6. Applications of AI

Course Code: AFI 124

Previous Class

Learning with Neural Networks:

- Introduction: Deep Learning, Neural Network, Artificial Neural Network
- Biological Neural Networks Vs. Artificial Neural Networks (ANN),
- Example of Neural Network: House Price Prediction
- Activation Functions: Sigmoid, Tanh, ReLu
- Types of ANN: Feed-forward, Single Layered, Multi-Layered
- Application of Artificial Neural Networks,
- Learning Techniques in Neural Networks
- Perceptron Learning,
- Back - propagation Learning
- Transfer Learning

Contents

VI Communicating, perceiving, and acting

- **Natural Language Processing**: Language Models, Grammar, Parsing, Augmented Grammars, Complications of Real Natural Language
- Natural Language Tasks

Natural Language Processing

Natural language processing is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. More like giving computers the ability to understand text and spoken words in much the same way human beings can.

3 Primary reasons for computers to do NLP

1. To **communicate** with humans. In many situations it is convenient for humans to use speech to interact with computers, and in most situations it is more convenient to use natural language rather than a formal language such as first-order predicate calculus.
2. To **learn**. Humans have written down a lot of knowledge using natural language. Wikipedia alone has 30 million pages of facts such as “Bush babies are small nocturnal primates,” whereas there are hardly any sources of facts like this written in formal logic. If we want our system to know a lot, it had better understand natural language.
3. To **advance the scientific understanding of languages** and language use, using the tools of AI in conjunction with linguistics, cognitive psychology, and neuroscience.

Language Models

A language model is a probability distribution over sequences of words. Given such a sequence of length m, a language model assigns a probability $P(w_1, \ldots, w_m)$ to the whole sequence. Language models generate probabilities by training on text corpora in one or many languages.

We define a language model as a probability distribution describing the likelihood of any string. Such a model should say that “Do I dare disturb the universe?” has a reasonable probability as a string of English, but “Universe dare the I disturb do?” is extremely unlikely.

Applications of Language model

- Predict Next word: **suggest completions for an email** or text message
- Compute which alterations to a text would make it more probable: **suggest spelling** or **grammar corrections**.
- With a pair of models, we can compute the most probable **translation** of a sentence: **Machine Translation**
- With some example question/answer pairs as training data, we can compute the most likely answer to a question: **Chatbots**

The Bag of words Model

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

Bag of words Model

```
Let’s say we have 2 text document:

Doc1: John likes to watch movies. Mary likes movies too.
Doc2: Mary also likes to watch football games.
```

Based on these two text documents, a list is constructed as follows for each document:

```
"John","likes","to","watch","movies","Mary","likes","movies","too"
"Mary","also","likes","to","watch","football","games"
```

Forming a Count Dictionary

```
BoW1 = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1};
BoW2 = {"Mary":1,"also":1,"likes":1,"to":1,"watch":1,"football":1,"games":1};
```

Get list of unique words

John, likes, to, watch, movies, Mary, too, also, football, games

Form a Matrices

words	**John**	**likes**	**to**	**watch**	**movies**	**Mary**	**too**	**also**	**football**	**games**
Doc1	1	2	1	1	2	1	1	0	0	0
Doc2	0	1	1	1	0	1	0	1	1	1

```
So, Vector generated are:
Vec1 = [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
Vec2 = [0, 1, 1, 1, 0, 1, 0, 1, 1, 1]
```

N-Gram word Model

The bag-of-words model has limitations. For example, the word “quarter” is common in both the and categories. But the four-word sequence “first quarter earnings report” is common only in and “fourth quarter touchdown passes” is common only in business and “fourth quarter touchdown passes” is common only in sports.

A sequence of written symbols of length is called an -gram, with special cases “**unigram**” for 1-gram, “**bigram**” for 2-gram, and “**trigram**” for 3-gram.

N-gram models work well for:

- classifying newspaper sections
- spam detection (distinguishing spam email from non-spam)
- sentiment analysis
- classifying a movie reviews
- product review as positive or negative)
- author attribution (Hemingway has a different style and vocabulary than Faulkner or Shakespeare).

N-Gram Model

["John likes", "likes to", "to watch", "watch movies", "Mary likes", "likes movies", "movies too",]

N-Gram Model

```
Taking the same 2 text document:

Doc1: John likes to watch movies. Mary likes movies too.
Doc2: Mary also likes to watch football games.
```

Based on these two text documents, a list is constructed as follows for each document using bigrams:

```
"John likes","likes to","to watch","watch movies","Mary likes","likes movies","movie too"
"Mary also","also likes","likes to","to watch","watch football","football games"
```

Forming a Count Dictionary

```
BoW1 = {"John likes":1, "likes to":1, "to watch":1, "watch movies":1, “Mary likes":1, "likes movies":1, "movies too":1,  “Mary
also”:0, “also likes":0, "watch football":0, "football games":0}
BoW2 = {"Mary also":1,"also likes":1,"likes to":1,"to watch":1,"watch football":1,"football games":1}
```

Get list of unique words

["John likes", "likes to", "to watch", "watch movies", “"Mary likes", "likes movies", "movies too", “Mary also”, “also likes", "watch football", "football games"]

Form a Matrices

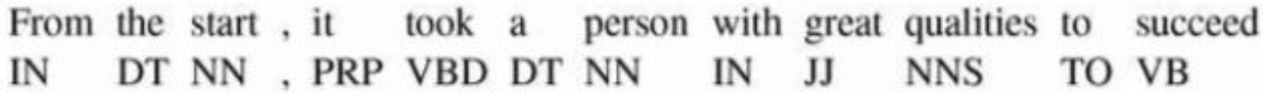
words	**John likes**	**Likes to**	**To watch**	**watch movies**	**Mary likes**	**Likes movies**	**Movies too**	**Mary also**	**Also likes**	**Watch football**	**Football games**
Doc1	1	1	1	1	1	1	1	0	0	0	0
Doc2	0	1	1	0	0	0	0	1	1	1	1

```
So, Vector generated are:
Vec1 = [1, 1, 1, 1, 1, 1, 1, 0, 0, 0,0]
Vec2 = [0, 1, 1, 0, 0, 0, 0, 1, 1, 1,1]
```

Part-of-speech (POS) tagging

- One basic way to categorize words is by their part of speech (POS), also called lexical category or tag: noun, verb, adjective, and so on.

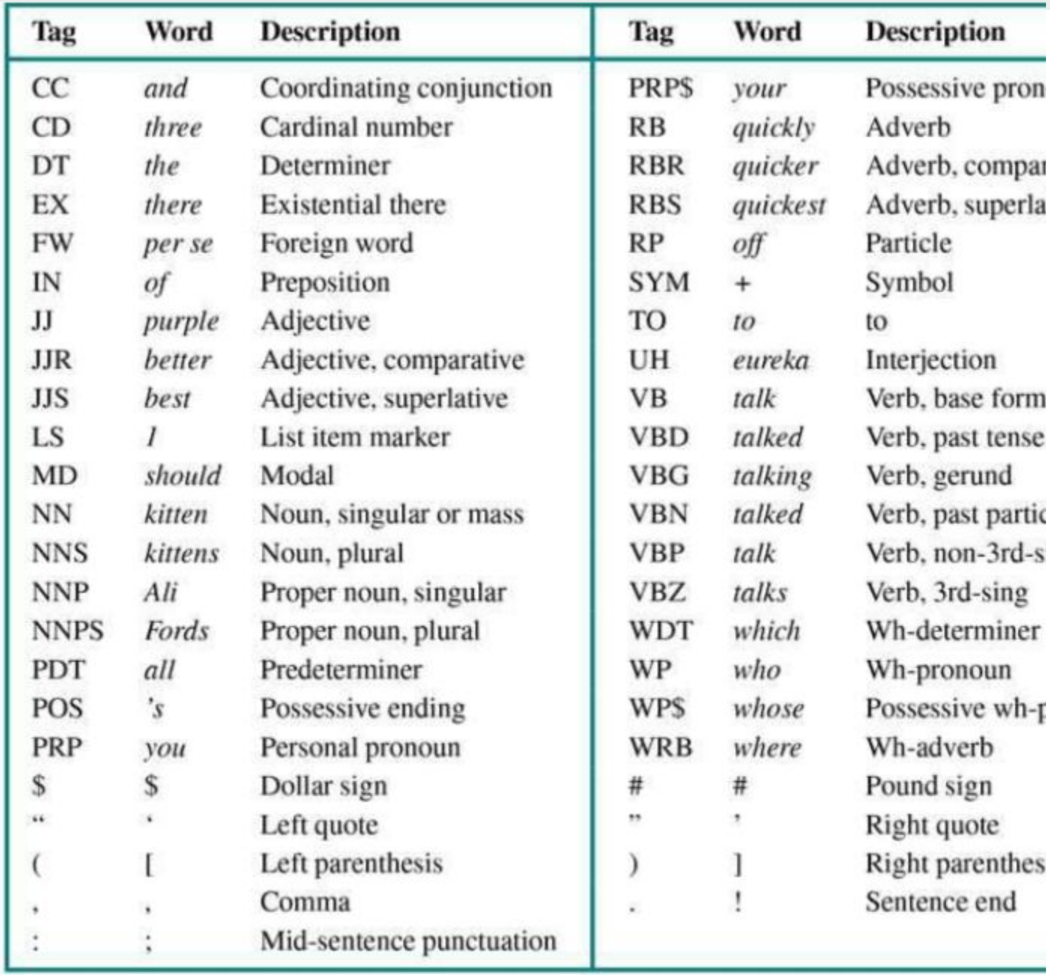
- Penn Treebank, a corpus of over three million words of text annotated with part-of-speech tags.



From	the	start	,	it	took	a	person	with	great	qualities	to	succeed
IN	DT	NN	,	PRP	VBD	DT	NN	IN	JJ	NNS	TO	VB

- The task of assigning a part of speech to each word in a sentence is called part-of-speech tagging

Tags



Tag	Word	Description	Tag	Word	Description
CC	*and*	Coordinating conjunction	PRP$	*your*	Possessive pron
CD	*three*	Cardinal number	RB	*quickly*	Adverb
DT	*the*	Determiner	RBR	*quicker*	Adverb, compa
EX	*there*	Existential there	RBS	*quickest*	Adverb, superla
FW	*per se*	Foreign word	RP	*off*	Particle
IN	*of*	Preposition	SYM	+	Symbol
JJ	*purple*	Adjective	TO	*to*	to
JJR	*better*	Adjective, comparative	UH	*eureka*	Interjection
JJS	*best*	Adjective, superlative	VB	*talk*	Verb, base form
LS	*1*	List item marker	VBD	*talked*	Verb, past tense
MD	*should*	Modal	VBG	*talking*	Verb, gerund
NN	*kitten*	Noun, singular or mass	VBN	*talked*	Verb, past partic
NNS	*kittens*	Noun, plural	VBP	*talk*	Verb, non-3rd-s
NNP	*Ali*	Proper noun, singular	VBZ	*talks*	Verb, 3rd-sing
NNPS	*Fords*	Proper noun, plural	WDT	*which*	Wh-determiner
PDT	*all*	Predeterminer	WP	*who*	Wh-pronoun
POS	*'s*	Possessive ending	WP$	*whose*	Possessive wh-p
PRP	*you*	Personal pronoun	WRB	*where*	Wh-adverb
$	$	Dollar sign	#	#	Pound sign
“	‘	Left quote	”	’	Right quote
(	[	Left parenthesis	)	]	Right parenthes
,	,	Comma	.	!	Sentence end
:	;	Mid-sentence punctuation			

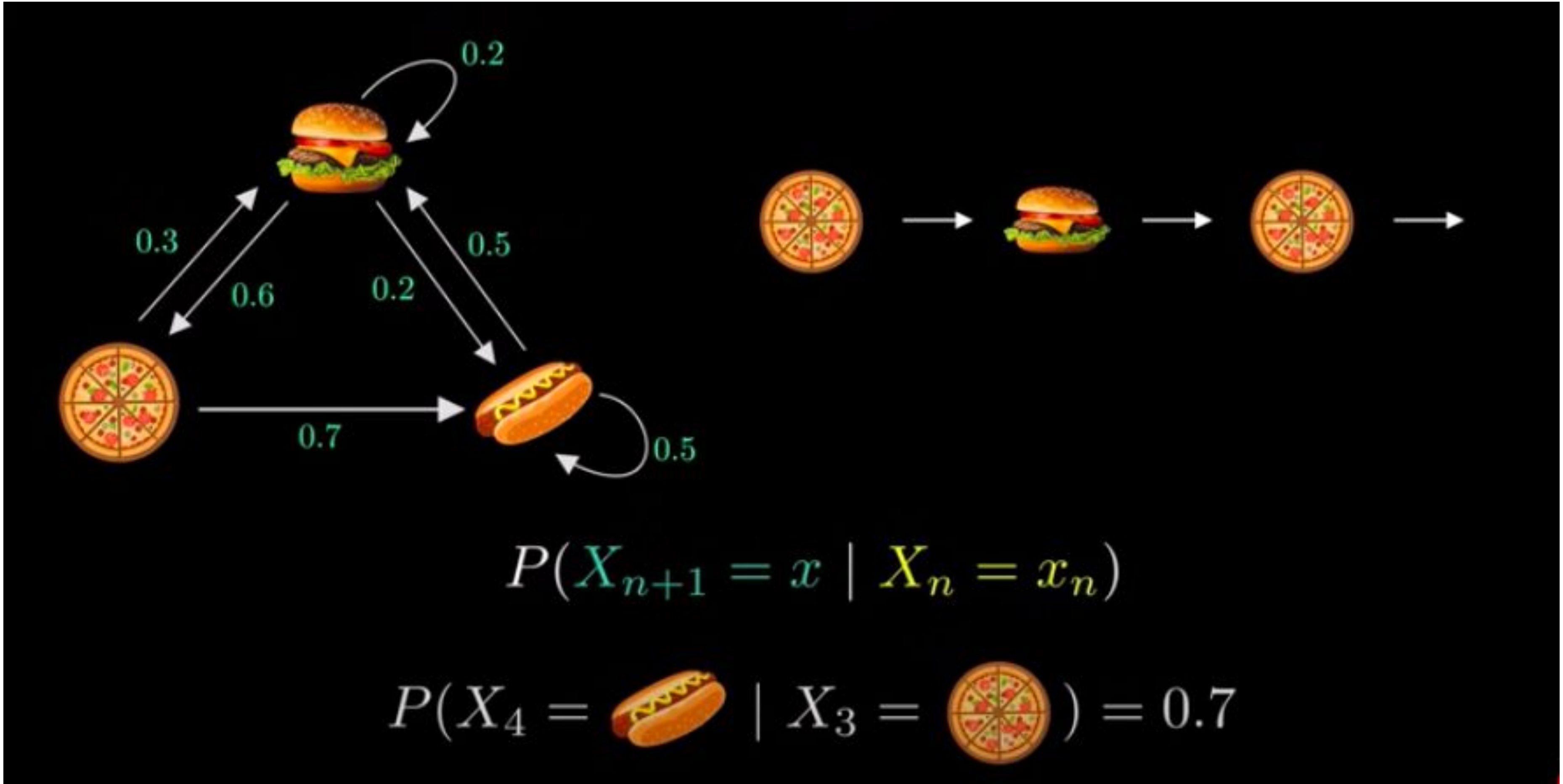
Hidden Markov Model

- Predicts the next state based on the current state.
- HMM uses Markov Chain.

Markov Chain: https://www.youtube.com/watch?v=i3AkTO9HLXo&list=PLoD3ZpkiaA_pwMrAMlES6CGBnS1UBOVpk&index=2&t=16s

HMM: https://www.youtube.com/watch?v=RWkHJnFj5rY

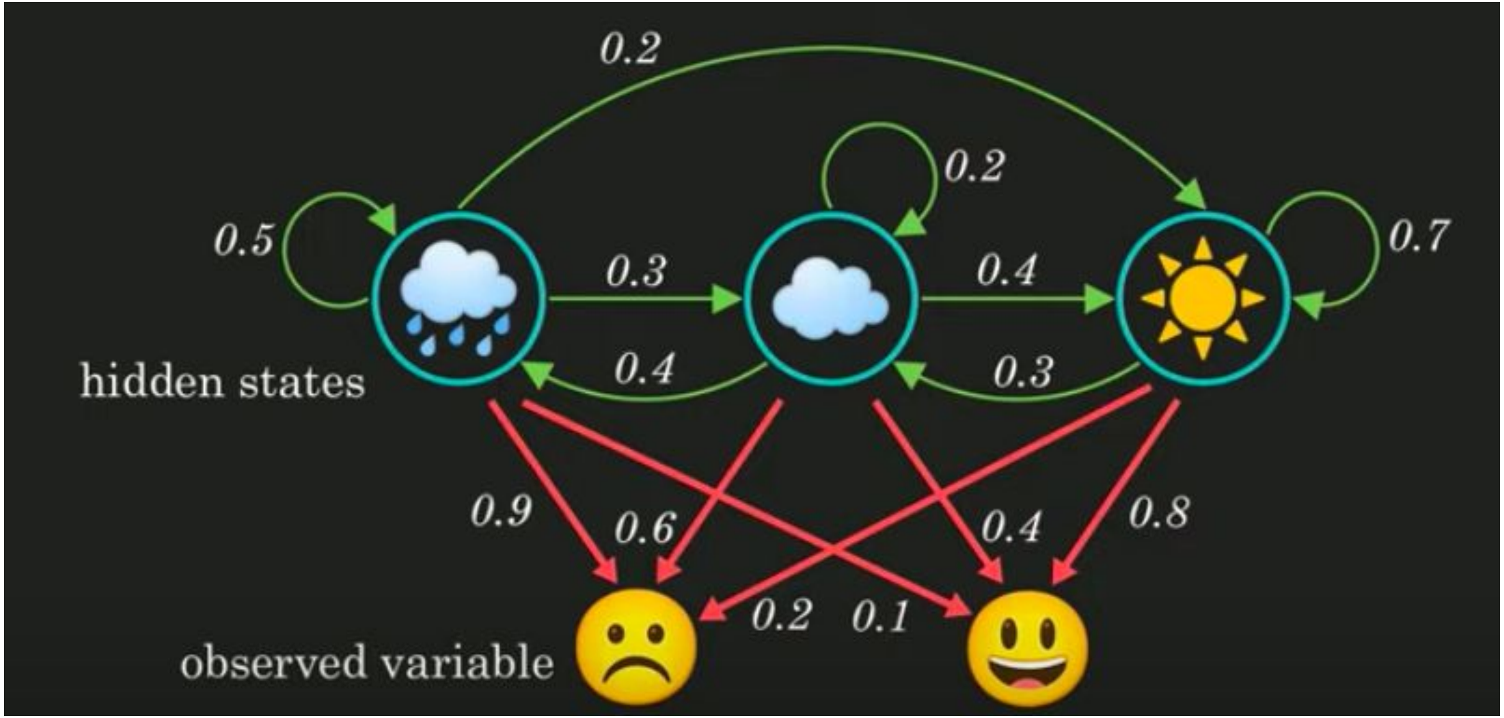
Markov Chain Example



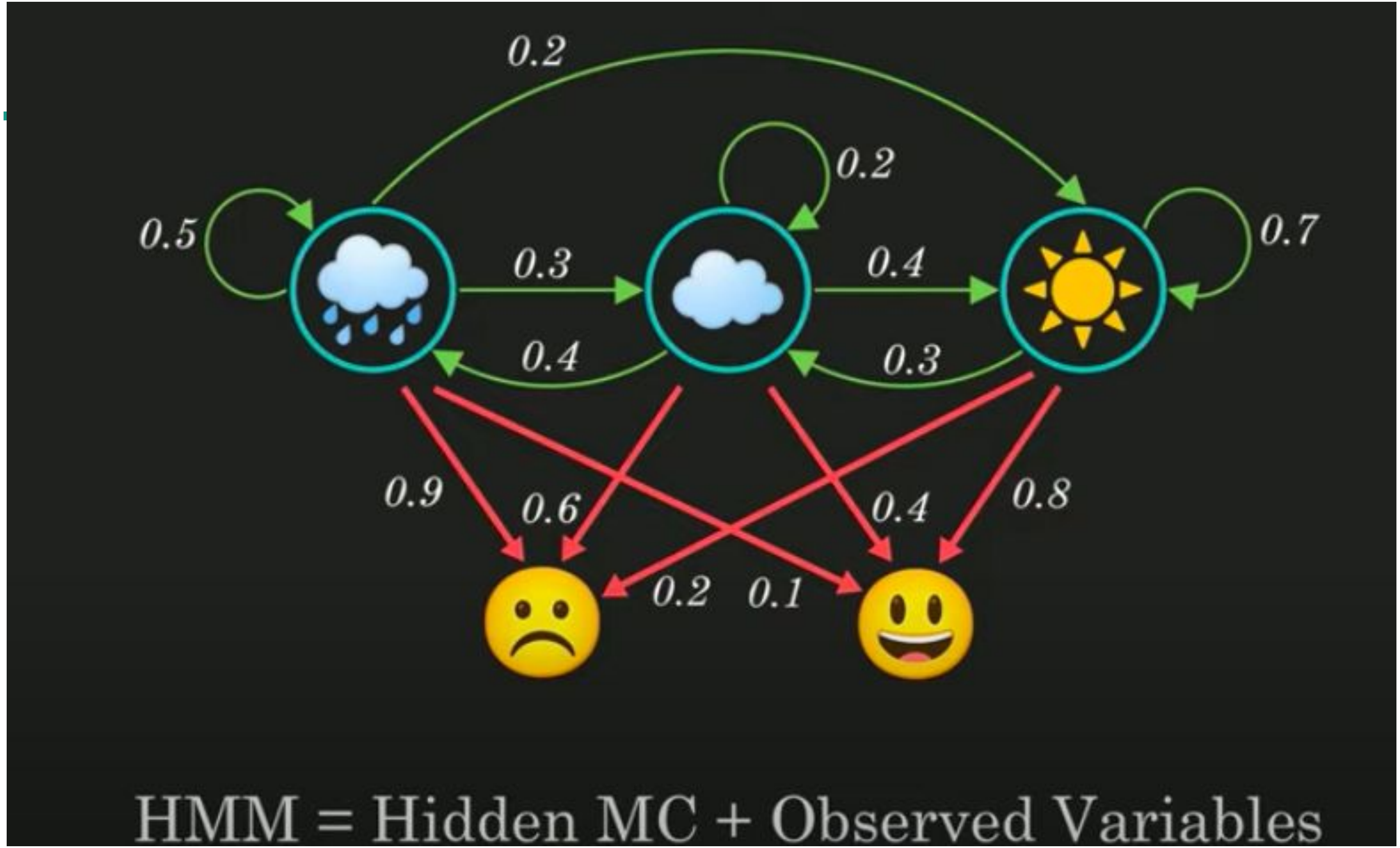
$$P(X_{n+1} = x \mid X_n = x_n)$$

$$P(X_4 = 🌭 \mid X_3 = 🍕) = 0.7$$

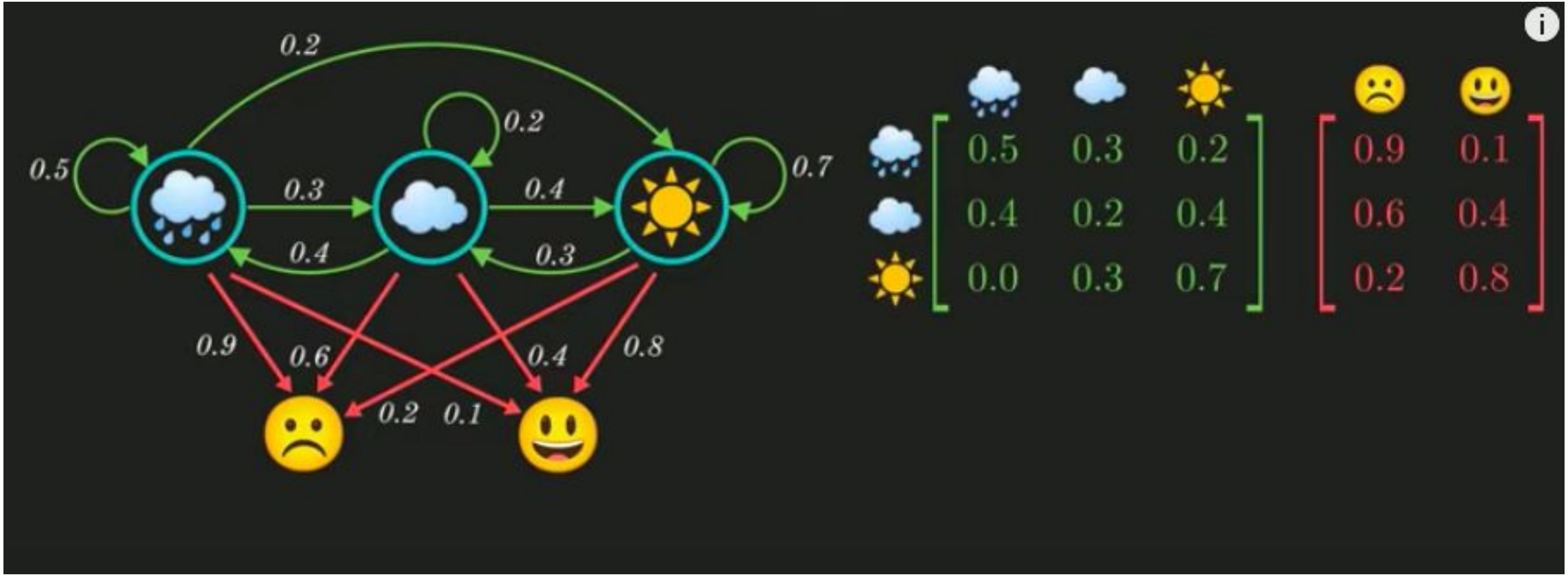
Hidden Markov Model Example



Hidden Markov Model Example

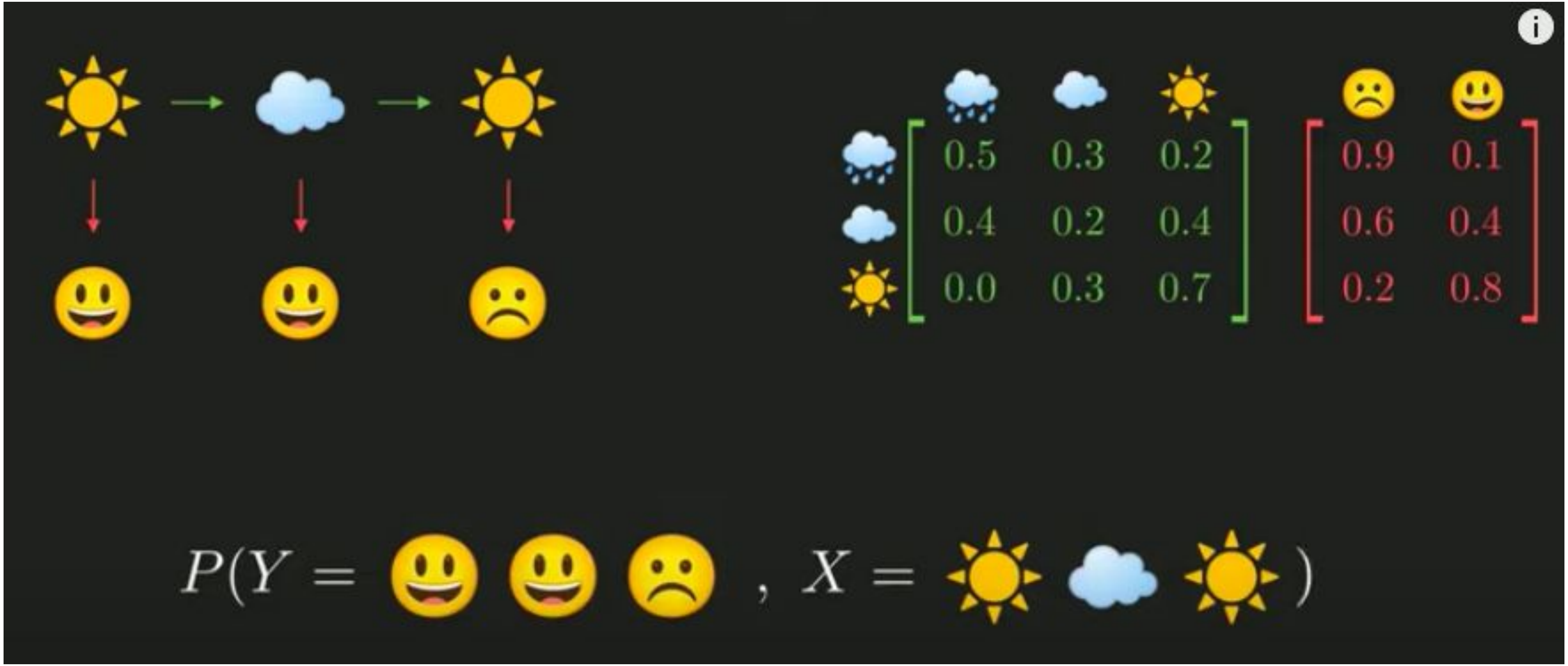


Hidden Markov Model Example



$$\begin{array}{c|ccc} & \text{🌧} & \text{☁} & \text{☀} \\ \hline \text{🌧} & 0.5 & 0.3 & 0.2 \\ \text{☁} & 0.4 & 0.2 & 0.4 \\ \text{☀} & 0.0 & 0.3 & 0.7 \end{array} \qquad \begin{array}{cc} \text{☹} & \text{😃} \\ \hline 0.9 & 0.1 \\ 0.6 & 0.4 \\ 0.2 & 0.8 \end{array}$$

Hidden Markov Model Example

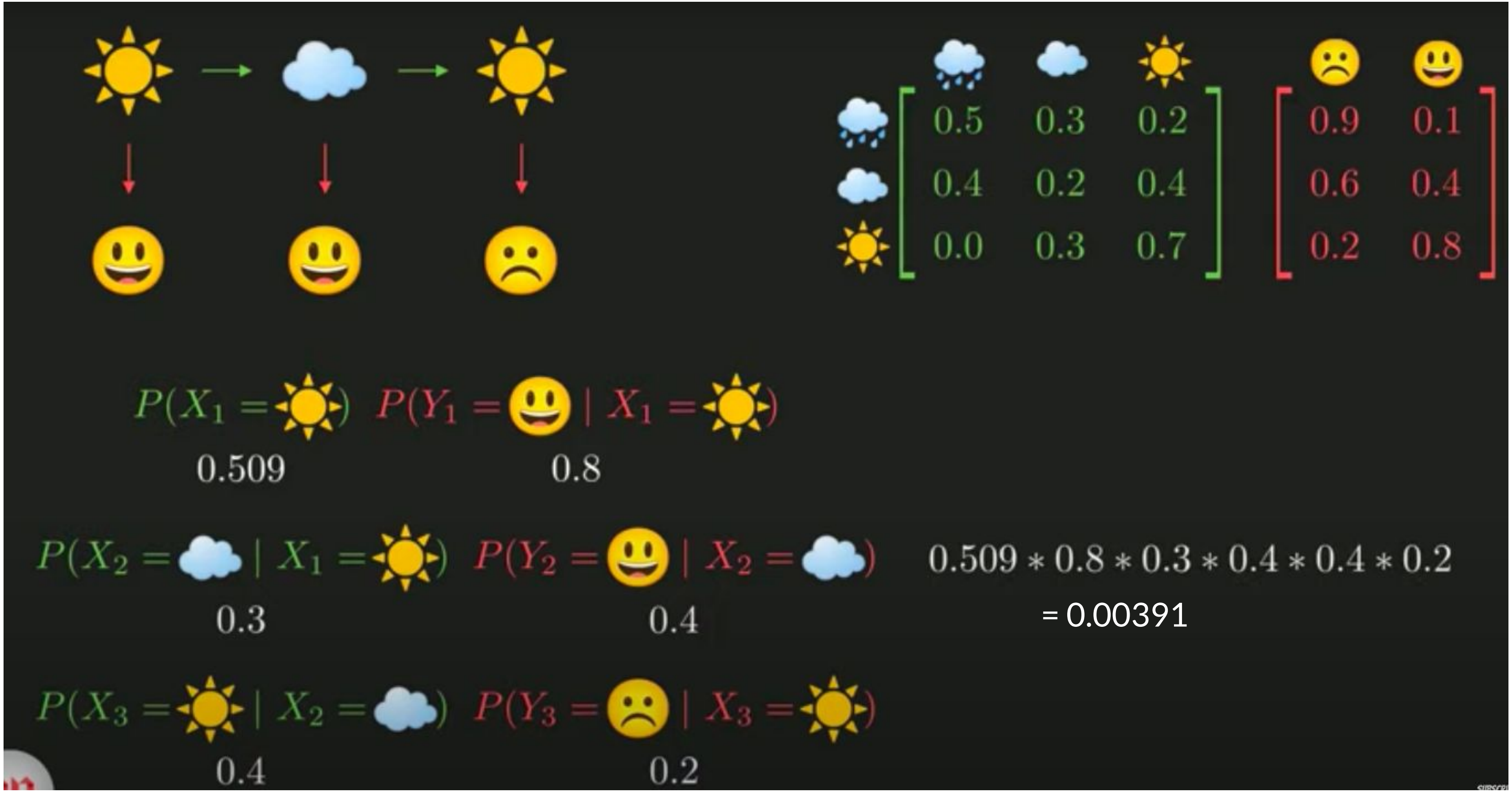


	🌧️	☁️	☀️
🌧️	0.5	0.3	0.2
☁️	0.4	0.2	0.4
☀️	0.0	0.3	0.7

☹️	😃
0.9	0.1
0.6	0.4
0.2	0.8

$P(Y =$ 😃 😃 ☹️ $, X =$ ☀️ ☁️ ☀️ $)$

Hidden Markov Model Example



$$\begin{tikzcd}
\text{☀} \arrow[r] \arrow[d] & \text{☁} \arrow[r] \arrow[d] & \text{☀} \arrow[d] \\
\text{😃} & \text{😃} & \text{☹}
\end{tikzcd}$$

$$\begin{array}{c|ccc}
 & \text{🌧} & \text{☁} & \text{☀} \\
\hline
\text{🌧} & 0.5 & 0.3 & 0.2 \\
\text{☁} & 0.4 & 0.2 & 0.4 \\
\text{☀} & 0.0 & 0.3 & 0.7
\end{array}
\qquad
\begin{array}{cc}
\text{☹} & \text{😃} \\
\hline
0.9 & 0.1 \\
0.6 & 0.4 \\
0.2 & 0.8
\end{array}$$

$P(X_1 = \text{☀})$ $P(Y_1 = \text{😃} \mid X_1 = \text{☀})$

0.509 0.8

$P(X_2 = \text{☁} \mid X_1 = \text{☀})$ $P(Y_2 = \text{😃} \mid X_2 = \text{☁})$

0.3 0.4

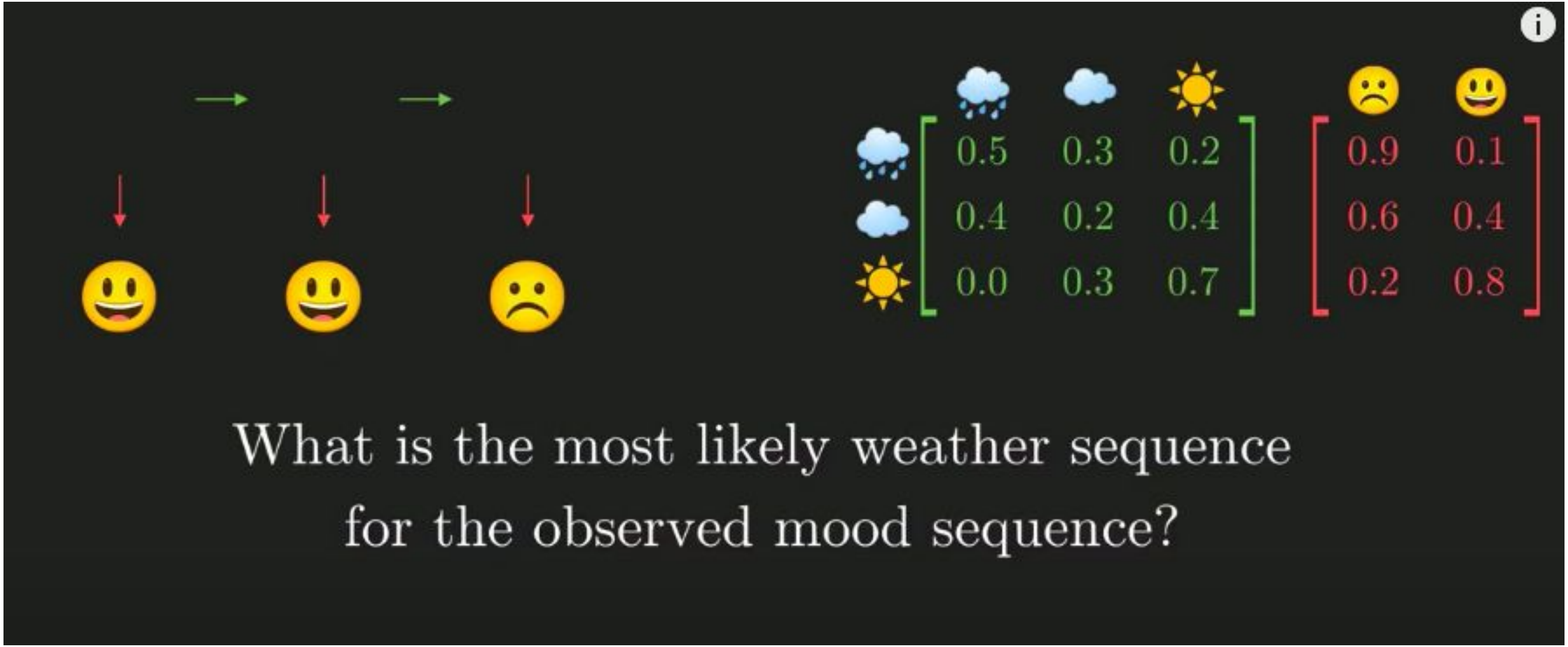
$$0.509 * 0.8 * 0.3 * 0.4 * 0.4 * 0.2$$

= 0.00391

$P(X_3 = \text{☀} \mid X_2 = \text{☁})$ $P(Y_3 = \text{☹} \mid X_3 = \text{☀})$

0.4 0.2

Hidden Markov Model Example



$$\begin{array}{c|ccc} & \text{🌧️} & \text{☁️} & \text{☀️} \\ \hline \text{🌧️} & 0.5 & 0.3 & 0.2 \\ \text{☁️} & 0.4 & 0.2 & 0.4 \\ \text{☀️} & 0.0 & 0.3 & 0.7 \end{array} \qquad \begin{array}{cc} \text{☹️} & \text{😃} \\ \hline 0.9 & 0.1 \\ 0.6 & 0.4 \\ 0.2 & 0.8 \end{array}$$

What is the most likely weather sequence for the observed mood sequence?

For the answer, watch https://www.youtube.com/watch?v=RWkHJnFj5rY

Grammar

A grammar is a set of rules that defines the tree structure of allowable phrases, and a language is the set of sentences that follow those rules.

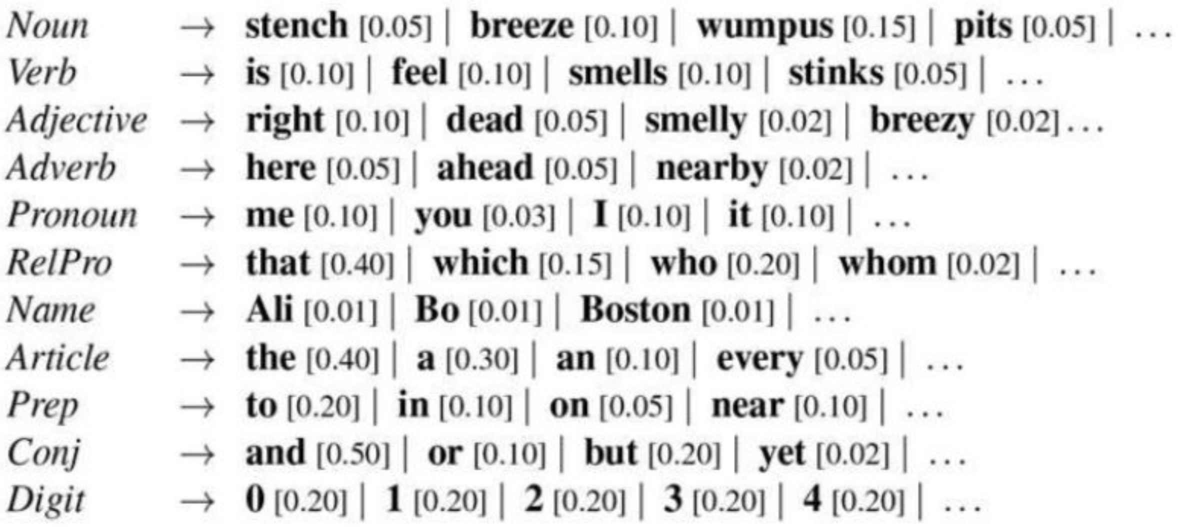
Syntactic categories such as noun phrase or verb phrase help to constrain the probable words at each point within a sentence, and the phrase structure provides a framework for the meaning or **semantics** of the sentence.

Example: The sentence “I ate a banana” is fine, but “Me ate a banana” is ungrammatical, and “I ate a bandanna” is unlikely. Here, “I ate a bandanna” is syntactically correct but semantically wrong.

Lexicon

The definition of a lexicon is a dictionary or the vocabulary of a language, a people or a subject. An example of lexicon is YourDictionary.com.

An example of lexicon is **a set of medical terms**. The vocabulary of a language.



Noun → **stench** [0.05] | **breeze** [0.10] | **wumpus** [0.15] | **pits** [0.05] | . . .

Verb → **is** [0.10] | **feel** [0.10] | **smells** [0.10] | **stinks** [0.05] | . . .

Adjective → **right** [0.10] | **dead** [0.05] | **smelly** [0.02] | **breezy** [0.02] . . .

Adverb → **here** [0.05] | **ahead** [0.05] | **nearby** [0.02] | . . .

Pronoun → **me** [0.10] | **you** [0.03] | **I** [0.10] | **it** [0.10] | . . .

RelPro → **that** [0.40] | **which** [0.15] | **who** [0.20] | **whom** [0.02] | . . .

Name → **Ali** [0.01] | **Bo** [0.01] | **Boston** [0.01] | . . .

Article → **the** [0.40] | **a** [0.30] | **an** [0.10] | **every** [0.05] | . . .

Prep → **to** [0.20] | **in** [0.10] | **on** [0.05] | **near** [0.10] | . . .

Conj → **and** [0.50] | **or** [0.10] | **but** [0.20] | **yet** [0.02] | . . .

Digit → **0** [0.20] | **1** [0.20] | **2** [0.20] | **3** [0.20] | **4** [0.20] | . . .

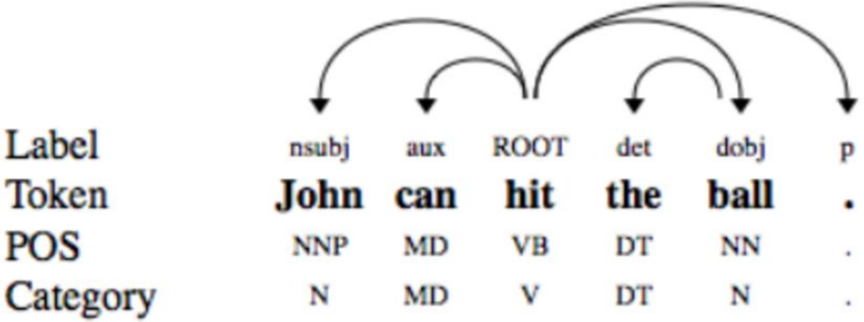
The lexicon for E_0. *RelPro* is short for relative pronoun, *Prep* for preposition, and *Conj* for conjunction. The sum of the probabilities for each category is 1.

Parsing

Parsing is the process of analyzing a string of words to uncover its phrase structure, according to the rules of a grammar. We can think of it as a search for a valid parse tree whose leaves are the words of the string.

Dependency Parsing

The term Dependency Parsing (DP) refers to **the process of examining the dependencies between the phrases of a sentence in order to determine its grammatical structure**.



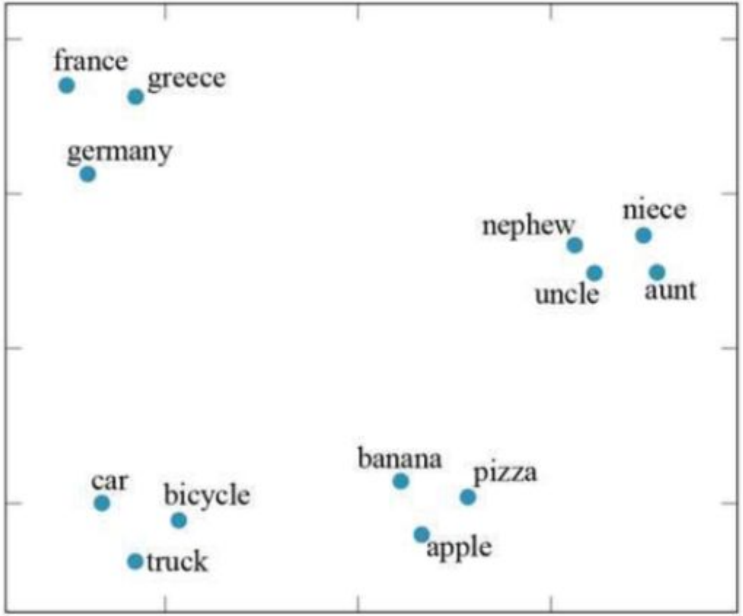
Label	nsubj	aux	ROOT	det	dobj	p
Token	John	can	hit	the	ball	.
POS	NNP	MD	VB	DT	NN	.
Category	N	MD	V	DT	N	.

Natural Language Task

1. Speech recognition
2. Text-to-speech
3. Machine translation
4. Information extraction
5. Question Answering

Word Embeddings

In natural language processing, word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.



Recurrent Neural Network for NLP

To be continued in next class….

CONTD….

Previous Class

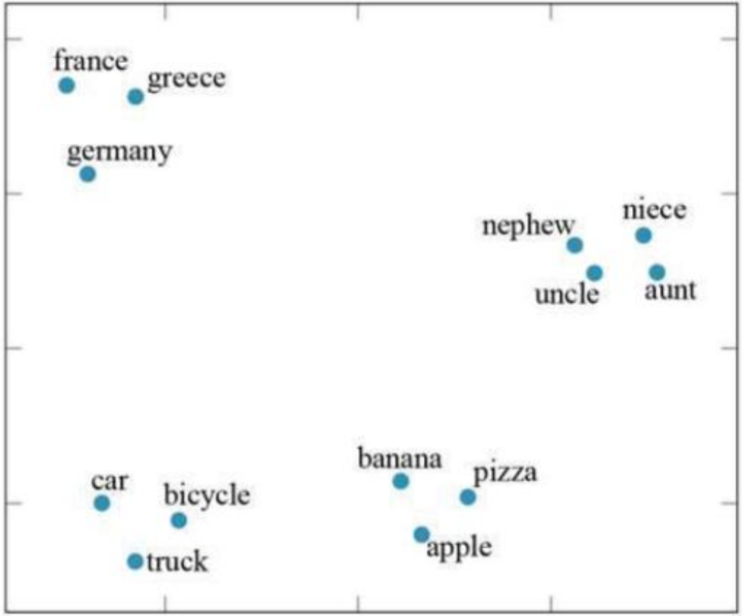
- **Natural Language Processing**: Language Models, Grammar, Parsing, Augmented Grammars, Complications of Real Natural Language
- Natural Language Tasks
- NLP Demo using Python

Contents

- Deep Learning for Natural Language Processing: Pg: 1573
  - Word Embeddings,
  - Recurrent Neural Networks for NLP,
  - Sequence-to-Sequence Models,
  - The Transformer Architecture ,
  - Pretraining and Transfer Learning,
  - State of the art

Word Embeddings

One of the most significant findings to emerge from the application of deep learning to language tasks is that a great deal deal of mileage comes from re-representing individual words as vectors in a high-dimensional space—so-called **word embeddings** (see Section 24.1). The vectors are usually extracted from the weights of the first hidden layer of a network trained on large quantities of text, and they capture the statistics of the lexical contexts in which words are used. Because words with similar meanings are used in similar contexts, they end up close to each other in the vector space. This allows the network to generalize effectively across categories of words, without the need for humans to predefine those categories. For example, a sentence beginning “John bought a watermelon and two pounds of ...” is likely to continue with “apples” or “bananas” but not with “thorium” or “geography.” Such a prediction is much easier to make if “apples” and “bananas” have similar representations in the internal layer.



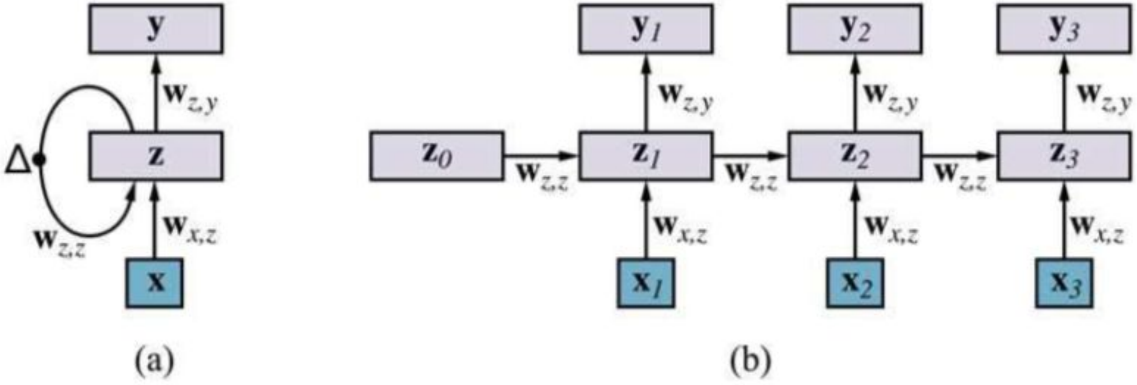
Recurrent Neural Networks for NLP

For example, in the sentence “*Eduardo told me that Miguel was very sick so I took **him** to the hospital*,” knowing that **him** refers to *Miguel* and not *Eduardo* requires context that spans from the first to the last word of the 14-word sentence.

Recurrent Neural Networks for NLP

In an RNN language model each input word is encoded as a word embedding vector, . There is a hidden layer which gets passed as input from one time step to the next. We are interested in doing multiclass classification: the classes are the words of the vocabulary. Thus the output will be a softmax probability distribution over the possible values of the next word in the sentence.

The RNN architecture solves the problem of too many parameters. The number of parameters in the weight matrixes , , and stays constant, regardless of the number of words



(a) Schematic diagram of an RNN where the hidden layer **z** has recurrent connections; the Δ symbol indicates a delay. Each input **x** is the word embedding vector of the next word in the sentence. Each output **y** is the output for that time step. (b) The same network unrolled over three timesteps to create a feedforward network. Note that the weights are shared across all timesteps.

Fig: Architecture of RNN

Recurrent Neural Networks for NLP

To train RNN model, the inputs, , are the words in a training corpus of text, and the observed outputs are the same words offset by 1. That is, for the training text “hello world,” the first input is the word embedding for “hello” and the first output is the word embedding for “world.” We are training the model to predict the next word, and expecting that in order to do so it will use the hidden layer to represent useful information. We compute the difference between the observed output and the actual output computed by the network, and back-propagate through time, taking care to keep the weights the same for all time steps.I,

Once the model has been trained, we can use it to generate random text. We give the model an initial input word , from which it will produce an output which is a softmax probability distribution over words. We sample a single word from the distribution, record the word as the output for time , and feed it back in as the next input word . We repeat for as long as desired. In sampling from we have a choice: we could always take the most likely word; we could sample according to the probability of each word; or we could oversample the less-likely words, in order to inject more variety into the generated output. The sampling weight is a hyperparameter of the model.

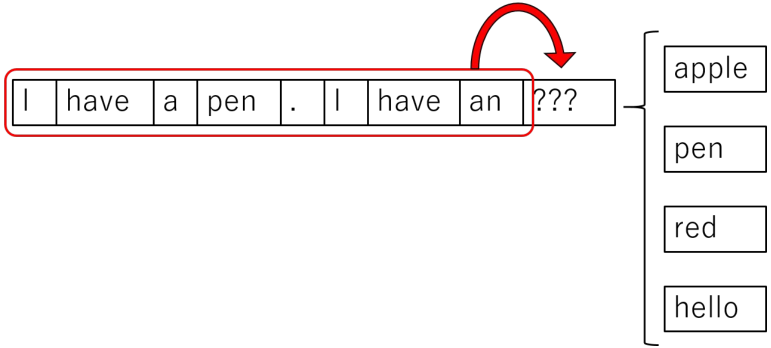
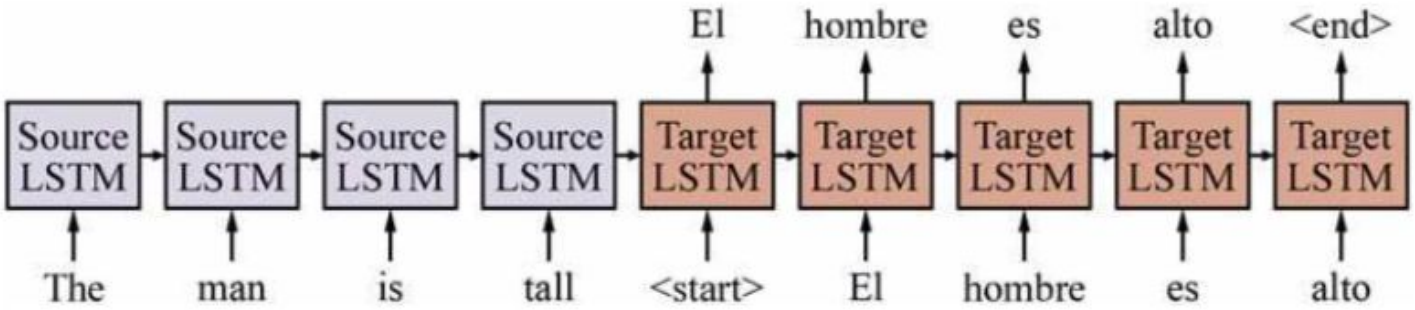


Fig: Example of RNN

Sequence to Sequence Model

Sequence to Sequence (often abbreviated to seq2seq) models is **a special class of Recurrent Neural Network architectures** that we typically use (but not restricted) to solve complex Language problems like Machine Translation, Question Answering, creating Chatbots, Text Summarization, etc.

Sequence to Sequence Model



Basic sequence-to-sequence model. Each block represents one LSTM timestep. (For simplicity, the embedding and output layers are not shown.) On successive steps we feed the network the words of the source sentence "The man is tall," followed by the `<start>` tag to indicate that the network should start producing the target sentence. The final hidden state at the end of the source sentence is used as the hidden state for the start of the target sentence. After that, each target sentence word at time t is used as input at time $t + 1$, until the network produces the `<end>` tag to indicate that sentence generation is finished.

Fig: Example of Seq-Seq model

The transformer Architecture

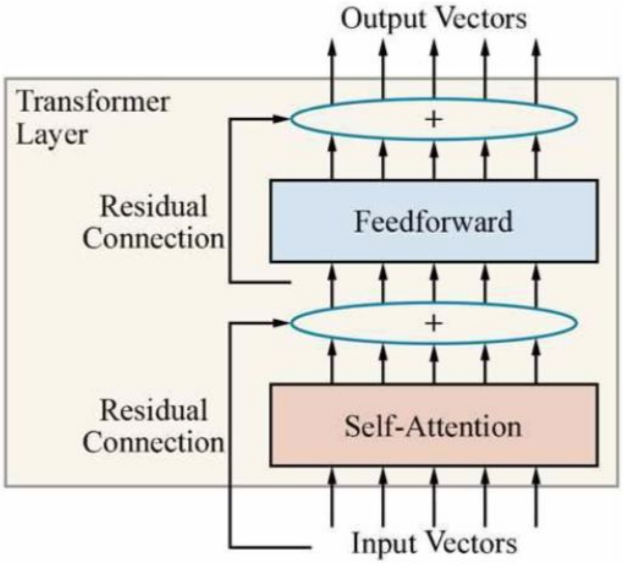
A transformer is a deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data. It is used primarily in the fields of natural language processing (NLP) and computer vision (CV).

Like recurrent neural networks (RNNs), transformers are designed to process sequential input data, such as natural language, with applications towards tasks such as translation and text summarization. However, unlike RNNs, transformers process the entire input all at once. The attention mechanism provides context for any position in the input sequence. For example, if the input data is a natural language sentence, the transformer does not have to process one word at a time. This allows for more parallelization than RNNs and therefore reduces training times.

Transformers were introduced in 2017 by a team at Google Brain[1] and are increasingly the model of choice for NLP problems,[3] replacing RNN models such as long short-term memory (LSTM).

The transformer Architecture

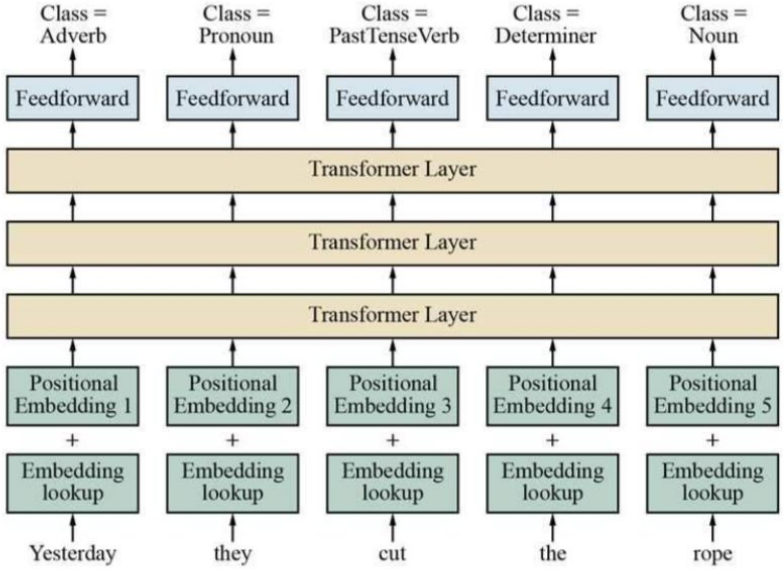
The transformer architecture does not explicitly capture the order of words in the sequence, since context is modeled only through self-attention, which is agnostic to word order. To capture the ordering of the words, the transformer uses a technique called positional embedding. If our input sequence has a maximum length of , then we learn new embedding vectors—one for each word position. The input to the first transformer layer is the sum of the word embedding at position plus the positional embedding corresponding to position .



A single-layer transformer consists of self-attention, a feedforward network, and residual connections.

The transformer Architecture

In this section, we have actually only told half the transformer story: the model we described here is called the transformer encoder. It is useful for text classification tasks. The full transformer architecture was originally designed as a sequence-to-sequence model for machine translation. Therefore, in addition to the encoder, it also includes a transformer decoder. The encoder and decoder are nearly identical, except that the decoder uses a version of self-attention where each word can only attend to the words before it, since text is generated left-to-right. The decoder also has a second attention module in each transformer layer that attends to the output of the transformer encoder.



Using the transformer architecture for POS tagging.

PreTraining and Transfer Learning

Pretraining: a form of transfer learning in which we use a large amount of shared general- domain language data to train an initial version of an NLP model. From there, we can use a smaller amount of domain-specific data (perhaps including some labeled data) to refine the model. The refined model can learn the vocabulary, idioms, syntactic structures, and other linguistic phenomena that are specific to the new domain.

Pretraining word embedding: GloVe, Google Word2Vec.

Masked language model (MLM). MLMs are trained by masking (hiding) individual words in the input and asking the model to predict the masked words. For this task, one can use a deep bidirectional RNN or transformer on top of the masked sentence. For example, given the input sentence “The river__rose five feet” we can mask the middle word to get “The river five feet” and ask the model to fill in the blank.

State of the Art

Deep learning and transfer learning have markedly advanced the state of the art for NLP. It started with simple word embeddings from systems such WORD2VEC in 2013 and GloVe in 2014. Researchers can download such a model or train their own relatively quickly without access to supercomputers.

State of the Art Models

- BERT
- XLNET
- State of the Art models became feasible only after hardware advances (GPUs and TPUs).
- The transformer model allowed for efficient training of much larger and deeper neural networks than was previously possible
- A ROBERTA model with some fine-tuning achieves state-of-the-art results in question answering and reading comprehension tests
- GPT-2, a transformer-like language model with 1.5 billion parameters trained on 40GB of Internet text, achieves good results on such diverse tasks as translation between French and English
- T5 (the Text-to-Text Transfer Transformer) is designed to produce textual responses to various kinds of textual input.

CONTD….

Prev Contents

- Deep Learning for Natural Language Processing:
    - Word Embeddings,
    - Recurrent Neural Networks for NLP,
    - Sequence-to-Sequence Models,
    - The Transformer Architecture ,
    - Pretraining and Transfer Learning,
    - State of the art

Contents

- **Computer Vision:**
  - Image Formation,
  - Simple Image Features,
  - Classifying Images,
  - Detecting Objects,
  - The 3D World,
  - Using Computer Vision
- **Robotics:**
  - Robot Hardware,
  - Robotic Perception,
  - Planning and Control,
  - Planning Uncertain Movements,
  - Reinforcement Learning in Robotics,
  - Humans and Robots,
  - Application Domains

Computer Vision

Computer vision is **a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs** — and take actions or make recommendations based on that information.

Image Formation

Pg: 1619

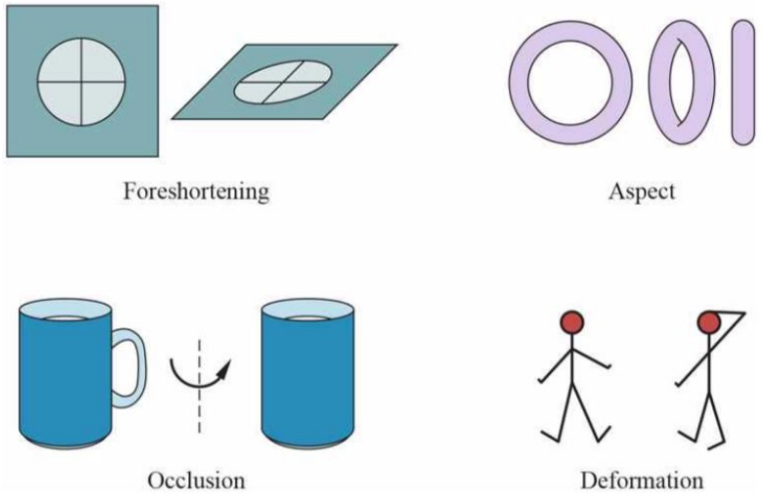
Simple Image Feature

- Edges
- Boundary
- Texture
- Segementation

Noise here means changes to the value of a pixel that don’t have to do with an edge.

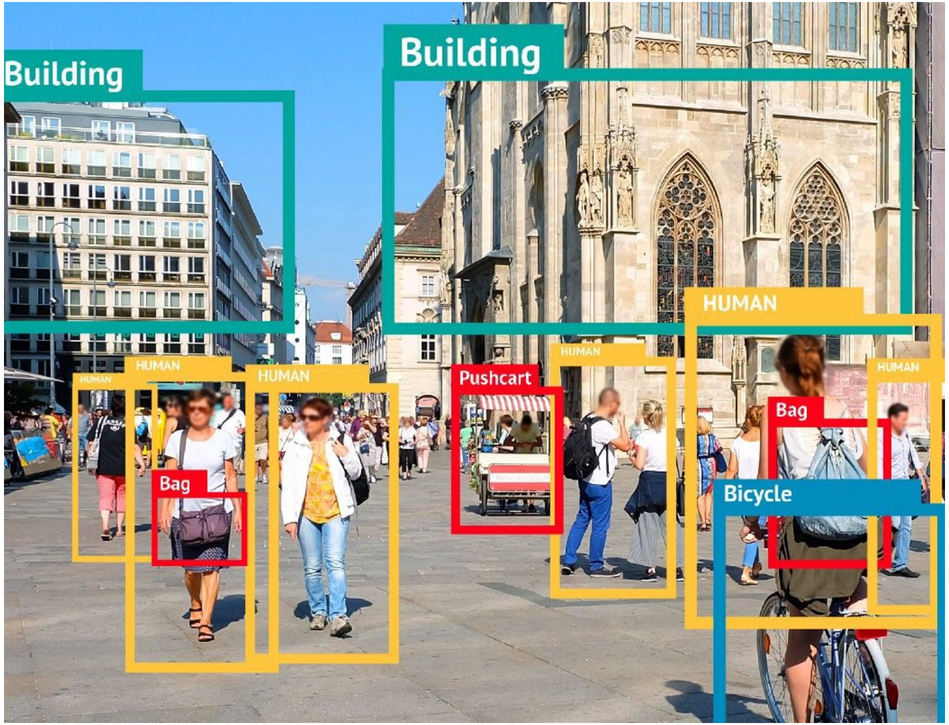
Classifying images

- **LIGHTING**, which changes the brightness and color of the image.
- **FORESHORTENING**, which causes a pattern viewed at a glancing angle to be distorted.
- **ASPECT**, which causes objects to look different when seen from different directions. A doughnut seen from the side looks like a flattened oval, but from above it is an annulus.
- **OCCLUSION**, where some parts of the object are hidden. Objects can occlude one another, or parts of an object can occlude other parts, an effect known as self-occlusion.
- **DEFORMATION**, where the object changes its shape. For example, the tennis player moves her arms and legs.



Detecting Objects

Object detection is used to locate and identify objects in images. You can use Custom Vision to train a model to detect specific classes of object in images.



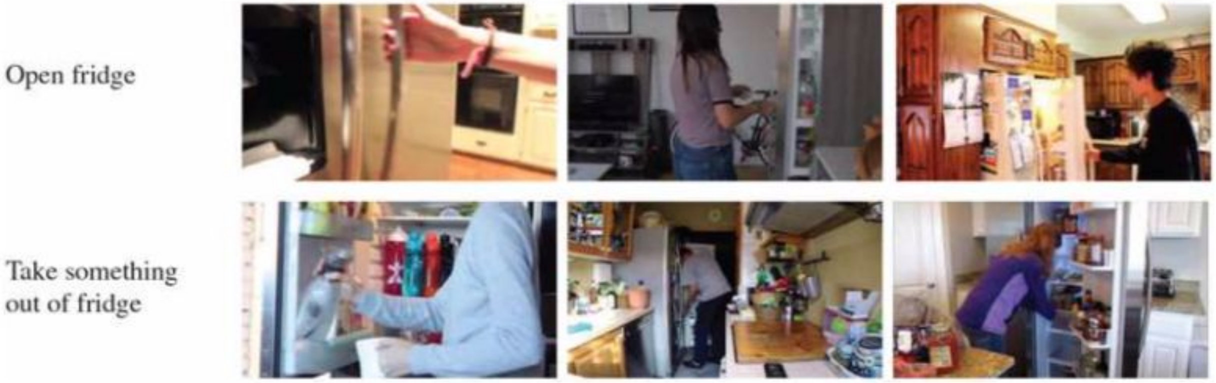
The 3D World

PG: 1654

Using Computer vision

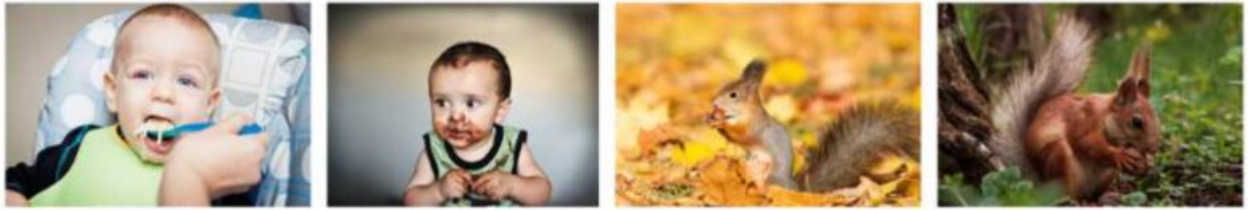
Classifying what people are doing is harder. Video that shows rather structured behaviors, like ballet, gymnastics, or tai chi, where there are quite specific vocabularies that refer to very precisely delineated activities on simple backgrounds, is quite easy to deal with. Good results can be obtained with a lot of labeled data and an appropriate convolutional neural network. However, it can be difficult to prove that the methods actually work, because they rely so strongly on context. For example, a classifier that labels “swimming” sequences very well might just be a swimming pool detector, which wouldn’t work for (say) swimmers in rivers.

Using Computer vision



The same action can look very different; and different actions can look similar. These examples show actions taken from a data set of natural behaviors; the labels are chosen by the curators of the data set, rather than predicted by an algorithm. **Top:** examples of the label "opening fridge," some shown in closeup and some from afar. **Bottom:** examples of the label "take something out of fridge." Notice how in both rows the subject's hand is close to the fridge door—telling the difference between the cases requires quite subtle judgment about where the hand is and where the door is. Figure courtesy of David Fouhey, taken from a data set described in Fouhey *et al.* (2018).

Using Computer vision



A baby eating a piece of food in his mouth

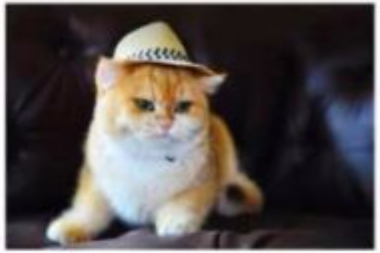
A young boy eating a piece of cake

A small bird is perched on a branch

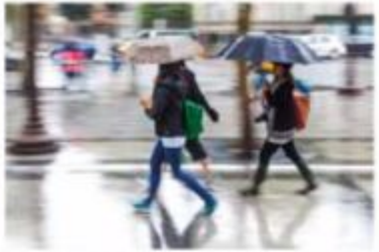
A small brown bear is sitting in the grass

Automated image captioning systems produce some good results and some failures. The two captions at left describe the respective images well, although "eating ... in his mouth" is a disfluency that is fairly typical of the recurrent neural network language models used by early captioning systems. For the two captions on the right, the captioning system seems not to know about squirrels, and so guesses the animal from context; it also fails to recognize that the two squirrels are eating. Image credits: geraine/Shutterstock; ESB Professional/Shutterstock; BushAlex/Shutterstock; Maria.Tem/Shutterstock. The images shown are similar but not identical to the original images from which the captions were generated. For the original images see Aneja *et al.* (2018).

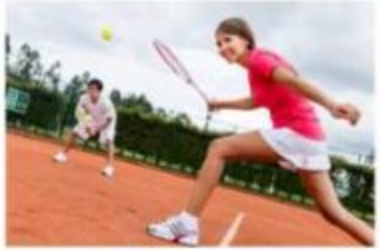
Using Computer vision: Visual Question Answering



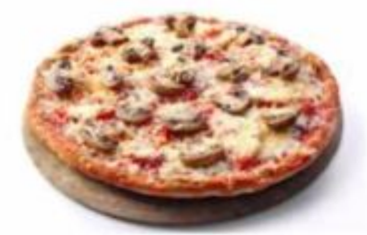
Q. What is the cat wearing?
A. Hat



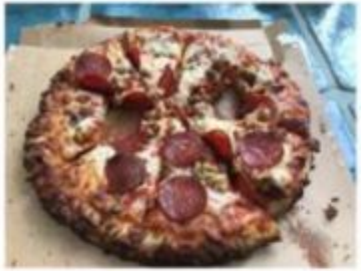
Q. What is the weather like?
A. Rainy



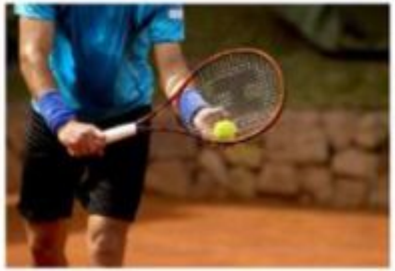
Q. What surface is this?
A. Clay



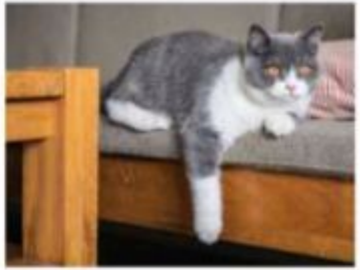
Q. What toppings are on the pizza?
A. Mushrooms



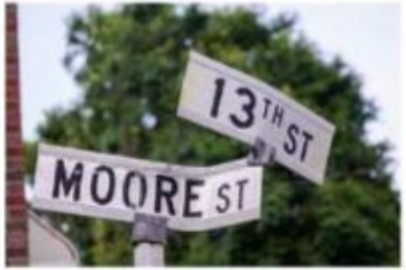
Q. How many holes are in the pizza?
A. 8



Q. What letter is on the racket?
A. w



Q. What color is the right front leg?
A. Brown

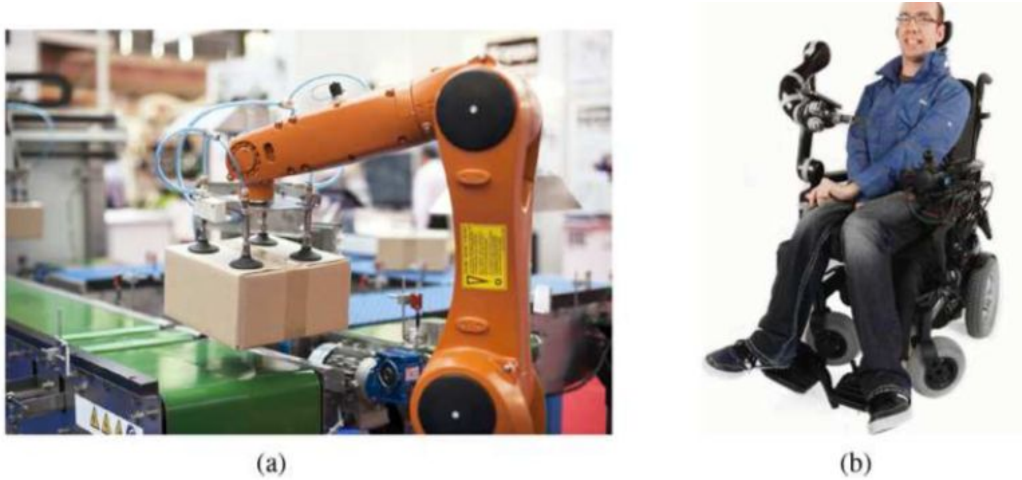


Q. Why is the sign bent?
A. It's not

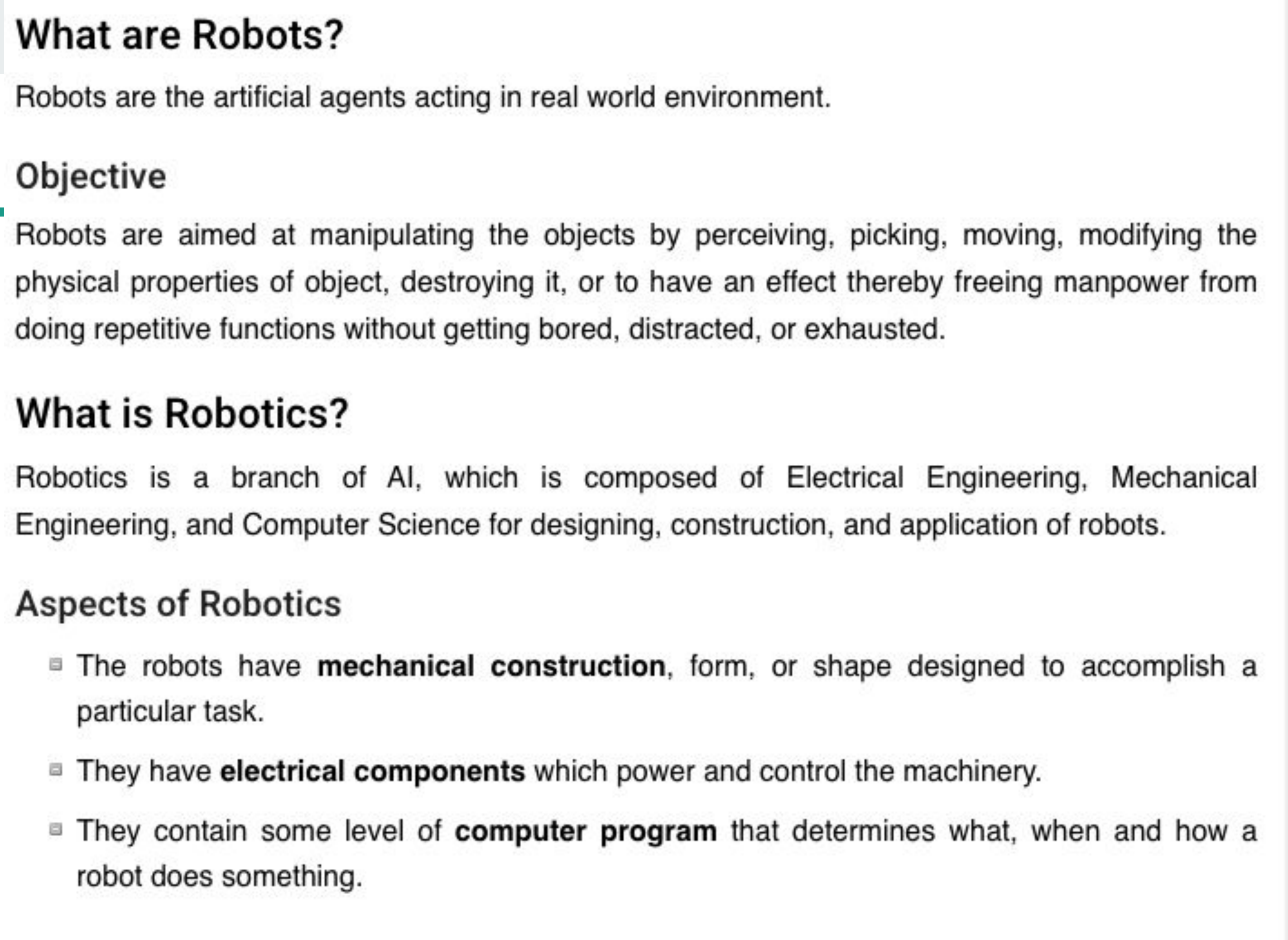
Robots

Robots are physical agents that perform tasks by manipulating the physical world. To do so, they are equipped with effectors such as legs, wheels, joints, and grippers.

Effectors are designed to assert physical forces on the environment. When they do this, a few things may happen: **the robot's state might change** (e.g., a car spins its wheels and makes progress on the road as a result), **the state of the environment might change** (e.g., a robot arm uses its gripper to push a mug across the counter), and even **the state of the people around the robot might change** (e.g., an exoskeleton moves and that changes the configuration of a person's leg; or a mobile robot makes progress toward the elevator doors, and a person notices and is nice enough to move out of the way, or even push the button for the robot).



(a) An industrial robotic arm with a custom end-effector. Image credit: Macor/123RF. (b) A Kinova® JACO® Assistive Robot arm mounted on a wheelchair. Kinova and JACO are trademarks of Kinova, Inc.



What are Robots?

Robots are the artificial agents acting in real world environment.

Objective

Robots are aimed at manipulating the objects by perceiving, picking, moving, modifying the physical properties of object, destroying it, or to have an effect thereby freeing manpower from doing repetitive functions without getting bored, distracted, or exhausted.

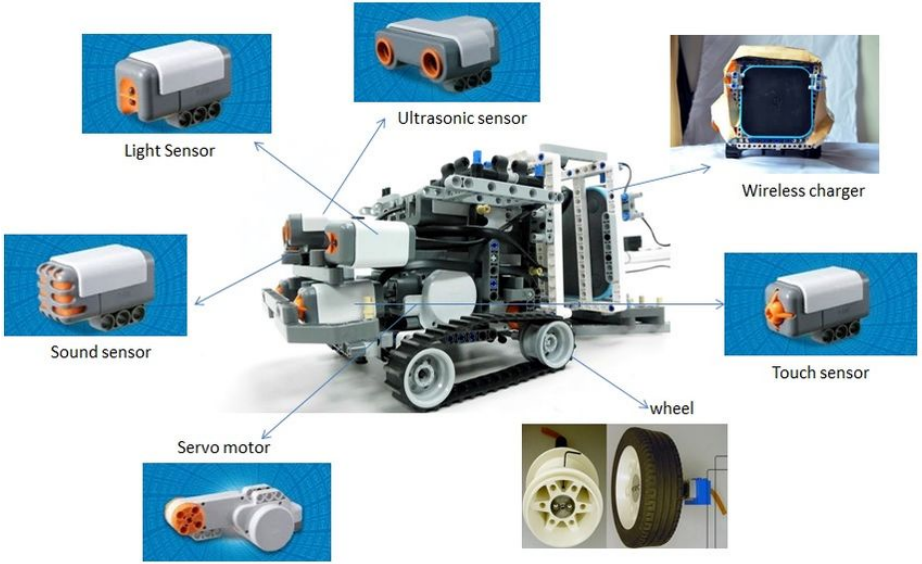
What is Robotics?

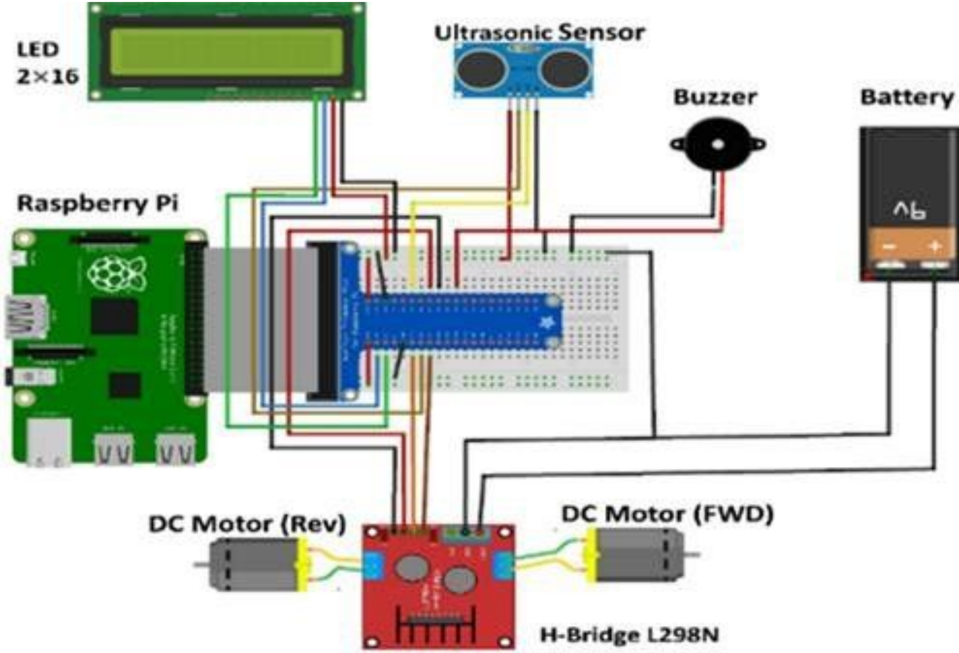
Robotics is a branch of AI, which is composed of Electrical Engineering, Mechanical Engineering, and Computer Science for designing, construction, and application of robots.

Aspects of Robotics

- The robots have **mechanical construction**, form, or shape designed to accomplish a particular task.
- They have **electrical components** which power and control the machinery.
- They contain some level of **computer program** that determines what, when and how a robot does something.

Robot Hardware





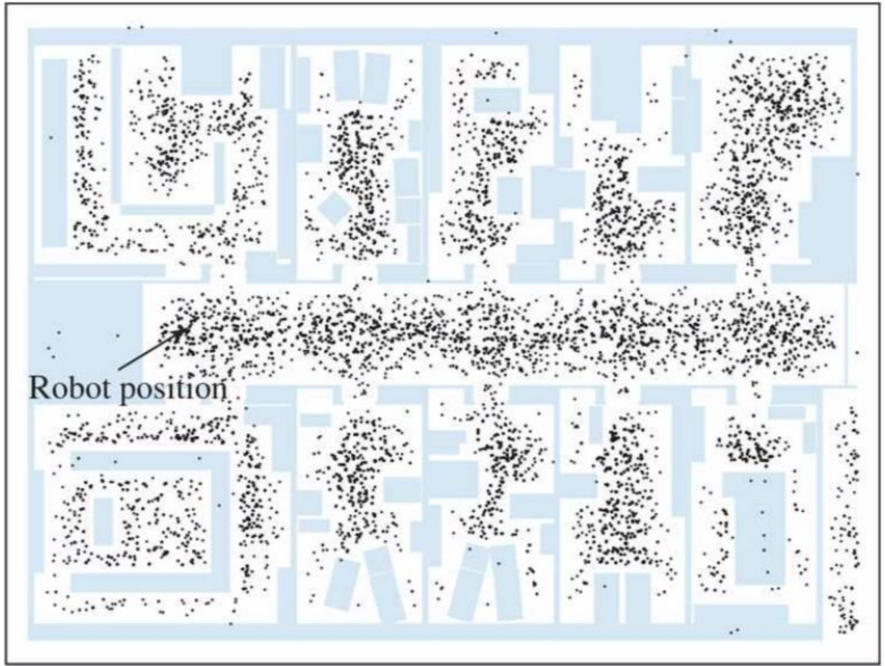
Robotic Perceptron

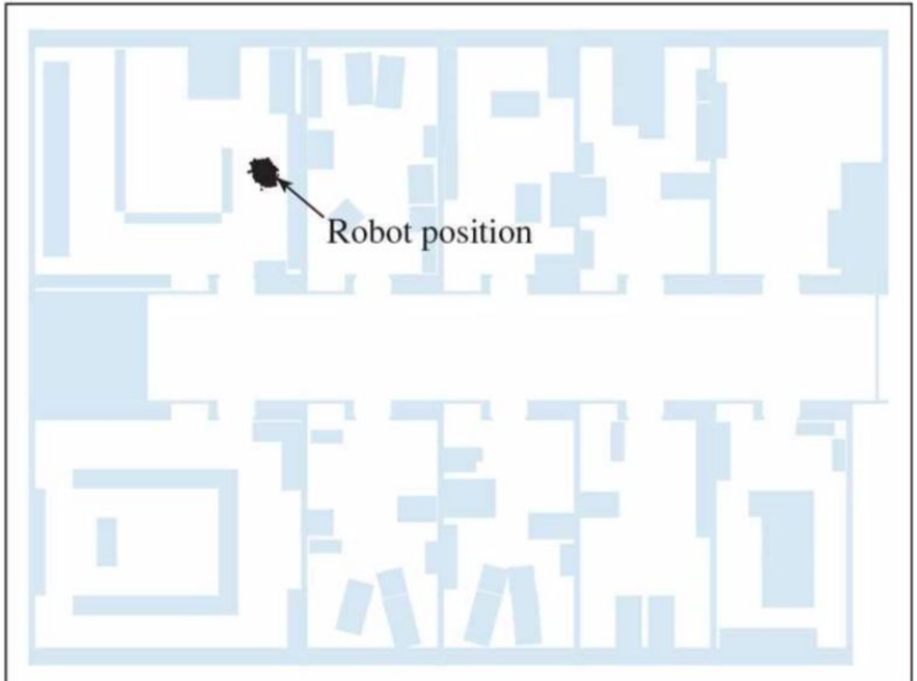
Perception is the process by which robots map sensor measurements into internal representations of the environment. Much of it uses the computer vision techniques from the previous chapter. But perception for robotics must deal with additional sensors like lidar and tactile sensors.

A good internal representations for robots have three properties:

- They contain enough information for the robot to make good decisions.
- They are structured so that they can be updated efficiently.
- They are natural in the sense that internal variables correspond to natural state variables in the physical world.

Example





Planning and Control

The robot’s deliberations ultimately come down to deciding how to move, from the abstract task level all the way down to the currents that are sent to its motors. In this section, we simplify by assuming that perception (and, where needed, prediction) are given, so the world is observable. We further assume deterministic transitions (dynamics) of the world.

We start by separating motion from control. We define a path as a sequence of points in geometric space that a robot (or a robot part, such as an arm) will follow. Here we mean a sequence of points in space rather than a sequence of discrete actions. The task of finding a good path is called **motion planning.**

Once we have a path, the task of executing a sequence of actions to follow the path is called trajectory tracking control. A **trajectory** is a path that has a time associated with each point on the path. A path just says “go from A to B to C, etc.” and a trajectory says “start at A, take 1 second to get to B, and another 1.5 seconds to get to C, etc.”

Planning uncertain movement

In robotics, uncertainty arises from partial observability of the environment and from the stochastic (or unmodeled) effects of the robot's actions. Errors can also arise from the use of approximation algorithms such as particle filtering, which does not give the robot an exact belief state even if the environment is modeled perfectly.

Applications

The robotics has been instrumental in the various domains such as −

- Industries − Robots are used for handling material, cutting, welding, color coating, drilling, polishing, etc.
- Military − Autonomous robots can reach inaccessible and hazardous zones during war. A robot named *Daksh*, developed by Defense Research and Development Organization (DRDO), is in function to destroy life-threatening objects safely.
- Medicine − The robots are capable of carrying out hundreds of clinical tests simultaneously, rehabilitating permanently disabled people, and performing complex surgeries such as brain tumors.
- Exploration − The robot rock climbers used for space exploration, underwater drones used for ocean exploration are to name a few.
- Entertainment − Disney's engineers have created hundreds of robots for movie making.