UNIT-3

Machine Learning Applications: Computer vision, Speech Recognition, Natural Language Processing, Decision making process.

Q) Define Computer Vision? Explain applications of Computer Vision.

- **Computer Vision** is the process of understanding digital images and videos using computers.
- It seeks to automate tasks that human vision can achieve.
- This involves various steps:
  - Acquiring: Gathering the image data from different sources.
  - Pre-processing: pre-processing is done to improve the image data if required like scaling, translating, smoothing (blurring an image to reduce noise).
  - Analyzing: It is the extraction of meaningful information from images. Examples of image analysis techniques in different fields include: 2D and 3D object recognition, image segmentation, motion detection
  - Produce information: Generating the useful information after analysing the image.

Applications of Computer Vision:

Image Classification:

- Image classification is the task of labelling the whole image with an object or concept with confidence.
- Eg. gender classification given an image of a person's face, identifying the type of pet, tagging photos, and so on.

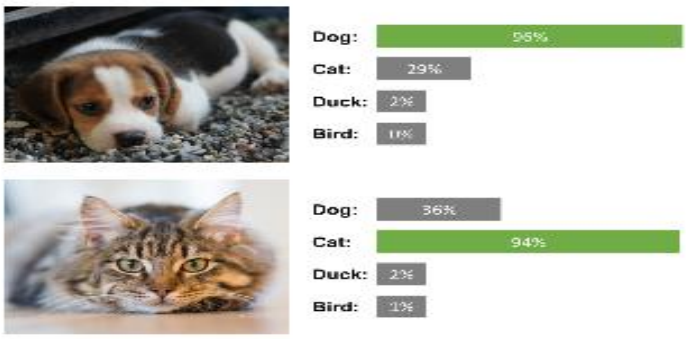


Fig. Classifying Cats and Dogs.

Detection or Localization:

- Detection finds all the objects within the image along with the labels
- Localization detects one object in an image within a label i.e; localize the object with a bounding box.
- Eg. Finding pedestrians, finding sign boards for self-driving vehicles.

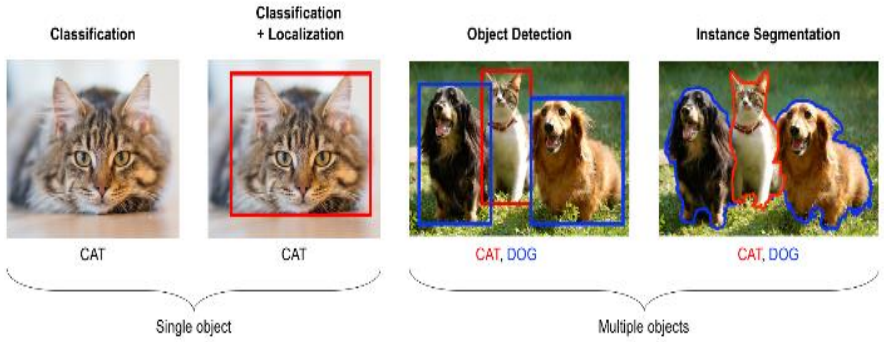


Fig. Illustrating Classification vs Localization and Object Detection

Image Segmentation:

- It is a task of doing pixel wise classification. This gives a fine separation of objects.
- Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images
- Eg. Processing medical images and satellite images.

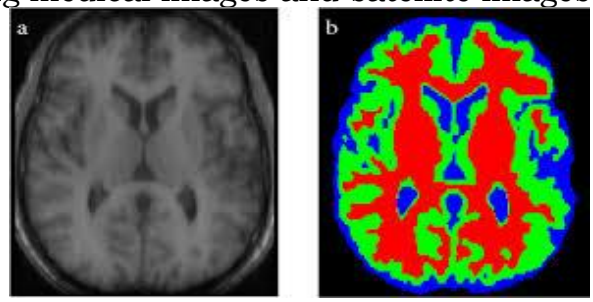
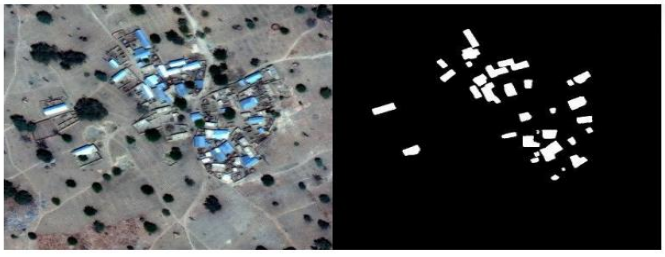
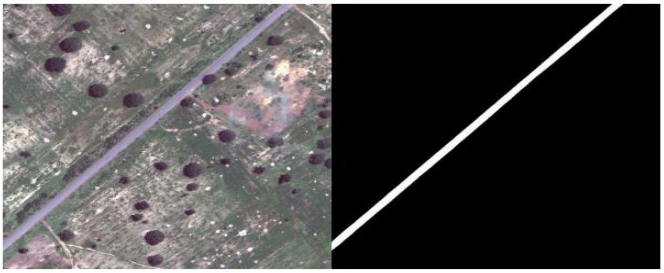


Fig. Medical image segmentation.



Buildings



Roads

Fig. Satellite image segmentation for detecting Buildings and Roads

Image Captioning:

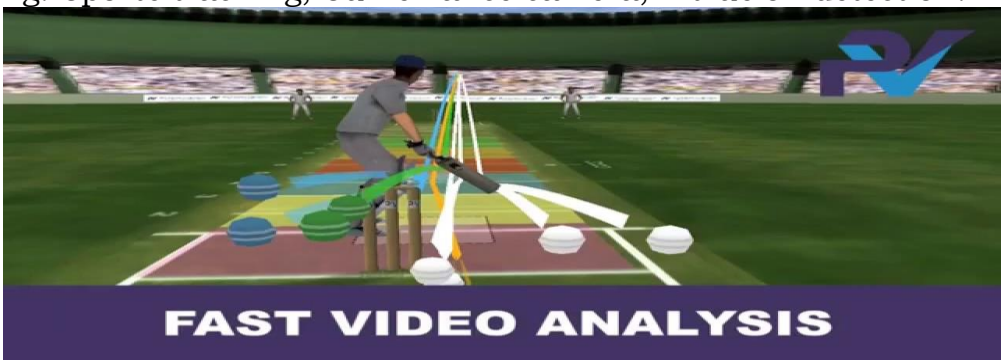
- It is the task of describing the image with text i.e; action in the image. For this NLP is to be combined.

Eg.Persion riding a bike

Video Analysis:

Video motion analysis is a technique used to get information about moving objects from video.

Eg. Sports tracking, Surveillance camera, intrusion detection.



Q) Explain how ML is used for Object Detection. (or) Explain SIFT algorithm for Object Detection.

- A common approach that uses machine learning for object detection is the **Scale-Invariant Feature Transform (SIFT)** algorithm**.**
- SIFT uses key points of objects and stores them in a database. When categorizing an image, SIFT checks the key points of the image, which matches those found in the database.

SIFT algorithm:

- **Scale-space peak selection:** Search over multiple scales and image locations.
- **Keypoint Localization:** Fit a model to determine location and scale. Select keypoints based on a measure of stability.
- **Orientation Assignment:** Compute best orientation(s) for each keypoint region.
- **Keypoint descriptor:** Use local image gradients at selected scale and rotation to describe each keypoint region. Describing the keypoints as a high dimensional vector.
- **Keypoint Matching:** Matches the keypoints to detect whether the objects in the images are same or not.

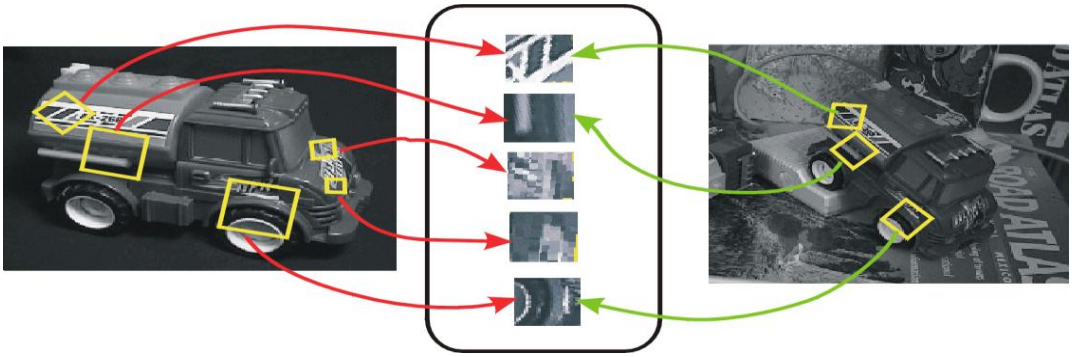


Fig. Illustrating SIFT algorithm by matching keypoints of an object

1. Detection of scale-space extrema:

- For scale invariance, search for stable features across all possible scales using a continuous function of scale, scale space.
- SIFT uses Difference of Gaussian(DoG) filter for scale space because it is efficient and as stable as scale-normalized Laplacian of Gaussian.

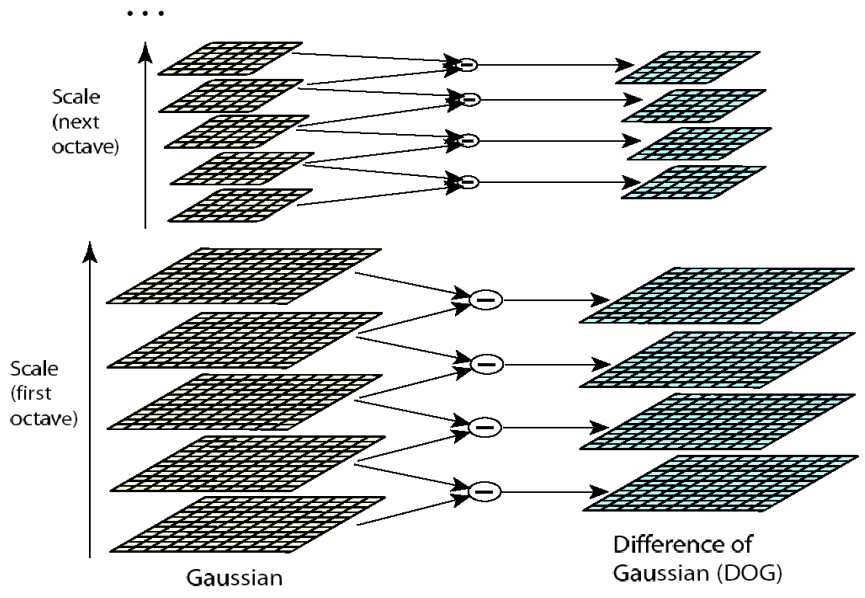


Fig. Scale space and Difference of Gaussian

2. Keypoint localization
   Scale-space extrema detection produces too many keypoint candidates, some of which are unstable. The next step in the algorithm is to perform a detailed fit to the nearby data for accurate location, scale, and ratio of principal curvatures.
   This information allows points to be rejected that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

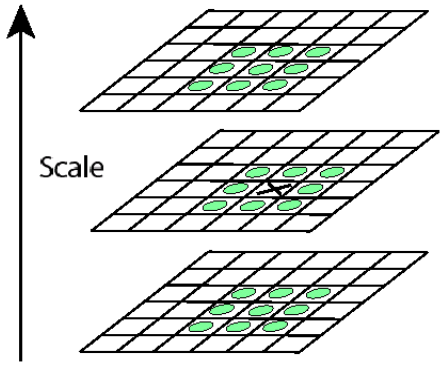


Fig. Keypoint localization

X is selected if it is larger or smaller than all 26 neighbors

3. Orientation assignment
   In this step, each keypoint is assigned one or more orientations based on local image gradient directions.
   This is the key step in achieving invariance to rotation as the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation.

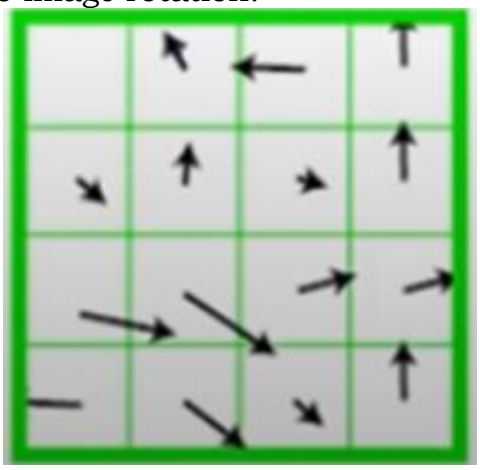


Fig. Keypoint orientation assignment

4. Local image descriptor:

- Now we want to compute a descriptor vector for each keypoint such that the descriptor is highly distinctive and partially invariant to the remaining variations such as illumination, 3D viewpoint, etc.
- A 16x16 neighbourhood around the keypoint is taken and we create an array of orientation histograms w.r.t. key orientation.

- The descriptor then becomes a vector of all the values of these histograms. Since there are 4 × 4 = 16 histograms each with 8 bins the vector has 128 elements (dimensions).
- This vector is then normalized to unit length in order to enhance invariance to affine changes in illumination. The thresholding process, also referred to as clamping.

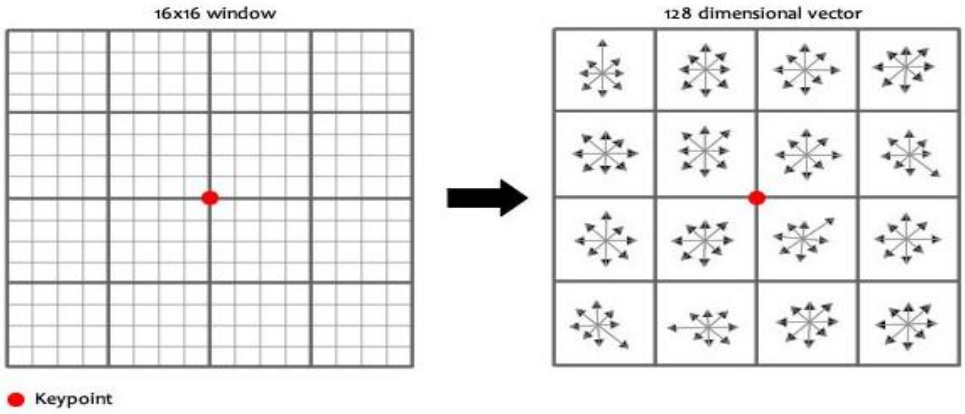


Fig. Keypoint Descriptor

5. Feature matching:

- Keypoints between two images are matched by identifying their nearest neighbours.
- In some cases due to noise for a feature x, found the closest feature x_1 and the second closest feature x_2. If the distance ratio of d(x, x_1) and d(x, x_2) is smaller than 0.8, then it is accepted as a match.

Q) Briefly explain about Image Segmentation based on Clustering.

- It is a task of doing pixel wise classification. This gives a fine separation of objects.
- Divides the pixels of the image into homogeneous clusters.
- easy to implement
- Works really well on small datasets and generates excellent clusters.
- Computation time is too large and expensive. It relies on the number of data elements, number of clusters and number of iteration

Segmentation using K-means:

1. Initialize number of cluster k and centre.
2. For each pixel of an image, calculate the Euclidean distance d, between the center and each pixel of an image using the relation given below.

$$d = || p(x, y) - ck ||$$

3. Assign all the pixels to the nearest centre based on distance d.
4. After all pixels have been assigned, recalculate new position of the centre using the relation given below.

$$c_k = \frac{1}{k} \sum_{y \in c_k} \sum_{x \in c_k} p(x, y)$$

5. Repeat the process until it satisfies the tolerance or error value.
6. Reshape the cluster pixels into image.

Q) Explain how ML can be used for Decision making with an example.

ML learns the optimal decisions directly from the data without having to arbitrarily hardcode decision rules.
ML-based decision making means that the decisions will be more accurate and will improve over time as more examples are made.

The ML workflow: from data to deployment

The ML workflow has five main components:
data preparation, model building, evaluation, optimization, and predictions on new data. The application of these steps has an inherent order.

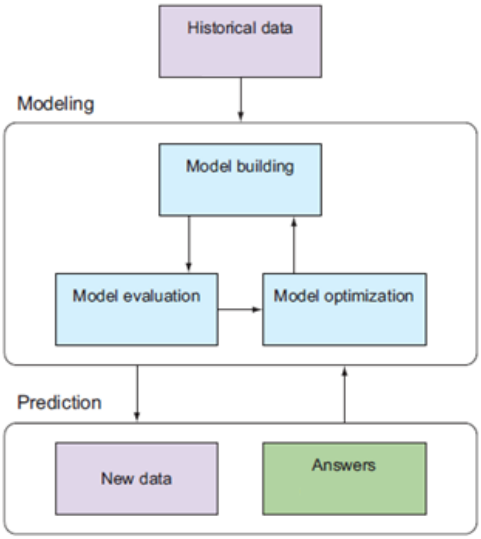


Fig. ML work flow in a real world problem.

Data collection and preparation

Collecting and preparing data for machine-learning systems usually entails getting the data into a tabular format, if it's not already.

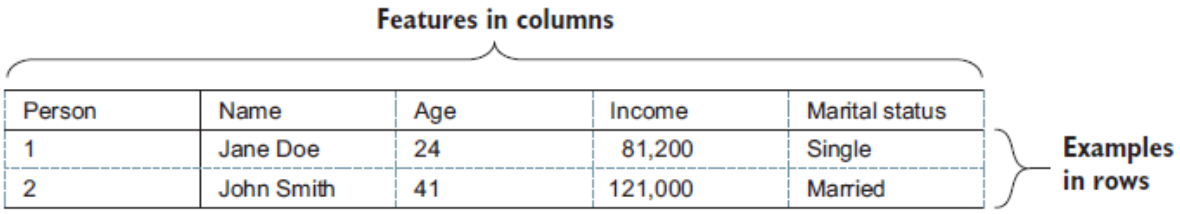


Figure **In a tabular dataset, rows are called *instances* and columns represent *features*.**

Real-world data can be "messy" in a variety of other ways. Suppose that a particular measurement is unavailable for an instance in the data-gathering phase, and there's no way of going back to find the missing piece of information.

In this case, the table will contain a missing value in one or more cells, and this can complicate both model building and subsequent predictions.

Learning a model from data

The first part of building a successful machine-learning system is to ask a question that can be answered by the data. With this simple Person table, you could build an ML model that could predict whether a person is married or single.

In this case, you'd use the Marital status variable as the target, or label, and the remaining variables as features. The job of the ML algorithm will then be to find how the set of input features can successfully predict the target. Then, for people whose marital status is unknown, you can use the model to predict marital status based on the input variables for each individual

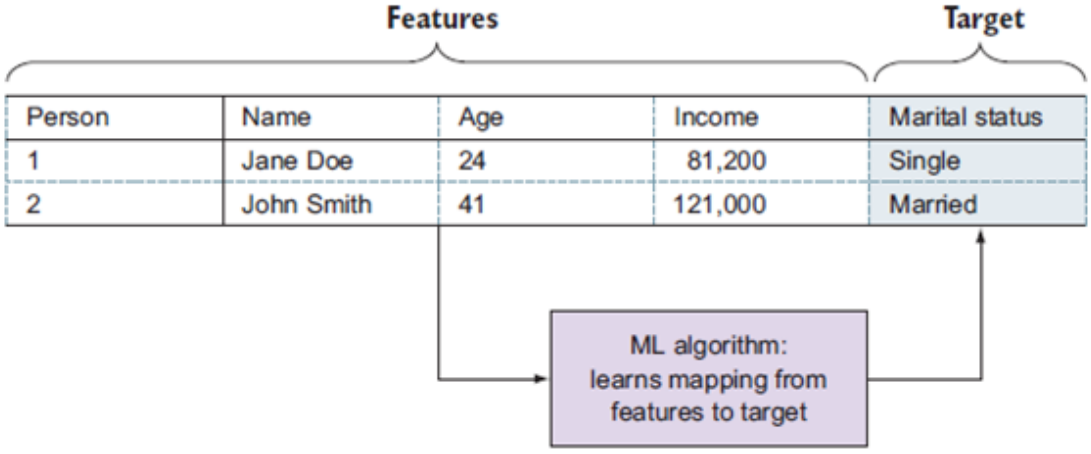


Fig. ML Modeling Process

At this point, think of the ML algorithm as a magical box that performs the mapping from input features to output data.

Using the model for prediction on new data

With our ML model in hand, you can now make predictions on new data—data for which the target variable is unknown

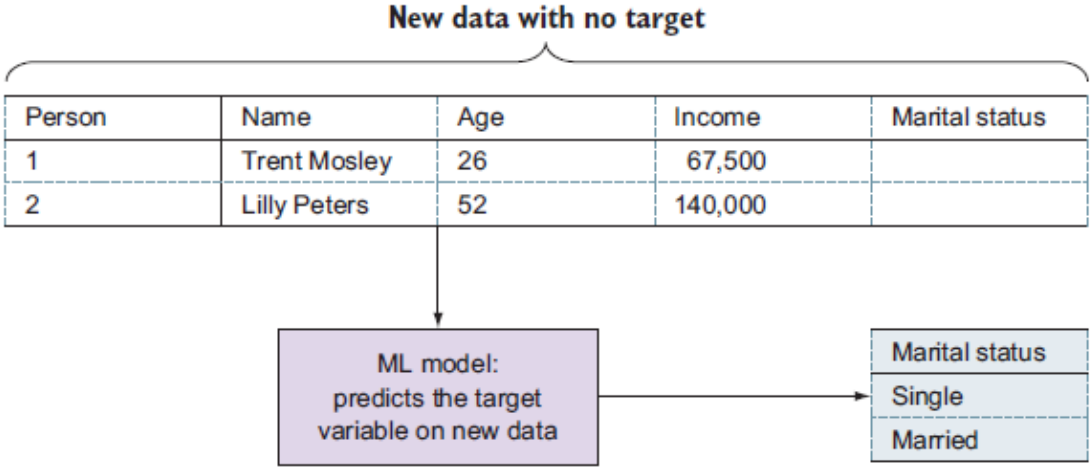


Figure 1.10 Using the model for prediction on new data

Pseudo code:
data = load_data("data/people.csv")
model = build_model(data, target="Marital status")
new_data = load_data("data/new_people.csv")
predictions = model.predict(new_data)

Eg. Decision making for Loan approval

In machine learning, the data provides the foundation for deriving insights about the problem.

To determine whether to accept each new loan application, ML uses historical training data to predict the best course of action for each new application.

To get started with ML for loan approval, you begin by assembling the training data for the 1,000 loans that have been granted. This training data consists of the input data for each loan application, along with the known outcome of whether each loan was repaid on time.

The input data, in turn, consists of a set of features—numerical or categorical metrics that capture the relevant aspects of each application—such as the applicant's credit score, gender, and occupation.

ML modeling, then, determines how the input data for each applicant can be used to best predict the loan outcome. By finding and using patterns in the

training set, ML produces a model that produces a prediction of the outcome for each new applicant, based on that applicant's data.

In below figure, historical data trains the machine-learning model. Then, as new loan applications come in, predictions of the probability of future repayment are generated instantaneously from the application data.

The next step is to select an ML algorithm to use.

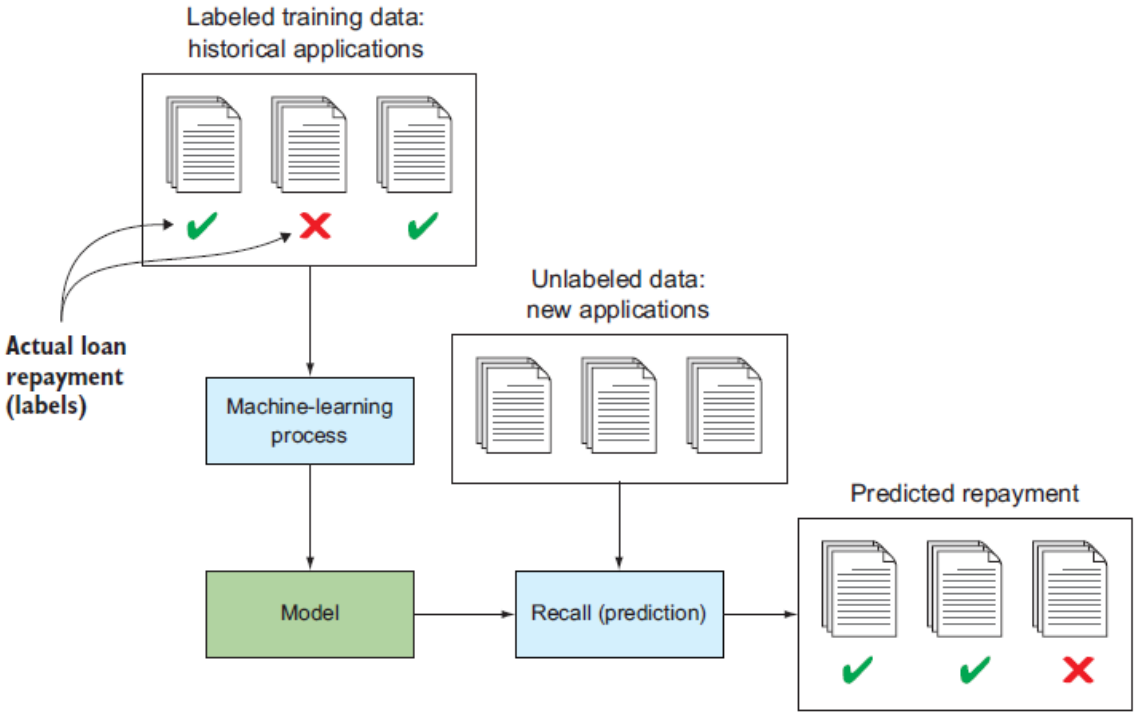


Figure Basic ML workflow, as applied to the microloan example

Advantages:
Accurate—ML uses data to discover the optimal decision-making engine for your problem. As you collect more data, the accuracy can increase automatically.
Automated—As answers are validated or discarded, the ML model can learn new patterns automatically. This allows users to embed ML directly into an automated workflow.
Fast—ML can generate answers in a matter of milliseconds as new data streams in, allowing systems to react in real time.
Customizable— Many data-driven problems can be addressed with machine learning. ML models are custom built from your own data, and can be configured to optimize whatever metric drives your business.
Scalable— As your business grows, ML easily scales to handle increased data rates. Some ML algorithms can scale to handle large amounts of data on many machines in the cloud.

Optimizing model performance

Tuning the model parameters—ML algorithms are configured with parameters specific to the underlying algorithm, and the optimal value of these parameters often depends on the type and structure of the data. The value of each parameter, or any of them combined, can have an impact on the performance of the model.

Selecting a subset of features—Many ML problems include a large number of features, and the noise from those features can sometimes make it hard for the algorithm to find the real signal in the data. You have to carefully determine the features that make up the most general and accurate model.

Preprocessing — the data Most real-world datasets, however, aren't in such a clean state, and you'll have to perform cleaning and processing, a process widely referred to as data munging or data wrangling. The dataset may include names that are spelled differently, although they refer to the same entity, or have missing or incorrect values, and these things can hurt the performance of the model.

Q) What is Natural Language Processing? Explain different steps involved in NLP.

- **NLP** is a subfield of computer science and artificial intelligence concerned with interactions between computers and human (natural) languages.
- It provides ability to a computer to understand, analyze, manipulate, and potentially generate human language.

NLP Components:

- Natural Language Understanding (NLU)

Understanding involves the following tasks −

Mapping the given input natural language into useful representations.
Analyzing different aspects of the language.

- Natural Language Generation (NLG)

It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation. It involves:

Text planning − It includes retrieving the relevant content from knowledge base.

Sentence planning − It includes choosing required words, forming meaningful phrases, setting tone of the sentence.

Text Realization − It is mapping sentence plan into sentence structure.

Ambiguity problem in NLU:

- **Lexical ambiguity** − It is at very primitive level such as word-level.

Eg. She is looking for a *match.*

- **Syntax Level ambiguity** – A sentence can be parsed in different ways.

Eg. “He lifted the beetle with red cap.”

– Did he use cap to lift the beetle or he lifted a beetle that had red cap?

- **Referential ambiguity** – Referring to something using pronouns.

Eg. Rima went to Gauri. She said, “I am tired.”

– Exactly who is tired?

Steps in NLP: There are five steps –

- **Lexical Analysis** – It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis divides the whole chunk of txt into paragraphs, sentences, and words.
- **Syntactic Analysis (Parsing)** – It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words.

  The sentence such as “The school goes to boy” is rejected by English syntactic analyzer.

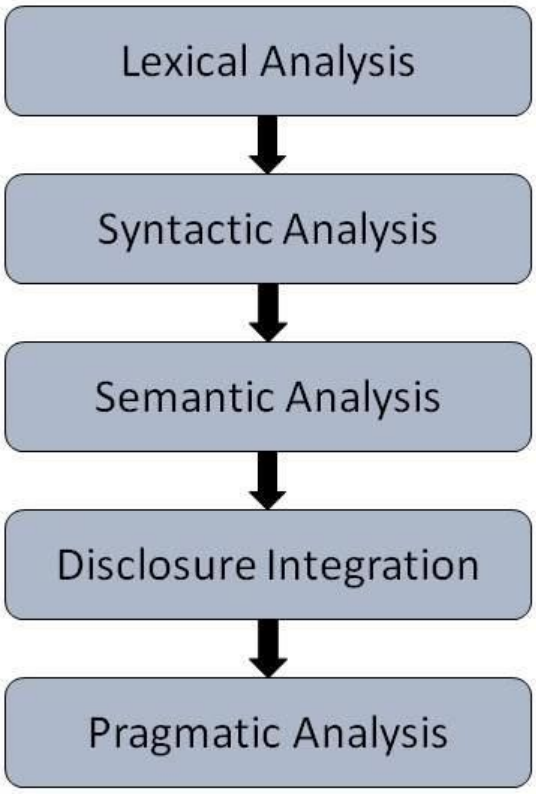


Fig. NLP steps

- **Semantic Analysis** – It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness.

It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as “hot ice-cream”.

- **Discourse Integration** − The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.
- **Pragmatic Analysis** – In this step, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.

Q) List out applications of NLP. Discuss advantages and disadvantages of NLP.

Applications:

- Machine Translation(Google Translate translates language from one language to another).
- Text Simplification(Rewordify simplifies the meaning of sentences).
- Text Summarization(Smmry or Reddit’s autotldr gives a summary of sentences).
- Spam Filter(Gmail filters spam emails separately).
- Sentiment Analysis(Hater News gives us the sentiment of the user).
- Auto-Predict(Google Search predicts user search results).
- Auto-Correct(Google Keyboard and Grammarly correct words otherwise spelled wrong).
- Speech Recognition (Google WebSpeech or Vocalware).
- Question Answering(IBM Watson’s answers to a query).
- Natural Language Generation(Generation of text from image or video data.)

Advantages of NLP

- Users can ask questions about any subject and get a direct response within seconds.
- NLP system provides answers to the questions in natural language
- NLP system offers exact answers to the questions, no unnecessary or unwanted information.
- The accuracy of the answers increases with the amount of relevant information provided in the question.
- NLP process helps computers communicate with humans in their language and scales other language-related tasks.
- Allows you to perform more language-based data compares to a human being without an unbiased way and is consistent.

Disadvantages of NLP

- Complex Query Language- the system may not be able to provide the correct answer it the question that is poorly worded or ambiguous.
- The system is built for a single and specific task only; it is unable to adapt to new domains and problems because of limited functions.

Q) Implement basic operations on text using nltk library.

1. **Sentence Tokenization**
   - Sentence tokenization (also called **sentence segmentation**) is the problem of **dividing a string** of written language **into** its component **sentences**.
   - To apply a sentence tokenization with NLTK we can use the nltk.sent_tokenize function.

Eg.

```
import nltk
nltk.download('punkt')
text = "Hello Mr.Praneeth. How are you?"
sentences = nltk.sent_tokenize(text)
for sentence in sentences:
    print(sentence)
    print()
```

Output:

```
Hello Mr.Praneeth.
How are you?
```

2. **Word Tokenization**
   - Word tokenization (also called **word segmentation**) is the problem of **dividing a string** of written language **into** its component **words**.
   - We can use the nltk.word_tokenize function.

Eg.

```
for sentence in sentences:

    words = nltk.word_tokenize(sentence)
    print(words)
    print()
```

Output:

```
['Hello', 'Mr.Praneeth', '.']
['How', 'are', 'you', '?']
```

3. **Text Lemmatization and Stemming**
   - Stemming is a process where words are reduced to a root by removing inflection(as -s, -ing, -ed) through dropping unnecessary characters, usually a suffix.

Porter Stemming:

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
words = ["go","goes","going","gone","went"]
for word in words:
    print(word,"--",ps.stem(word))
```

Output:

```
go -- go
goes -- goe
going -- go
gone -- gone
went -- went
```

LancasterStemmer:

```
from nltk.stem import LancasterStemmer
ls = LancasterStemmer()
words = ["go","goes","going","gone","went"]
for word in words:
        print(word,"--",ls.stem(word))
```

Output:

```
go -- go
goes -- goe
going -- going
gone -- gon
went -- went
```

- **Lemmatization** is the morphological analysis of the word.
- By determining the part of speech and utilizing WordNet's lexical database of English, lemmatization can get better results.

Eg.

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet


nltk.download('wordnet')

word_lem = WordNetLemmatizer()

words = ["go","goes","going","gone","went"]
for word in words:
    print(word_lem.lemmatize(word))
```

Output:

```
go
go
going
gone
went
```

4. POS: Parts of Speech

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun

Tag	Description
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun
WP$	Possessive whpronoun
WRB	Whadverb

eg

```
words = nltk.word_tokenize("Ramu is a good boy")
tagged = nltk.pos_tag(words)
print(tagged)
```

Output:

```
[('Ramu', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('good', 'JJ'), ('boy', 'NN')]
```

5. Stop words

Stop words are words which are **filtered out** before or after processing of text. When applying machine learning to text, these words can add a lot of **noise**. That's why we want to remove these **irrelevant words**.

Eg.

```
nltk.download("stopwords")
from nltk.corpus import stopwords
print(stopwords.words("english"))
```

Ouput:

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
"you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they',
'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being',
'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but',
'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below',
'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',
'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own',
'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should',
```

"should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

6. **Bag-of-words**

Machine learning algorithms cannot work with raw text directly, we need to convert the text into vectors of numbers. This is called feature extraction. It describes the occurrence of each word within a document.

- To use this model, we need to:
  - Design a **vocabulary** of known words (also called **tokens**)
  - Choose a **measure of the presence** of known words

Bag of Words

- In order to perform machine learning on text, we need to transform our documents into vector representations such that we can apply numeric machine learning.
- This process is called feature extraction or more simply, vectorization.
- Steps to create a bag-of-words model:
  - **Load the Data**
  - **Design the Vocabulary**
  - **Create the Document Vectors**

- Types of vectorizer
  - **Count vectorizer**
  - **Tf-Idf Vectorizer**

- **Count Vectorizer** gives equal weightage to all the words, i.e. a word is converted to a column for each document, it is equal to 1 if it is present in that doc, else 0.
- Apart from giving this information, tfidf says how important that word is to that document with respect to the corpus.

Eg. Count Vectorizer

```
# Step 1. Gathering data
with open("simple_movie_reviews.txt", "r") as file:
   documents = file.read().splitlines()

print(documents)
```

Output:

["I like this movie, it's funny funny.", 'I hate this movie.', 'This was awesome! I like it.', 'Nice one. I love it.']

```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

# Step 2. Design the Vocabulary
```

```
# The default token pattern removes tokens of a single character. That's why
#we don't have the "I" and "s" tokens in the output
count_vectorizer = CountVectorizer()

# Step 3. Create the Bag-of-Words Model
bag_of_words = count_vectorizer.fit_transform(documents)

# Show the Bag-of-Words Model as a pandas DataFrame
feature_names = count_vectorizer.get_feature_names()
pd.DataFrame(bag_of_words.toarray(), columns = feature_names)
```

Output:

	awesome	funny	hate	it	like	love	movie	nice	one	this	was
0	0	2	0	1	1	0	1	0	0	1	0
1	0	0	1	0	0	0	1	0	0	1	0
2	1	0	0	1	1	0	0	0	0	1	1
3	0	0	0	1	0	1	0	1	1	0	0

Tf-Idf has two parts

- Term Frequency
- Inverse Document Frequency

Term Frequency (Tf): How many times a particular word appears in a single doc.

Eg.

tf (“the”) = 100 i.e; word the appears for 100 times
tf(“messi”) = 5
tf(“divertissement”) = 1

- **Inverse Document Frequency (idf):** It is calculated by taking the **log of {number of docs in your corpus divided by the number of documents in which this term appears}.**

  Eg.
- idf (“the”) = log(10/10) = 0 #here word the appears for 10 times in 10 documents.
- idf (“messi”) = log(10/3) = 0.52
- idf (“divertissement”) = log(10/1) = 1

- **tfidf = tf * idf**
  - "the" = 100 * 0 = 0
  - " divertissement" = 1*1 = 1
  - "messi" = 5*.52 = 2.6

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
```

```
tfidf_vectorizer = TfidfVectorizer()
values = tfidf_vectorizer.fit_transform(documents)

# Show the Model as a pandas DataFrame
feature_names = tfidf_vectorizer.get_feature_names()
pd.DataFrame(values.toarray(), columns = feature_names)
```

Output:

	awesome	funny	hate	it	like	love	movie	nice	one	this	was
0	0.000000	0.571848	0.000000	0.365003	0.450852	0.000000	0.450852	0.000000	0.000000	0.365003	0.000000
1	0.000000	0.000000	0.702035	0.000000	0.000000	0.000000	0.553492	0.000000	0.000000	0.448100	0.000000
2	0.539445	0.000000	0.000000	0.344321	0.425305	0.000000	0.000000	0.000000	0.000000	0.344321	0.539445
3	0.000000	0.000000	0.000000	0.345783	0.000000	0.541736	0.000000	0.541736	0.541736	0.000000	0.000000

7. Regular Expressions: **regexp** is a sequence of characters that define a **search pattern**.

Metacharacters

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"

Special Sequences

Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"
\b	Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\Bain" r"ain\B"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"

Sets

Set	Description
[arn]	Returns a match where one of the specified characters (a , r , or n) are present
[a-n]	Returns a match for any lower case character, alphabetically between a and n
[^arn]	Returns a match for any character EXCEPT a , r , and n
[0123]	Returns a match where any of the specified digits (0 , 1 , 2 , or 3) are present
[0-9]	Returns a match for any digit between 0 and 9
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a and z , lower case OR upper case
[+]	In sets, + , * , . , | , () , $, {} has no special meaning, so [+] means: return a match for any + character in the string

Eg.
#1. Replace every white-space character with the number 7
```
import re

txt = "Now it is raining in vijayawada"
x = re.sub("\s", "p", txt)
print(x)
```

Output:
Nowpitpisprainingpinpvijayawada

#2. finding words that start with a or b or c or d or e and ends with at
```
text = "A fat cat doesn't eat oat but a rat eats bats."
res = re.findall("[a-e]at", text)
print(res)
```

Output:
['cat', 'eat', 'eat', 'bat']

Q) Define Speech Recognition. Explain different types of speech recognition systems.

Speech recognition is an interdisciplinary subfield of computer science and computational linguistics that develops methodologies and technologies that enable the recognition and translation of spoken language into text by computers. It is also known as **automatic speech recognition** (**ASR**), **computer speech recognition** or **speech to text** (**STT**).

Speech recognition is the process by which computer maps an acoustic speech signal to some form of abstract meaning of the speech.

The fundamental aspect of speech recognition is the translation of sound into text and commands.

This process is highly difficult since sound has to be matched with stored sound bites on which further analysis has to be done because sound bites do not match with pre-existing sound pieces.

Feature extraction technique and pattern matching techniques plays important role in speech recognition system to maximize the rate of speech recognition of various persons.

Types of speech recognition system based on utterances

1. Isolated Words

Isolated word recognition system which recognizes single utterances i.e. single word. Isolated word recognition is suitable for situations where the user is required to give only one word response or commands.

It is simple and easiest for implementation because word boundaries are obvious and the words tend to be clearly pronounced which is the major advantage of this type.

Eg. Yes/No, Listen/Not-Listen

2. Connected Words

A connected words system is similar to isolated words, but it allows separate utterances to be "run-together" with a minimal pause between them. Utterance is the vocalization of a word or words that represent a single meaning to the computer.

Eg. drive car(action + object), baby book(agent + object)

3. Continuous Speech

Continuous speech recognition system allows users to speak almost naturally, while the computer determines its content. Continuous speech recognition system is difficult to develop.

Eg. dictation

4. Spontaneous Speech

Spontaneous speech recognition system recognizes the natural speech. Spontaneous speech is natural that comes suddenly through mouth. An ASR system with spontaneous speech is able to handle a variety of natural speech features such as words being run together. Spontaneous speech may include mispronunciation, false-starts and non words.

Eg. Spontaneous speech may include mispronunciation, false-starts(will you – will you go to your college?) and non words("ums").

Types of speech recognition based on Speaker Model

1. Speaker Dependent Models

- Speaker dependent systems are developed for a particular type of speaker. They are generally more accurate for the particular speaker, but could be less accurate for other type of speakers.
- These systems are usually cheaper, easier to develop and more accurate. But these systems are not flexible as speaker independent systems.

2. Speaker Independent Models

- Speaker Independent system can recognize a variety of speakers without any prior training.
- A speaker independent system is developed to operate for any particular type of speaker.
- It is used in Interactive Voice Response System (IVRS) that must accept input from a large number of different users.
- But drawback is that it limits the number of words in a vocabulary. Implementation of Speaker Independent system is the most difficult. Also it is expensive and its accuracy is lower than speaker dependent systems.

3. Speaker Adaptive Models

Speaker adaptive speech recognition system uses the speaker dependent data and adapt to the best suited speaker to recognize the speech and decreases error rate by adaption. They adapt operation according to characteristics of speakers.

Q) Explain about Automatic Speech Recognition System

Speech recognition is an interdisciplinary subfield of computer science and computational linguistics that develops methodologies and technologies that enable the recognition and translation of spoken language into text by computers. It is also known as **automatic speech recognition** (**ASR**), **computer speech recognition** or **speech to text** (**STT**).

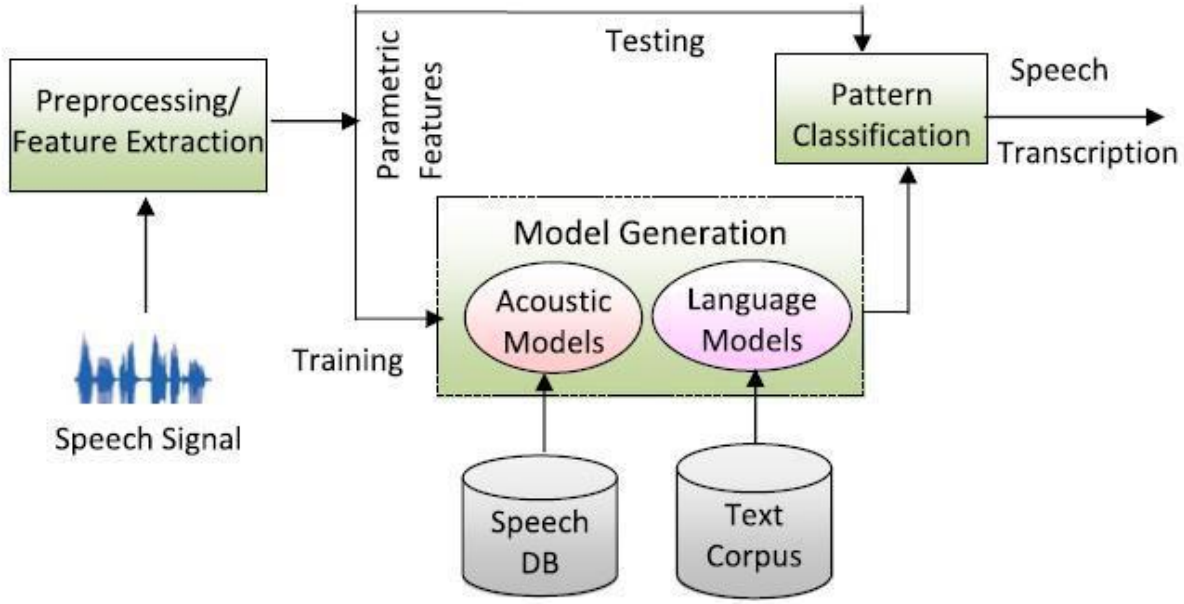


Fig. System Architecture of for Automatic Speech Recognition System

Pre-processing/Digital Processing

The recorded acoustic signal is an analog signal. An analog signal cannot directly transfer to the ASR systems. So these speech signals need to transform in the form of digital signals and then only they can be processed. These digital signals are move to the first order filters to spectrally flatten the signals. This procedure increases the energy of signal at higher frequency.

Feature Extraction

Feature extraction step finds the set of parameters of utterances that have acoustic correlation with speech signals and these parameters are computed through processing of the acoustic waveform. These **These parameters are known as features or feature vectors (x_1, x_2, x_3, ...).**

The main focus of feature extractor is to keep the relevant information and discard irrelevant one.

Feature extractor divides the acoustic signal into 10-25 ms. Data acquired in these frames is multiplied by window function. There are many types of window functions that can be used such as hamming Rectangular, Blackman, Welch or Gaussian etc.

Feature extraction methods : Principal component Analysis (PCA), Linear Discriminate Analysis(LDA) , Wavelet, Independent component Analysis(ICA) etc.

Acoustic Modeling

Acoustic modeling is the fundamental part of ASR system. In acoustic modeling, the connection between the acoustic information and phonetics is established.

Training establishes co-relation between the basic speech units and the acoustic observations.

Training of the system requires creating a pattern representative for the features of class using one or more patterns that correspond to speech sounds of the same class.

Many models are available for acoustic modeling out of them Hidden Markov Model (HMM) is widely used and accepted as it is efficient algorithm for training and recognition.

Language Modeling

A language model contains the structural constraints available in the language to generate the probabilities of occurrence. **It induces the probability of a word occurrence after a word sequence.**

Each language has its own constraints. Generally Speech recognition systems uses bi-gram, tri-gram, n-gram language models for finding correct word sequence by predicting the likelihood of the nth word, using the n-1 earlier words.

In speech recognition, the computer system matches sounds with word sequence. The language model distinguishes word and phrase that has similar sound.

Eg. In American English, the phrases like “recognize speech" and "wreck a nice beach" have same pronunciation but mean very different things.

Pattern Classification

Pattern Classification (or recognition) is the process of **comparing the unknown test pattern with each sound class reference pattern** and computing a measure of similarity between them. After completing training of the system at the time of testing patterns are classified to recognize the speech.

Challenges in ASR:

- Acoustic modeling plays a critical role to improve the accuracy.
- The most noticeable ones are context variations, speaker variations, and environment variations.