be successfully developed for NLP.

Domain adaptation
: If we utilize a large DL model that is trained on datasets originating from some common domains (e.g., news articles) and apply the trained model to a newer domain that is different from the common domains (e.g., social media posts), it may yield poor performance. This loss in generalization performance indicates that DL models are not always useful. For example, models trained on internet texts and product reviews will not work well when applied to domains such as law, social media, or healthcare, where both the syntactic and semantic structure of the language is specific to the domain. We need specialized models to encode the domain knowledge, which could be as simple as domain-specific, rule-based models.

Interpretable models
: Apart from efficient domain adaptation, controllability and interpretability is hard for DL models because, most of the time, they work like a black box. Businesses often demand more interpretable results that can be explained to the customer or end user. In those cases, traditional techniques might be more useful. For example, a Naive Bayes model for sentiment classification may explain the effect of strong positive and negative words on the final prediction of sentiment.

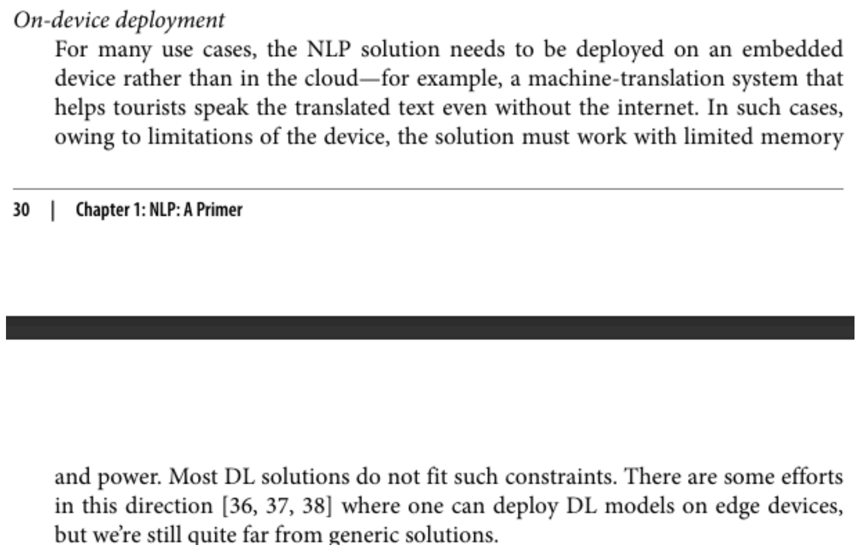
As of today, obtaining such insights from an LSTM-based classification model is difficult. This is in contrast to computer vision, where DL models are not black boxes. There are plenty of techniques [35] in computer vision that are used to gain insight into why a model is making a particular prediction. Such approaches for NLP are not as common.

Common sense and world knowledge

Even though we have achieved good performance on benchmark NLP tasks using ML and DL models, language remains a bigger enigma to scientists. Beyond syntax and semantics, language encompasses knowledge of the world around us. Language for communication relies on logical reasoning and common sense regarding events from the world. For example, "I like pizza" implies "I feel happy when I eat pizza." A more complex reasoning example would be, "If John walks out of the bedroom and goes to the garden, then John is not in the bedroom anymore, and his current location is the garden." This might seem trivial to us humans, but it requires multistep reasoning for a machine to identify events and understand their consequences. Since this world knowledge and common sense are inherent in language, understanding them is crucial for any DL model to perform well on various language tasks. Current DL models may perform well on standard benchmarks but are still not capable of common sense understanding and logical reasoning. There are some efforts to collect common sense events and logical rules (such as if-them reasoning), but they are not well integrated yet with ML or DL models.

Cost

Building DL-based solutions for NLP tasks can be pretty expensive. The cost, in terms of both money and time, stems from multiple sources. DL models are known to be data guzzlers. Collecting a large dataset and getting it labeled can be very expensive. Owing to the size of DL models, training them to achieve desired performance can not only increase your development cycles but also result in a heavy bill for the specialized hardware (GPUs). Further, deploying and maintaining DL models can be expensive in terms of both hardware requirements and effort. Last but not least, because they're bulky, these models may cause latency issues during inference time and may not be useful in cases where low latency is a must. To this list, one can also add technical debt arising from building and maintaining a heavy model. Loosely speaking, technical debt is the cost of rework that arises from prioritizing speedy delivery over good design and implementation choices.


On-device deployment

For many use cases, the NLP solution needs to be deployed on an embedded device rather than in the cloud—for example, a machine-translation system that helps tourists speak the translated text even without the internet. In such cases, owing to limitations of the device, the solution must work with limited memory

30 | Chapter 1: NLP: A Primer

and power. Most DL solutions do not fit such constraints. There are some efforts in this direction [36, 37, 38] where one can deploy DL models on edge devices, but we're still quite far from generic solutions.

Assignment Questions

1. What are the different tasks and applications of NLP
2. Explain Language Building Blocks in NLP.
3. What are the different approaches in solving a NLP problem
4. Differentiate between Linear Regression and Logistic Regression
5. Give the challenges of NLP
6. State Bayes Theorem and Predict whether ‘The movie was not bad and had amazing scenes’ is positive or negative review using Naive Bayes with the following dataset.

Review Text	Label
"The movie was amazing and enjoyable"	Positive
"I loved the scenes, amazing direction"	Positive
"The movie was bad and boring"	Negative
"It had bad scenes and poor acting"	Negative
"The scenes were enjoyable but acting poor"	Negative
"Amazing movie, I would recommend it"	Positive
" The acting was good but the plot was bad"	Negative

7. Using the below scenario,find the probability of buys a computer for X
8. How does SVM classify Text data? Give a brief description involving all the steps

Prepared By:
Esther Sara Thomson
Assistant Professor
SBCE