UNIT V APPLICATIONS 9

AI applications –Language Models –Information Retrieval-Information Extraction –Natural Language Processing -Machine Translation –Speech Recognition –Robot –Hardware –Perception –Planning –Moving

1. AI APPLICATIONS

AI programs are developed to perform specific tasks, that is being utilized for a wide range of activities including *medical diagnosis, electronic trading platforms, robot control, and remote sensing*. AI has been used to develop and advance *numerous fields and industries, including finance, healthcare, education, transportation, and more*.

- Information Retrieval
- Natural Language Processing
- Machine Translation
- Speech Recognition
- Robot Marketing
- Banking
- Finance
- Agriculture
- HealthCare
- Gaming
- Space Exploration
- Autonomous Vehicles
- Chatbots
- Artificial Creativity

Marketing

- we search for an item on any e-commerce store, we get all possible results related to the item. It's like these search engines read our minds! In a matter of seconds, we get a list of all relevant items.

With the growing advancement in AI, in the near future, it may be possible for consumers on the web to buy products by snapping a photo of it

Banking

- A lot of banks have already adopted AI-based systems to provide customer support, detect anomalies and credit card frauds. An example of this is HDFC Bank.
- HDFC Bank has developed an AI-based chatbot called *EVA* (Electronic Virtual Assistant), built by Bengaluru-based Senseforth AI Research.
- Since its launch, Eva has addressed over 3 million customer queries, interacted with over half a million unique users, and held over a million conversations.
- Eva can collect knowledge from thousands of sources and provide simple answers in less than 0.4 seconds.
- AI solutions can be used to enhance security across a number of business sectors, including retail and finance.
- By tracing card usage and endpoint access, security specialists are more effectively preventing fraud. Organizations rely on AI to trace those steps by analyzing the behaviors of transactions.

Companies such as MasterCard and RBS WorldPay have relied on AI and *Deep Learning* to detect fraudulent transaction patterns and prevent card fraud for years now. This has saved millions of dollars.

Finance

- Ventures have been relying on computers and data scientists to determine future patterns in the market. Trading mainly depends on the ability to predict the future accurately.
- Machines are great at this because they can crunch a huge amount of data in a short span. Machines can also learn to observe patterns in past data and predict how these patterns might repeat in the future.
- Financial organizations are turning to AI to improve their stock trading performance and boost profit.

Agriculture

- AI can help farmers get more from the land while using resources more sustainably.
- Issues such as climate change, population growth, and food security concerns have pushed the industry into seeking more innovative approaches to improve crop yield.
- Organizations are using automation and robotics to help farmers find more efficient ways to protect their crops from weeds.
- Blue River Technology has developed a robot called See & Spray which uses computer vision technologies like *object detection* to monitor and precisely spray weedicide on cotton plants. Precision spraying can help prevent herbicide resistance
- The image recognition app identifies possible defects through images captured by the user's smartphone camera. Users are then provided with soil restoration techniques, tips, and other possible solutions. The company claims that its software can achieve pattern detection with an estimated accuracy of up to 95%.

Health Care

- When it comes to saving our lives, a lot of organizations and medical care centers are relying on AI.
- clinical decision support system for stroke prevention that can give the physician a warning when there's a patient at risk of having a heart stroke.
- Preventive care
- Personalized medicine

Gaming

- Artificial Intelligence has become an integral part of the gaming industry. In fact, one of the biggest accomplishments of AI is in the gaming industry.
- DeepMind's AI-based AlphaGo software, which is known for defeating Lee Sedol, the world champion in the game of GO, is considered to be one of the most significant accomplishment in the field of AI
- The actions taken by the opponent AI are unpredictable because the game is designed in such a way that the opponents are trained throughout the game and never repeat the same mistakes.
- They get better as the game gets harder. This makes the game very challenging and prompts the players to constantly switch strategies and never sit in the same position.

Space Exploration

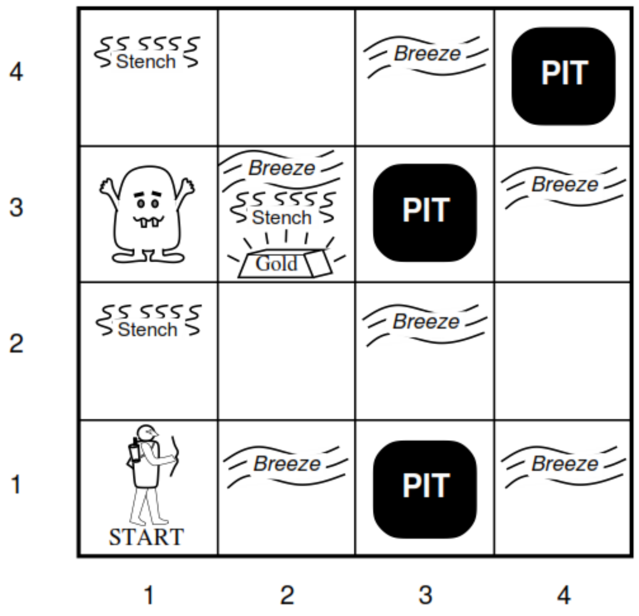
- Space expeditions and discoveries always require analyzing vast amounts of data.
- Artificial Intelligence and Machine learning is the best way to handle and process data on this scale.
- After rigorous research, astronomers used Artificial Intelligence to sift through years of data obtained by the Kepler telescope in order to identify a distant eight-planet solar system.

- Artificial Intelligence is also being used for NASA's next rover mission to Mars, the Mars 2020 Rover. The AEGIS, which is an AI-based Mars rover is already on the red planet.
- The rover is responsible for autonomous targeting of cameras in order to perform investigations on Mars.

Autonomous Vehicles

- For the longest time, self-driving cars have been a buzzword in the AI industry. The development of autonomous vehicles will definitely revolutionaries the transport system.
- Companies like Waymo conducted several test drives in Phoenix before deploying their first AI-based public ride-hailing service.
- The AI system collects data from the vehicles radar, cameras, GPS, and cloud services to produce control signals that operate the vehicle.
- Advanced Deep Learning algorithms can accurately predict what objects in the vehicle's vicinity are likely to do. This makes Waymo cars more effective and safer.
- Another famous example of an autonomous vehicle is Tesla's self-driving car. Artificial Intelligence implements computer vision, image detection and deep learning to build cars that can automatically detect objects and drive around without human intervention

2. WUMPUS WORLD



- The Wumpus world is a simple world example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. It was inspired by a video game Hunt the Wumpus by Gregory Yob in 1973.
- The Wumpus world is a cave which has **4/4 rooms connected with passageways**. So there are total 16 rooms which are connected with each other.
- We have a knowledge-based agent who will go forward in this world. The cave has a room with a *beast which is called Wumpus, who eats anyone who enters the room*.
- The Wumpus can be shot by the agent, but the agent has a single arrow.
- In the Wumpus world, *there are some Pits rooms which are bottomless*, and if agent falls in Pits, then he will be stuck there forever.
- The exciting thing with this cave is that in one room there is a possibility of finding a heap of gold. So the agent goal is *to find the gold and climb out the cave* without fallen into Pits or eaten by Wumpus.
- The agent will get a reward if he comes out with gold, and he will get a penalty if eaten by Wumpus or falls in the pit.

There are also some components which can help the agent to navigate the cave. These components are given as follows:

a. The rooms *adjacent to the Wumpus room are smelly*, so that it would have some stench.
b. The *room adjacent to PITs has a breeze*, so if the agent reaches near to PIT, then he will perceive the breeze.
c. There will be *glitter in the room if and only if the room has gold*.
d. The Wumpus can be killed by the agent if the agent is facing to it, and Wumpus will emit a horrible scream which can be heard anywhere in the cave.

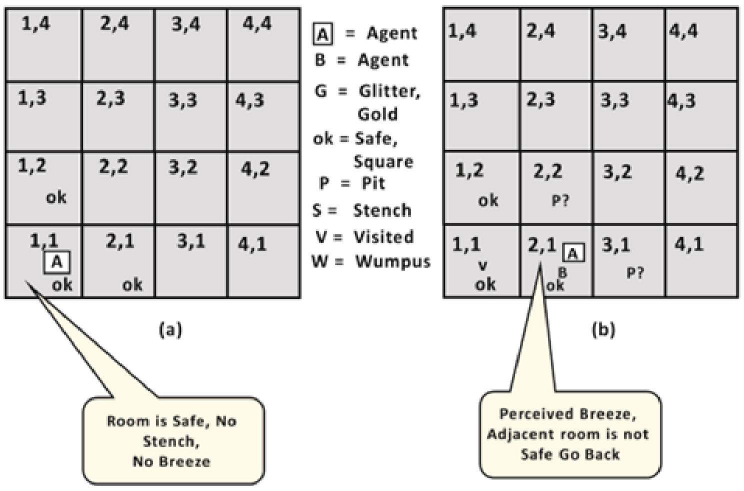
The Wumpus world Properties:

- **Partially observable:** The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room.
- **Deterministic:** It is deterministic, as the result and outcome of the world are already known.
- **Sequential:** The order is important, so it is sequential.
- **Static:** It is static as Wumpus and Pits are not moving.
- **Discrete:** The environment is discrete.
- **One agent:** The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.

Exploring the Wumpus world:

- Now we will explore the Wumpus world and will determine how the agent will find its goal by applying logical reasoning.

  Agent's First step:
- Initially, the agent is in the first room or on the square [1,1], and we already know that this room is safe for the agent, so to represent on the below diagram (a) that room is safe we will add symbol OK. Symbol A is used to represent agent, symbol B for the breeze, G for Glitter or gold, V for the visited room, P for pits, W for Wumpus.
- At Room [1,1] agent does not feel any breeze or any Stench which means the adjacent squares are also OK.

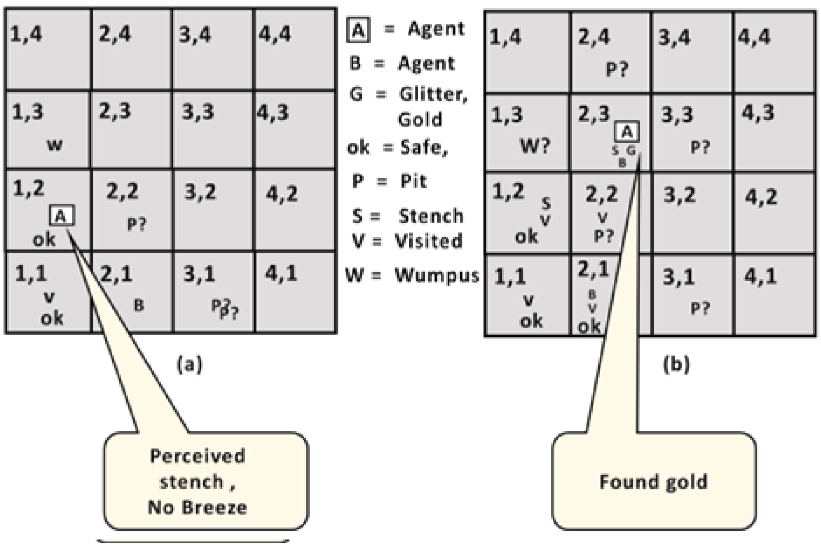


Agent's second Step:
- Now agent needs to move forward, so it will either move to [1, 2], or [2,1]. Let's suppose agent moves to the room [2, 1], at this room agent perceives *some breeze which means Pit is around this room*. The pit can be in [3, 1], or [2,2], so we will add symbol P? to say that, is this Pit room?

- Now agent will stop and think and will not make any harmful move. The agent will go back to the [1, 1] room. The room [1,1], and [2,1] are visited by the agent, so we will use symbol V to represent the visited squares.

Agent's third step:

- At the third step, now agent will move to the room [1,2] which is OK. In the room [1,2] agent perceives a stench which means there must be a Wumpus nearby. But Wumpus cannot be in the room [1,1] as by rules of the game, and also not in [2,2] (Agent had not detected any stench when he was at [2,1]). Therefore agent infers *that Wumpus is in the room [1,3], and in current state*, there is no breeze which means in [2,2] there is no Pit and no Wumpus. So it is safe, and we will mark it OK, and the agent moves further in [2,2].



Agent's fourth step:

- At room [2,2], here no stench and no breezes present so let's suppose agent decides to move to [2,3]. At room [2,3] agent perceives glitter, *so it should grab the gold and climb out of the cave*.

3. LANGUAGE MODELS

Humans use a limited number of conventional signs (smiling, shaking hands) to communicatein much the same way as other animals. Humans have also developed a complex, structuredsystem of signs known as **language** that enables them to communicate most of what they knowabout the world.

One of the actions that is available to an agent is to produce language.
This is called a **speechact.** "Speech" is used in the same sense as in "free speech," not "talking," so typing, skywriting,and using sign language all count as speech acts.
English has no neutral word for an agent thatproduces language, either by speaking or writing or anything else. We will use **speaker, hearer, andutterance** as generic terms referring to any mode of communication.

Imagine a group of **agents are exploring the wumpus world together**. The group gains anadvantage (collectively and individually) by being able to do the following:

- **Inform** each other about the part of the world each has explored, so that each agent hasless exploring to do. This is done by making statements: *There's a breeze here in 3 4.*
- **Query** other agents about particular aspects of the world. This is typically done by askingquestions: *Have you smelled the wumpus anywhere?*
- **Answer** questions. This is a kind of informing. *Yes, I smelled the wumpus in 2 5.*
- **Request or command** other agents to perform actions: *Please help me carry the gold.* Itcan be seen as impolite to make a direct requests, so often an **indirect speech act** (a requestin the form of a

statement or question) is used instead: / *could use some help carrying this*or*Could you help me cany this?*
- **Promise** to do things or **offer** deals: /'// *shoot the wumpus if you let me share the gold.*
- **Acknowledge** requests and offers: *OK.*
- **Share** feelings and experiences with each other: *You know, old chap, when I get in a spotlike that, I usually go back to the start and head out in another direction,* or *Man, thatwumpus sure needs some deodorant!*

Planning problem

The hard part for an agent is to decide *when* a speech act of some kind is called for,and to decide *which* speech act, out of all the possibilities, is the right one. At one level,this is just the familiar **planning** problem—an agent has a set of possible actions to choosefrom, and must somehow try to choose actions that achieve the goal of communicating someinformation to another agent.

Understanding problems

- The problem of understanding speech acts is much like other **understanding** problems,such as *image understanding or medical diagnosis*.
- Part of the speech act *understanding problem is specific to language*.
- We need to know something about the *syntax and semantics of a language* to determine why anotheragent performed a given speech act.
- The understanding problem also includes the more generalproblem of **plan recognition.**
- Part of the understanding problem can be *handled by logical reasoning*.
- We will see thatlogical implications are a good way of describing the ways that words and phrases combine toform larger phrases. Another part of the understanding problem can only be handled by uncertainreasoning techniques.
- Usually, there will be several states of the world that could all lead to thesame speech act, so the understander has to decide which one is more probable.

Fundamentals of language

Formal languages—the ones like Lisp and first-order logic that areinvented and rigidly defined
Natural languages—the ones like Chinese, Danish, andEnglish that humans use to talk to one another.

Formal language

- A formal language is defined as a set of strings, where each string is a sequence of symbolstaken from a finite set called the **terminal symbols.**
- most of them are similar in that they are based onthe idea of **phrase structure**
- Strings are composed of substrings called **phrases,** whichcome in different categories.
- Categorizing phrases helps us to describe the allowable strings of thelanguage.
- Any of the noun phrases can combine with a **verb phrase** (or *VP)* suchas "is dead" to form a phrase of category **sentence**
- **Grammatical categories** are essentially posited as part of ascientific theory of language that attempts to account for the difference between grammatical andungrammatical categories.

Example:

noun phrase"the wumpus," "the king,"
verb phrase : "is dead"
sentence :"the wumpus is dead"
Grammatical categories :"the wumpus is dead" , "wumpus the dead is"

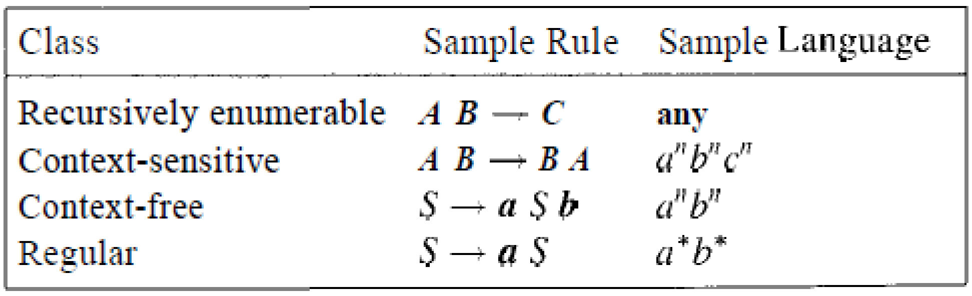
Categories such as *NP*, *VP*, and *S* are called **nonterminal symbols.** In the BNF notation,**rewrite rules** consist of a single nonterminal symbol on the left-hand side, and a sequence ofterminals or nonterminals on the right-hand side

The meaning of a rule such as S— *NP VP*

Grammatical formalisms can be classified by their **generative capacity:** the set oflanguages they can representfour classes of grammaticalformalisms that differ only in the form of the rewrite rules.

- **Recursively enumerable** grammars use unrestricted rules: both sides of therewrite rules can have any number of terminal and nonterminal symbols. Thesegrammars are equivalent to Turing machines in their expressive power.
- **Context-sensitive grammars** are restricted only in that the right-hand hand sidemust contain at least as many symbols as the left-hand side. The name contextsensitivecomes from the fact that a rule such as
  ASB — *AXB*
  says that an *S* can berewritten as an *X* in the context of a preceding *A* and a following *B.*
- **In context-free grammars** (or CFGs), the left-hand side consists of a singlenonterminal symbol. Thus, ***each rule licenses rewriting the nonterminal as the righthandside in any context***. CFGs are popular for natural language grammars, althoughit is now widely accepted that at least some natural languages are not context-free.
- **Regular** grammars are the most restricted class. Every rule has a single nonterminalon the left-hand side, and a terminal symbol optionally followed by a nonterminalon the right-hand side. *Regular grammars are equivalent in power to finite-statemachines*. They are poorly suited for programming languages because, for example,they cannot represent constructs such as balanced opening and closing parentheses.

To give you an idea of which languages can be handled by which classes, thelanguage*a"b"* (a sequence of *n* copies of *a* followed by the same number of *b)* canbe generated by a context-free grammar, but not a regular grammar. The language*a"b"c"* requires a context-sensitive grammar, whereas the language *a*b** (a sequenceof any number of *a* followed by any number of *b)* can be described by any of the fourclasses. A summary of the four classes follows.



Class	Sample Rule	Sample Language
Recursively enumerable	$A\ B \rightarrow C$	any
Context-sensitive	$A\ B \rightarrow B\ A$	$a^n b^n c^n$
Context-free	$S \rightarrow \mathbf{a}\ S\ \mathbf{b}$	$a^n b^n$
Regular	$S \rightarrow \mathbf{a}\ S$	$a^* b^*$

The component steps of communication

A typical communication episode, in which speaker *S* wants to convey proposition *P* to hearer *H* using words *W,* is composed of seven processes.
Three take place in the speaker:
Intention: *S* wants *H* to believe *P* (where *S* typically believes *P)*
Generation: *S* chooses the words *W* (because they express the meaning *P)*
Synthesis: 5 utters the words *W* (usually addressing them to *H)*

Four take place in the hearer:
Perception: *H* perceives *W* (ideally *W'* = *W,* but misperception is possible)
Analysis: *H* infers that *W* has possible meanings *P\,..., P,,* (words andphrases can have several meanings)
Disambiguation: *H* infers that *S* intended to convey P, (where ideally *Pj* = *P,*but misinterpretation is possible)
Incorporation: *H* decides to believe P, (or rejects it if it is out of line with what*H* already believes)

Intention :

- Speaker decides that there is something that is worth saying tothe hearer.
- This often involves *reasoning about the beliefs and goals of the hearer*, so that theutterance will have the desired effect.
- For our example, the speaker has the intention of havingthe hearer know that the wumpus is no longer alive.

Generation :

- The speaker uses knowledge about language to decide what to say.
- This is harder than the inverse problem of understanding (i.e., analysis and disambiguation).Generation has not been stressed as much as understanding in AI, mainly because we humansare anxious to talk to machines, but are not as excited about them talking back.
- For now, we justassume the hearer is able to choose the words "The wumpus is dead."

Synthesis :

- Most language-based AI systems synthesize typed output on a screen or paper,which is a trivial task.
- Speech synthesis has been growing in popularity, and some systems arebeginning to sound human.
- The agent in the below example is synthesizing a string of soundswritten in the phonetic alphabet "[thaxwahmpahsihzdeyd]."
- The sounds that get synthesized are *different fromthe words that the agent generates*.
- Also note that the words are run together; this is typical ofquickly spoken speech.

Perception:

When the medium is speech, the perception step is called **speech recognition;**when it is printing, it is called **optical character recognition.** Both have moved from being

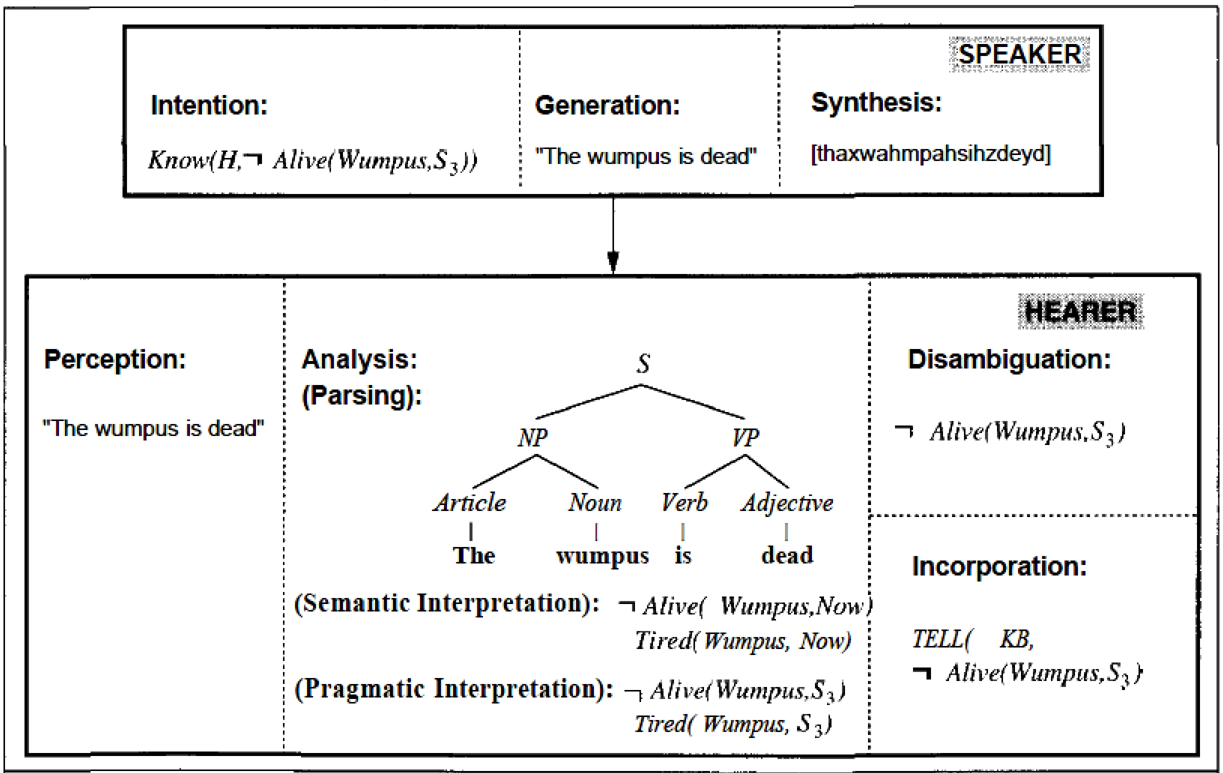


Figure 22.1 Seven processes involved in communication, using the example sentence "The wumpus is dead."

esoteric to being commonplace within the last five years. For the example, let us assume that the hearer perceives the sounds and recovers the spoken words perfectly.

Analysis:
Two main parts:
syntactic interpretation (or parsing) and semantic interpretation.

- **Semantic interpretation** includes both understanding the meanings of words and incorporating knowledge of the current situation (also called pragmatic interpretation).
- The word **parsing** is derived from the the Latin phrase *pars orationis,* or "part of speech," and refers to the process of assigning a part of speech (noun, verb, and so on) to each word in a sentence and grouping the words into phrases.
- One way of displaying the result of a syntactic analysis is with a **parse tree.**
- *A parse tree is a tree in which interior nodes represents phrases, links represent applications of grammar rules, and leaf nodes represent words*.
- We define **the yield of a node as the list of all the leaves below the node**, in left-to-right order, then we can say that the meaning of a parse tree is that each node with label *X* asserts that the yield of the node is a phrase of category *X.*
- **Semantic interpretation is the process of extracting the meaning of an utterance as an expression in some representation language.**
- Two possible semantic interpretations: that the wumpus is not alive, and that it is tired (a colloquial meaning of *dead).*

Ambiguity

- Utterances with several possible interpretations are said to be **ambiguous.**
- **lexical ambiguity,** where a word has more than one meaning.
- **Syntactic ambiguity** (also known as **structural ambiguity)** can occur with or without lexical ambiguity.

For example, the string "I smelled a wumpus in 2,2" has two parses: one where the propositional phrase modifies the noun, and one where it modifies the verb.

- **semantic ambiguity** :The syntactic ambiguity leads to a **semantic ambiguity,** because one parse means the wumpus is in 2,2 and the other means that a stench is in 2,2.
- **Referential ambiguity** occurs because natural languages consist almost entirely of words for categories, not for individual objects. There is no word for *the-apple-I-had-for-lunch-today,]\\si*categories like *apple.*
- **Pragmatic ambiguity** occurs when the speaker and hearer disagree on what the current situation is. If the speaker says "I'll meet you next Friday" thinking that they're talking about the 17th, and the hearer thinks that they are talking about the 24th, then there is miscommunication.
- **Local ambiguity** :Sometimes a phrase or sentence has **local ambiguity,** where a substring can be parsed several ways, but only one of those ways fits into the larger context of the whole string. For example, in the C programming language, the string *c means "pointer to c" when it appears in the declaration char *c; but it means "multiply by c" when it appears in the expression 2 *c.

.

Disambiguation.

- Most speakers are not intentionally ambiguous, but most utterances have several legal interpretations. Communication works because the hearer does the work of figuring out which interpretation is the one the speaker probably meant to convey.
- Disambiguation is the first process that depends heavily on uncertain reasoning. Analysis generates possible interpretations; if more than one interpretation is found, then disambiguation chooses the one that is best.

1. The world model: the probability that a fact occurs in the world.
2. The mental model: the probability that the speaker forms the intention of communicating this fact to the hearer, given that it occurs.9 (This combines models of what the speaker believes, what the speaker believes the hearer believes, and so on.)
3. The language model: the probability that a certain string of words will be chosen, given that the speaker has the intention of communicating a certain fact.

4. The acoustic model: the probability that a particular sequence of sounds will be generated, given that the speaker has chosen a given string of words

Incorporation.A totally naive agent might believe everything it hears, but a sophisticated agent treats the words *W* and the derived interpretation P, as additional pieces of evidence that get considered along with all other evidence for and against P,.

It only makes sense to use language when there are agents to communicate with who
(a) understand a common language,
(b) have a shared context on which to base the conversation, and
(c) are at least somewhat rational.

TWO MODELS OF COMMUNICATION

Study of communication centers on the way that an agent's beliefs are turned into words and back into beliefs in another agent's knowledge base (or head). There are two ways of looking at ENCODED MESSAGE the process.

ENCODED MESSAGE

The **encoded message** model says that the speaker has

- a definite proposition *P* in mind, and
- encodes the proposition into the words (or signs) *W.*

The hearer then tries to

- decode the message *W* to retrieve the original proposition *P* (cf. Morse code).

Under this model, the meaning in the speaker's head, the message that gets transmitted, and the interpretation that the hearer arrives at all ideally carry the same content. When they differ, it is because of noise in the communication channel or an error in encoding or decoding.

SITUATED LANGUAGE

Limitations of the encoded message model led to the **situated language** model, which says that the meaning of a message depends on both the words and the **situation** in which the words are uttered. In this model, just as in situation calculus, the encoding and decoding functions take an extra argument representing the current situation.

This accounts for the fact that the same words can have very different meanings in different situations.

"I am here now" represents one fact when spoken by Peter in Boston on Monday, and quite another fact when spoken by Stuart in Berkeley on Tuesday.

More subtly, "You must read this book" is a suggestion when written by a critic in the newspaper, and an assignment when spoken by an instructor to the class. "Diamond" means one thing when the subject is jewelry, and another when the subject is baseball.

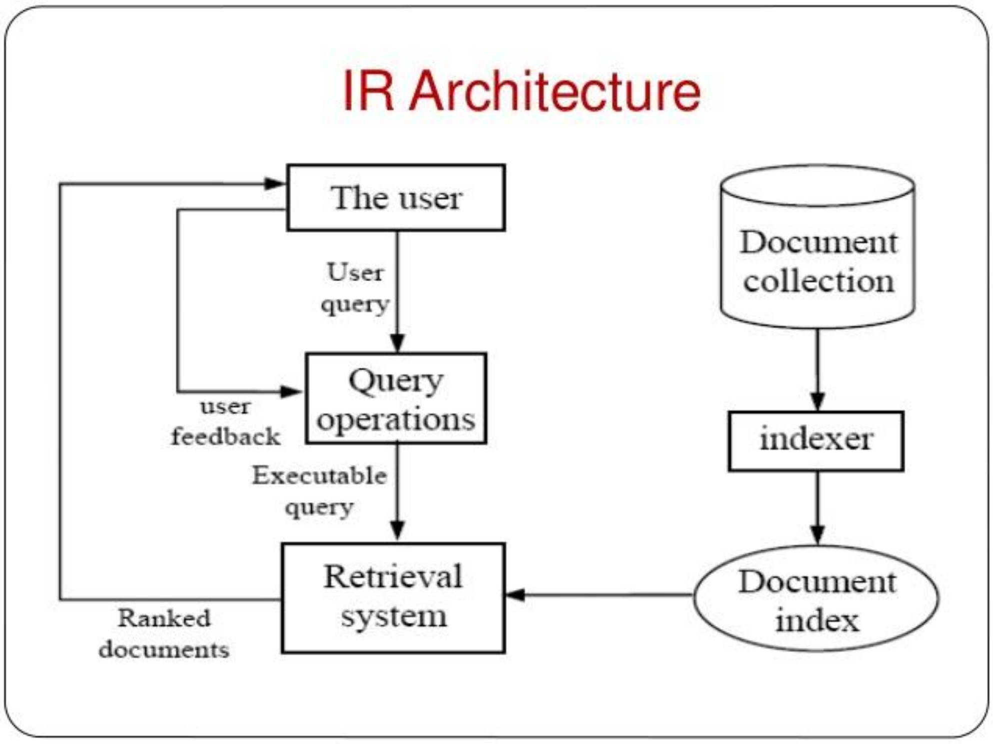
The situated language model points out a possible source of communication failure: if the speaker and hearer have different ideas of what the current situation is, then the message may not get through as intended.

For example, suppose agents *X*, *Y*, and Z are exploring the wumpus world together. *X* and *Y* secretly meet and agree that when they smell the wumpus they will both shoot it, and if they see the gold, they will grab it and run, and try to keep Z from sharing it. Now suppose*X* smells the wumpus, while *Y* in the adjacent square smells nothing, but sees the gold.

X yells "Now!," intending it to mean that now is the time to shoot the wumpus, but *Y* interprets itas meaning now is the time to grab the gold and run.

4. INFORMATION RETRIEVAL

Information retrieval is the task of finding documents that are relevant to a user's need for information. The best-known examples of information retrieval systems are search engines on the World Wide Web. A Web user can type a query such as [AI book] into a search engine and see a list of relevant pages.



An information retrieval (henceforth **IR**) system can be characterized by

1. **A corpus of documents.** Each system must decide what it wants to treat as a document: a paragraph, a page, or a multipage text.
2. **Queries posed in a query language**. A query specifies what the user wants to know.
The query language can be just a list of words, such as [AI book]; or it can specify a phrase of words that must be adjacent, as in ["AI book"]; it can contain Boolean operators as in [AI AND book]; it can include non-Boolean operators such as [AI NEAR book] or [AI book site:www.aaai.org].
3. A result set. This is the subset of documents that the IR system judges to be relevant to the query. By relevant, we mean likely to be of use to the person who posed the query, for the particular information need expressed in the query.

4. A presentation of the result set. This can be as simple as a ranked list of document titles or as complex as a rotating color map of the result set projected onto a three- dimensional space, rendered as a two-dimensional display.

Boolean keyword model

Each word in the document collection is treated as a Boolean feature that is true of a document if the word occurs in the document and false if it does not. The query language is the language of Boolean expressions over features. A document is relevant only if the expression evaluates to true.

Advantage

1. Simple to explain and implement.

Disadvantages

1. The degree of relevance of a document is a single bit, so there is no guidance as to how to order the relevant documents for presentation.
2. Boolean expressions are unfamiliar to users who are not programmers or logicians.

IR scoring functions

BM25 scoring function

A scoring function takes a document and a query and returns a numeric score; the most relevant documents have the highest scores. In the BM25 function, the score is a linear weighted combination of scores for each of the words that make up the query.
Three factors affect the weight of a query term:

1. **Frequency** with which a query term appears in a document (also known as TF for term frequency). For the query [farming in Kansas], documents that mention "farming" frequently will have higher scores.
2. **Inverse document frequency** of the term, or IDF . The word "in" appears in almost every document, so it has a high document frequency, and thus a low inverse document frequency, and thus it is not as important to the query as "farming" or "Kansas."
3. **Length of the document**. A million-word document will probably mention all the query words, but may not actually be about the query. A short document that mentions all the words is a much better candidate

BM25systems create an **index** ahead of time that lists, for each vocabulary word, the documents that contain the word. This is called the **hit list** for the word. Then when given a query, we intersect the hit lists of the query words and only score the documents in the intersection.

IR system evaluation

Two measures used in the scoring: recall and precision.
Precision measures the proportion of documents in the result set that are actually relevant.
Recall measures the proportion of all the relevant documents in the collection that are in the result set
A summary of both measures is the **F_1 score**, a single number that is **the harmonic mean of precision and recall**, $2PR/(P+R)$.

IR refinements

There are many possible refinements to the system described here, and indeed Web search engines are continually updating their algorithms as they discover new approaches and as the Web grows and changes.

pivoted document length normalization scheme; the idea is that the pivot is the document length at which the old-style normalization is correct; documents shorter than that get a boost and longer ones get a penalty.

Most IR systems do **case folding** of "COUCH" to "couch," and some use a **stemming** algorithm to reduce. "couches" to the stem form "couch," both in the query and the documents. This typically yields a small increase in recall but can harm precision

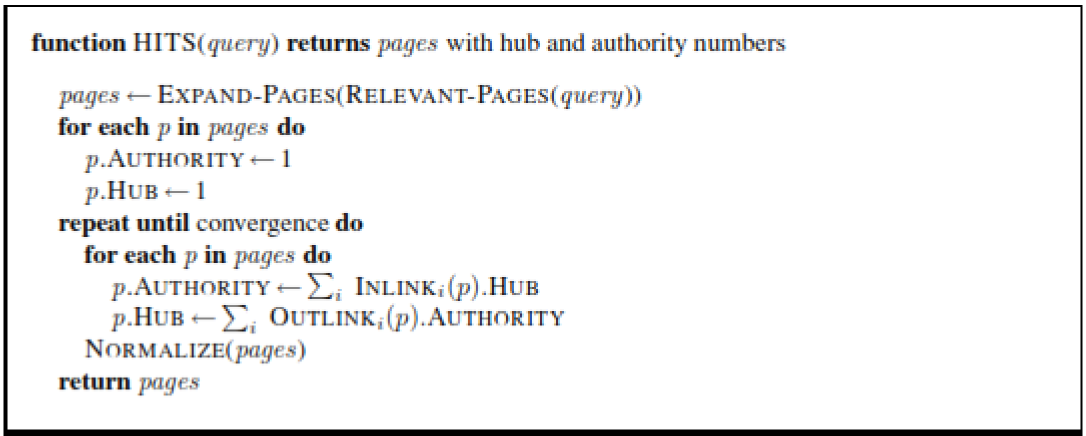
Recognize **synonyms**, such as "sofa" for "couch."
anytime there are two words that mean the same thing, speakers of the language conspire to evolve the meanings to remove the confusion.
Related words that are not synonyms also play an important role in ranking—terms like "leather", "wooden," or "modern" can serve to confirm that the document really is about "couch."

Synonyms and related words can be found in dictionaries or by looking for correlations in documents or in queries—if we find that many users who ask the query [new sofa] follow it up with the query [new couch], we can in the future alter [new sofa] to be [new sofa OR new couch]

As a final refinement, IR can be improved by considering **metadata**—data outside of the text of the document. Examples include human-supplied keywords and publication data. On the Web, hypertext **links** between documents are a crucial source of information

The PageRank algorithm

PageRank[3] was one of the two original ideas that set Google's search apart from other Web search engines when it was introduced in 1997.



```
function HITS(query) returns pages with hub and authority numbers
    pages ← EXPAND-PAGES(RELEVANT-PAGES(query))
    for each p in pages do
        p.AUTHORITY ← 1
        p.HUB ← 1
    repeat until convergence do
        for each p in pages do
            p.AUTHORITY ← $\sum_i$ INLINK$_i$(p).HUB
            p.HUB ← $\sum_i$ OUTLINK$_i$(p).AUTHORITY
        NORMALIZE(pages)
    return pages
```

Figure 22.1 The HITS algorithm for computing hubs and authorities with respect to a query. RELEVANT-PAGES fetches the pages that match the query, and EXPAND-PAGES adds in every page that links to or is linked from one of the relevant pages. NORMALIZE divides each page's score by the sum of the squares of all pages' scores (separately for both the authority and hubs scores).

PageRank was invented to solve the problem of the tyranny of TF scores: if the query is [IBM], how do we make sure that IBM's home page, `ibm.com`, is the first result, even if another page mentions the term "IBM" more frequently? The idea is that `ibm.com` has many in-links (links to the page), so it should be ranked higher: each in-link is a vote for the quality of the linked-to page.

The HITS algorithm

The Hyperlink-Induced Topic Search algorithm, also known as "Hubs and Authorities" or HITS, is another influential link-analysis algorithm.

HITS differs from PageRank in several ways.
First, it is a query-dependent measure: it rates pages with respect to a query. That means that it must be computed anew for each query—a computational burden that most search engines have elected not to take on.

Given a query, HITS first finds a set of pages that are relevant to the query.
It does that by intersecting hit lists of query words, and then adding pages in the link neighborhood of these pages—pages that link to or are linked from one of the pages in the original relevant set.

Each page in this set is considered an **authority** on the query to the degree that other pages in the relevant set point to it. A page is considered a **hub** to the degree that it points to other authoritative pages in the relevant set. Just as with PageRank, we don't want to merely count the number of links; we want to give more value to the high-quality hubs and authorities

Question answering

Information retrieval is the task of finding documents that are relevant to a query, where the query may be a question, or just a topic area or concept. **Question answering** is a somewhat different task, in which the query really is a question, and the answer is not a ranked list of documents but rather a short response—a sentence, or even just a phrase.

Once the n-grams are scored, they are filtered by expected type.
    If the original query starts with
    "who," then we filter on names of people;
    for "how many" we filter on numbers,

for “when,” on a date or time.
There is also a filter that says the answer should not be part of the question;

5. INFORMATION EXTRACTION

Information extraction is the process of acquiring knowledge by skimming a text and looking for occurrences of a particular class of object and for relationships among objects. A typical task is to extract instances of addresses from Web pages, with database fields for street, city, state, and zip code; or instances of storms from weather reports, with fields for temperature, wind speed, and precipitation.

Six different approaches to information extraction

1. Finite-state automata for information extraction
2. Probabilistic models for information extraction
3. Conditional random fields for information extraction
4. Ontology extraction from large corpora
5. Automated template construction
6. Machine reading

Finite-state automata for information extraction

The simplest type of information extraction system is an **attribute-based extraction** system that assumes that the entire text refers to a single object and the task is to extract attributes of that object.

The problem of extracting from the text “IBM ThinkBook 970. Our price: $399.00” the set of attributes {Manufacturer=IBM, Model=ThinkBook970, Price=$399.00}. We can address this problem by defining a **tem- plate** (also known as a pattern) for each attribute we would like to extract

Templates are often defined with three parts:

1. A prefix regex,
2. A target regex, and
3. A postfix regex.

For prices, the target regex is as above, the prefix would look for strings such as “price:” and the postfix could be empty. The idea is that some clues about an attribute come from the attribute value itself and some come from the surrounding text.

If a regular expression for an attribute matches the text exactly once, then we can pull
out the portion of the text that is the value of the attribute. If there is no match, all we can do is give a default value or leave the attribute missing; but if there are several matches, we need a process to choose among them. One strategy is to have several templates for each attribute, ordered by priority.

One step up from attribute-based extraction systems are **relational extraction** systems, which deal with **multiple objects and the relations among them**. Thus, when these systems see the text “$249.99,” they need to determine not just that it is a price, but also which object has that price. A typical relational-based extraction system is FASTUS, which handles news stories about corporate mergers and acquisitions

FASTUS consists of five stages:

1. **Tokenization**: segments the stream of characters into tokens
2. **Complex-word handling** : collocations such as “set up” and “joint venture,” as well as proper names such as “Bridgestone Sports Co.”
3. **Basic-group handling** : noun groups and verb groups
4. **Complex-phrase handling** : combines the basic groups, finite-state and thus can be processed quickly
5. **Structure merging** : **merges structures** that were built up in the previous step

Disadvantage:

When information extraction must be attempted from noisy or varied input, simple finite-state approaches fare poorly. It is too hard to get all the rules and their priorities right;

Probabilistic models for information extraction

. The simplest probabilistic model for sequences with hidden state is the hidden Markov model, or HMM. HMM models a progression through a sequence of hidden states, $\mathbf{x}_t$, with an observation $\mathbf{e}_t$ at each step. The observations are the words of the text, and the hidden states are whether we are in the target, prefix, or post fix part of the attribute template, or in the background.

HMMs have two big advantages over FSAs for extraction.

- First, HMMs are probabilistic, and thus tolerant to noise. In a regular expression, if a single expected character is missing, the regex fails to match; with HMMs there is graceful degradation with missing characters/words, and we get ***a probability indicating the degree of match***, not just a Boolean match/fail.
- Second HMMs can be **trained from data**; they don't require laborious engineering of templates, and thus they can more easily be kept up to date as text changes over time.

HMM templates:

- Consist of one or more target states,
- Prefix states must precede the targets,
- Postfix states most follow the targets, and
- other states must be background.

With sufficient training data, the HMM automatically learns a structure
Once the HMMs have been learned, we can apply them to a text

Conditional random fields for information extraction

- **conditional random field**, or CRF, which models a conditional probability distribution of a set of target variables given a set of observed variables
- Like Bayesian networks, CRFs can represent many different structures of dependencies among the variables. One common structure is the **linear-chain conditional random field** for repre- senting Markov dependencies among variables in a temporal sequence.
- **Conditional Random Fields** (CRFs) are the state of the art approaches taking the sequence characteristics to do better labeling. However, as the **information** on a Web page is two-dimensionally laid out, previous linear-chain CRFs have their limitations for Web **information extraction**.
- The feature functions are the key components of CRF.
- The general form of a feature function is fi(zn−1, zn, x1:N , n), which looks at a pair of adjacent states zn−1, zn, the whole input sequence x1:N , and where we are in the sequence. The feature functions produce a real value
- Features are not limited to binary functions. Any real-valued function is allowed. Designing the features of an CRF is the most important task. In CRFs for real applications it is not uncommon to have tens of thousands or more features.
- CRF formalism gives us a great deal of flexibility in defining them. This flexibility can lead to accuracies that are higher than with less flexible models such as HMMs

Ontology extraction from large corpora

Information extraction is finding a specific set of relations (e.g., speaker, time, location) in a specific text (e.g., a talk announcement). A different applica- tion of extraction technology is building a large knowledge base or ontology of facts from a corpus.

1. First it is open-ended—we want to acquire facts about all types of domains, not just one specific domain.
2. Second, with a large corpus, this task is dominated by precision, not recall—just as with question answering on the Web.
3. Third, the results can be statistical aggregates gathered from multiple sources, rather than being extracted from one specific text.

templates:

NP **such as** NP (**,** NP)* (**,**)? ((**and** | **or**) NP)? .

NP is a variable standing for a noun phrase

This template matches the texts "diseases such as rabies affect your dog" and "supports network protocols such as DNS," concluding that rabies is a disease and DNS is a network protocol.

With a large corpus we can afford to be picky; to use only the high-precision templates. We'll miss many statements of a subcategory relationship, but most likely we'll find a paraphrase of the statement somewhere else in the corpus in a form we can use.

Automated template construction

The *subcategory* relation is so fundamental that is worthwhile to handcraft a few templates to help identify instances of it occurring in natural language text.

It is possible to *learn* templates from a few examples, then use the templates to learn more examples, from which more templates can be learned, and so on. In one of the first experiments of this kind, Brin (1999) started with a data set of just five examples:

("Isaac Asimov", "The Robots of Dawn")
("David Brin", "Startide Rising")
("James Gleick", "Chaos—Making a New Science")
("Charles Dickens", "Great Expectations") ("William Shakespeare", "The Comedy of Errors")

Clearly these are examples of the author–title relation, but the learning system had no knowl- edge of authors or titles.

Each match is defined as a tuple of seven strings,

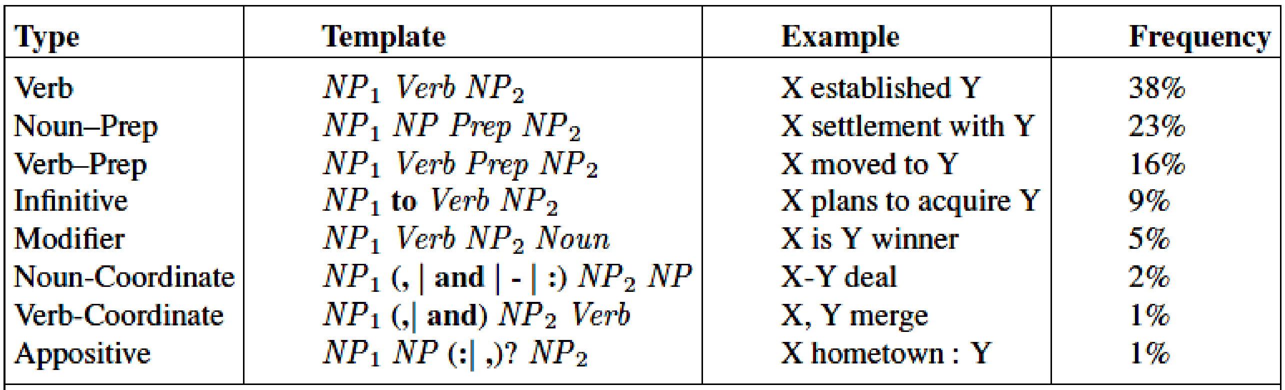
(*Author, Title, Order, Prefix, Middle, Postfix, URL*) ,

where *Order* is true if the author came first and false if the title came first, *Middle* is the characters between the author and title, *Prefix* is the 10 characters before the match, *Suffix* is the 10 characters after the match, and *URL* is the Web address where the match was made.

- Given a set of matches, a simple template-generation scheme can find templates to explain the matches.
- The biggest weakness in this approach is the sensitivity to noise. If one of the first few templates is incorrect, errors can propagate quickly.
- One way to limit this problem is to not accept a new example unless it is verified by multiple templates, and not accept a new template unless it discovers multiple examples that are also found by other templates.

Machine reading

- Automated template construction is a big step up from handcrafted template construction, but it still requires a handful of labeled examples of each relation to get started.
- To build a large ontology with many thousands of relations, even that amount of work would be onerous;
- we would like to have an extraction system with *no* human input of any kind—a system that could read on its own and build up its own database.
- Such a system would be relation-independent; would work for any relation.
- These systems work on *all* relations in parallel, because of the I/O demands of large corpora. They behave less like a traditional information- extraction system that is targeted at a few relations and more like a human reader who learns from the text itself; because of this the field has been called **machine reading**.
- Given a set of labeled examples of this type, TEXTRUNNER trains a linear-chain CRF to extract further examples from unlabeled text. The features in the CRF include function words like "to" and "of" and "the," but not nouns and verbs (and not noun phrases or verb phrases). Because TEX TRUNNER is domain-independent, it cannot rely on predefined lists of nouns and verbs.



Type	Template	Example	Frequency
Verb	NP_1 *Verb* NP_2	X established Y	38%
Noun–Prep	NP_1 *NP Prep* NP_2	X settlement with Y	23%
Verb–Prep	NP_1 *Verb Prep* NP_2	X moved to Y	16%
Infinitive	NP_1 **to** *Verb* NP_2	X plans to acquire Y	9%
Modifier	NP_1 *Verb* NP_2 *Noun*	X is Y winner	5%
Noun-Coordinate	NP_1 **(, | and | - | :)** NP_2 *NP*	X-Y deal	2%
Verb-Coordinate	NP_1 **(,| and)** NP_2 *Verb*	X, Y merge	1%
Appositive	NP_1 *NP* (:| ,)? NP_2	X hometown : Y	1%

Figure 22.3 Eight general templates that cover about 95% of the ways that relations are expressed in English.

6. NATURAL LANGUAGE PROCESSING

1. Natural Language Understanding
    - Taking some spoken/typed sentence and working out what it means
2. Natural Language Generation
    - Taking some formal representation of what you want to say and working out a way to express it in a natural (human) language (e.g., English)

Applications of NLP

- Machine Translation
- Database Access
- Information Retrieval
    - Selecting from a set of documents the ones that are relevant to a query
- Text Categorization
    - Sorting text into fixed topic categories
- Extracting data from text
    - Converting unstructured text into structured data
- Spoken language control systems
- Spelling and grammar checkers

Natural language understanding

Raw speech signal

↓ ● Speech recognition

Sequence of words spoken

↓ ● Syntactic analysis using knowledge of the grammar

Structure of the sentence

↓ ● Semantic analysis using info. about meaning of words

Partial representation of meaning of sentence

↓ ● Pragmatic analysis using info. about context

Final representation of meaning of sentence

LANGUAGE MODELS

Formal languages, such as the programming languages Java or Python, have precisely defined language models. A **language** can be defined as a set of strings; "`print(2 + 2)`" is a legal program in the language Python, whereas "`2)+(2 print`" is not. Since there are an infinite number of legal programs, they cannot be enumerated; instead they are specified by a set of rules called a **grammar**. Formal languages also have rules that define the meaning or **semantics** of a program; for example, the rules say that the "meaning" of "`2 + 2`" is 4, and the meaning of "`1/0`" is that an error is signalled

N-gram character models

Ultimately, a written text is composed of **characters**—letters, digits, punctuation, and spaces in English
A model of the probability distribution of n-letter sequences is thus called an n-**gram model**
An n-gram model is defined as a **Markov chain** of order n – 1

$$P(c_i \mid c_{1:i-1}) = P(c_i \mid c_{i-2:i-1}) .$$

For a trigram character model in a language with 100 characters, $\mathbf{P}(C_i|C_{i-2:i-1})$ has a million entries, and can be accurately estimated by counting character sequences in a body of text of
10 million characters or more. We call a body of text a **corpus** (plural *corpora*), from the
Latin word for *body*

language identification: given a text, determine what natural language it is written in One approach to language identification is to first build a trigram character model of each candidate language, $P(c_i \mid c_{i-2:i-1}, \;)$, where the variable ranges over languages. For each the model is built by counting trigrams in a corpus of that language.
Genre classification
Genre classification means deciding if a text is a news story, a legal document, a scientific article, etc. While many features help make this classification, counts of punctuation and other character n-gram features go a long way (Kessler *et al.*,
1997).
Named-entity recognition
Named-entity recognition is the task of finding names of things in a document and deciding what class they belong to. For example, in the text "Mr. Sopersteen was prescribed aciphex," we should recognize that "Mr. Sopersteen" is the name of a person and "aciphex" is the name of a drug. Character-level models are good for this task because they can associate the character sequence "ex " ("ex" followed by a space) with a drug name and "steen " with a person name, and thereby identify words that they have never seen before

Smoothing *n*-gram models

The major complication of n-gram models is that the training corpus provides only an esti- mate of the true probability distribution
we will adjust our language model so that sequences that have a count of zero in the training corpus will be assigned a small nonzero probability (and the other counts will be adjusted downward slightly so that the probability still sums to 1). The process od adjusting the probability of low-frequency counts is called **smoothing**.
simplest type of smoothing was suggested by Pierre-Simon Laplace in the 18th cen- tury: he said that, in the lack of further information, if a random Boolean variable X has been false in all n observations so far then the estimate for $P(X = true)$ should be $1/(n+2)$.

A better approach is a **backoff model**, in which we start by estimating n-gram counts, but for any particular sequence that has a low (or zero) count, we back off to $(n-1)$-grams. **Linear interpolation smoothing** is a backoff model that combines trigram, bigram, and unigram models by linear interpolation.

Model evaluation

With so many possible n-gram models—unigram, bigram, trigram, interpolated smoothing with different values of λ, etc.—how do we know what model to choose? We can evaluate a model with cross-validation Split the corpus into a training corpus and a validation corpus. Determine the parameters of the model from the training data. Then evaluate the model on the validation corpus.

The evaluation can be a task-specific metric, such as measuring accuracy on language identification. Alternatively we can have a task-independent model of language quality: cal- culate the probability assigned to the validation corpus by the model; the higher the proba- bility the better. This metric is inconvenient because the probability of a large corpus will be a very small number, and floating-point underflow becomes an issue. A different way of describing the probability of a sequence is with a measure called **perplexity**, defined as

$$\text{Perplexity}(c_{1:N}) = P(c_{1:N})^{\ N} .$$

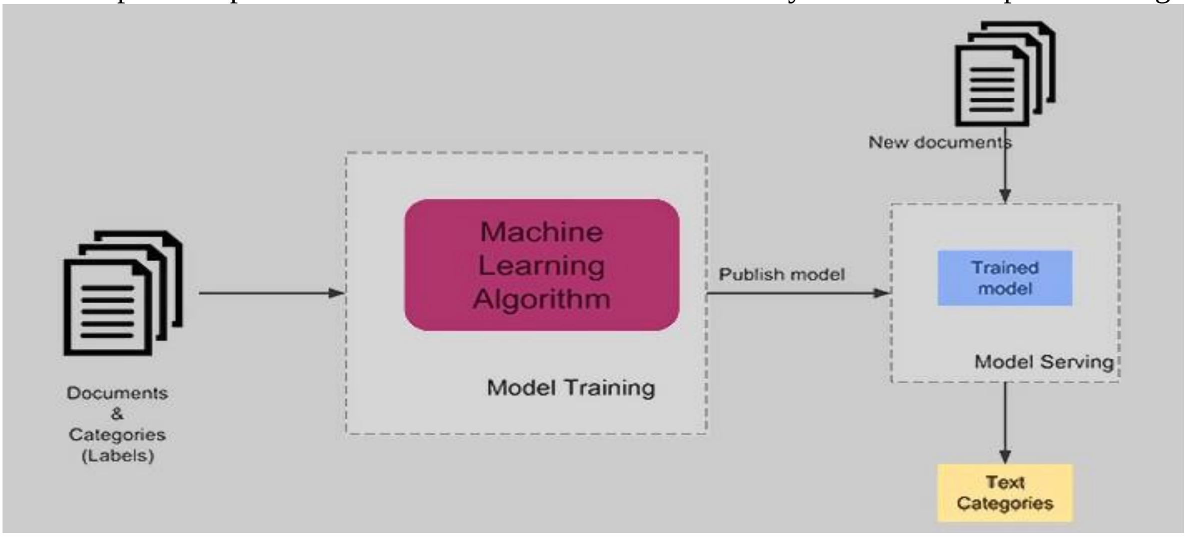
Perplexity can be thought of as the reciprocal of probability, normalized by sequence length

N-gram word models

Word n-gram models need to deal with **out of vocabulary** words With character mod- els, we didn't have to worry about someone inventing a new letter of the alphabet.[1] But with word models there is always the chance of a new word that was not seen in the training corpus, so we need to model that explicitly in our language model.

Text categorization

NLP techniques have proven successful in a related task: sorting text into fixed topic categories. There are several commercial services that provide access to news wire stories in this manner. A subscriber can ask for all the news on a particular industry, company, or geographic area, for example. The providers of these services have traditionally used human experts to assign



the categories. In the last few years, NLP systems have proven to be just as accurate, correctly categorizing over 90% of the news stories. They are also far faster and more consistent, so there has been a switch from humans to automated systems.
Text categorization is amenable to NLP techniques where IR is not because the categories are fixed, and thus the system builders can spend the time tuning their program to the problem. For example, in a dictionary, the primary definition of the word "crude" is vulgar, but in a large sample of the *Wall Street Journal*, "crude" refers to oil 100% of the time.

7. MACHINE TRANSLATION

Machine translation (MT) is automated translation. It is the process by which computer software is used to translate a text from one natural language (such as English) to another (such as Spanish).

To do translation well, a translator (human or machine) must read the **original text, understandthe situation to which it is referring, and find a corresponding text in the target languag**e that does a good job of describing the same or a similar situation.

Often this involves a choice. For example, the English word "you" can be translated into French as either the formal "vous" or the informal "tu." There is just no way that one can refer to the concept of "you" in French without also making a choice of formal or informal. Translators (both machine and human) sometimes find it difficult to make this choice.

To process any translation, human or automated, the meaning of a text in the original (source) language must be fully restored in the target language, i.e. the translation. While on the surface this seems straightforward, it is far more complex.

Translation is not a mere word-for-word substitution. A translator must interpret and analyze all of the elements in the text and know how each word may influence another. This requires extensive expertise in grammar, syntax (sentence structure), semantics (meanings), etc., in the source and target languages, as well as familiarity with each local region.

Human and machine translation each have their share of challenges. For example, no two individual translators can produce identical translations of the same text in the same language pair, and it may take several rounds of revisions to meet customer satisfaction. But the greater challenge lies in how machine translation can produce publishable quality translations.

A representation language that makes all the distinctions necessary for a set of languages is called an **interlingua.** Some systems attempt to analyze the source language text all the way into an interlingua knowledge representation and then generate sentences in the target language from that representation.

This is difficult because it involves three unsolved problems:

1. Creating a complete knowledge representation of everything;
2. Parsing into that representation; and
3. Generating sentences from that representation.

Rule-Based Machine Translation Technology

→ Rule-based machine translation relies on countless built-in linguistic rules and millions of bilingual dictionaries for each language pair.

→ The software parses text and creates a transitional representation from which the text in the target language is generated.

→ This process requires extensive lexicons with morphological, syntactic, and semantic information, and large sets of rules. The software uses these complex rule sets and then transfers the grammatical structure of the source language into the target language.

→ Translations are built on gigantic dictionaries and sophisticated linguistic rules. Users can improve the out-of-the-box translation quality by adding their terminology into the translation process. They create user-defined dictionaries which override the system's default settings.

In most cases, there are two steps:

→ An initial investment that *significantly increases the quality* at a limited cost, and

→ An *ongoing investment to increase quality incrementally*.

→ While rule-based MT brings companies to the quality threshold and beyond, the quality improvement process may be long and expensive.

Statistical Machine Translation Technology

Statistical machine translation utilizes statistical translation models whose parameters stem from the **analysis of monolingual and bilingual corpora**.

Building statistical translation models is a quick process, but **the technology relies heavily on existing multilingual corpora**.

A minimum of 2 million words for a specific domain and even more for general language are required. Theoretically it is possible to reach the quality threshold but most companies do not have such large amounts of existing multilingual corpora to build the necessary translation models.

Additionally, statistical machine translation is CPU intensive and requires an extensive hardware configuration to run translation models for average performance levels.

Rule-Based MT vs. Statistical MT

→ Rule-based MT provides *good out-of-domain quality* and is by nature predictable.

→ Dictionary-based customization guarantees improved quality and compliance with corporate terminology. But translation results may lack the fluency readers expect.

→ In terms of investment, the customization cycle needed to reach the quality threshold can be long and costly. The performance is high even on standard hardware.

Statistical MT provides good quality when large and qualified corpora are available. The translation is fluent, meaning it reads well and therefore meets user expectations. However, the translation is neither predictable nor consistent. Training from good corpora is automated and cheaper. But training on general language corpora, meaning text other than the specified domain, is poor. Furthermore, statistical MT requires significant hardware to build and manage large translation models.

Transfer model

They keep a database of translation rules (or examples), and whenever the rule (or example) matches, they translate directly.

Transfer can occur at the lexical, syntactic, or semantic level.

For example, a strictly syntactic rule maps English [*Adjective Noun*] to French [*Noun Adjective*]. A mixed syntactic and lexical rule maps French [S1 “et puis” S2] to English [S1 “and then” S2].

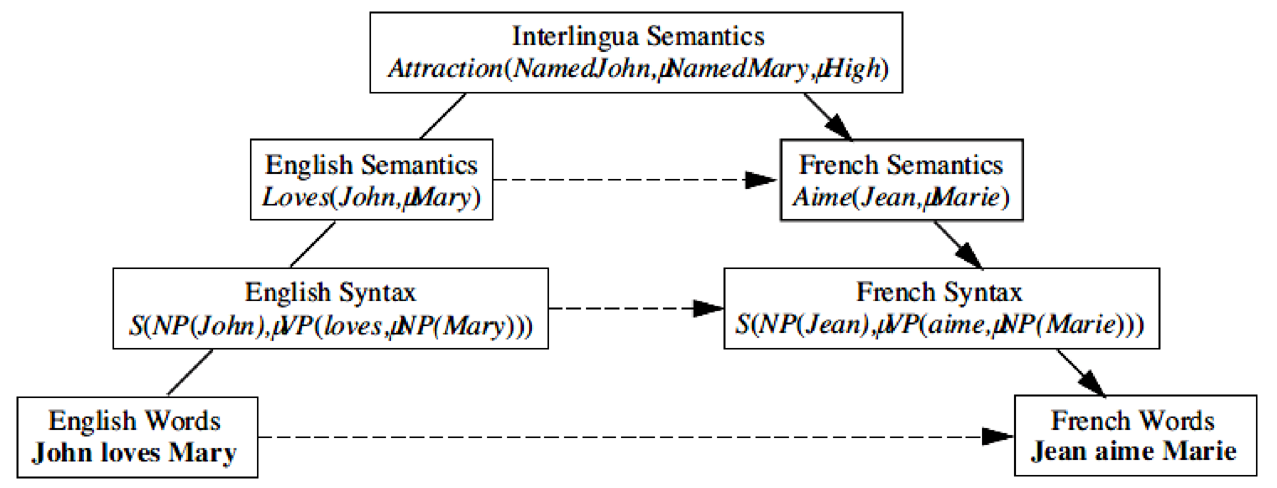


Figure 23.12 The Vauquois triangle: schematic diagram of the choices for a machine translation system (Vauquois, 1968). We start with English text at the top. An interlingua-based system follows the solid lines, parsing English first into a syntactic form, then into a semantic representation and an interlingua representation, and then through generation to a semantic, syntactic, and lexical form in French. A transfer-based system uses the dashed lines as a shortcut. Different systems make the transfer at different points; some make it at multiple points.

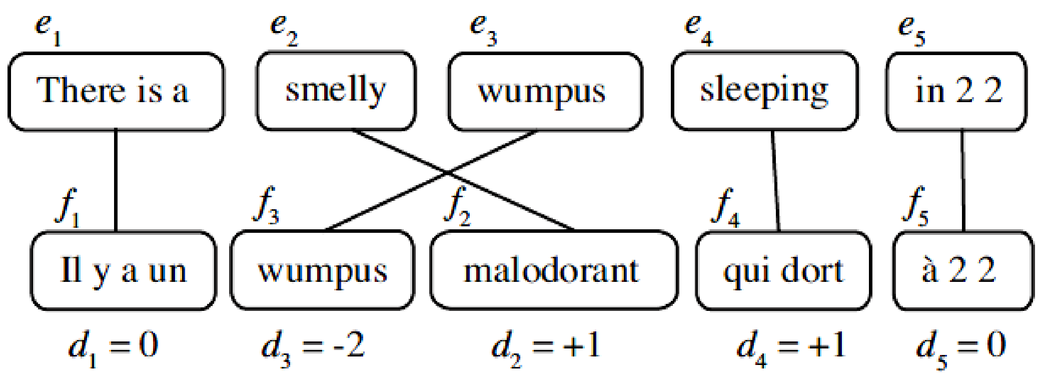


Figure 23.13 Candidate French phrases for each phrase of an English sentence, with distortion (d) values for each French phrase.

All that remains is to learn the phrasal and distortion probabilities

1. Find parallel texts
2. Segment into sentences
3. Align sentences
4. Align phrases
5. Extract distortions
6. Improve estimates with EM

EXAMPLE

TAUM-METEO system, developed by the University of Montreal, which translates weather reports from English to French. It works because the language used in these government weather reports is highly stylized and regular.

A representative system is SPANAM (Vasconcellos and Leon, 1985), which can translate a Spanish passage into English of this quality.

→ This is mostly understandable, but not always grammatical and rarely fluent.

→ Another possibility is to invest the human effort on pre-editing the original document.
→ If the original document can be **made to conform to a restricted subset of English** (or whatever the original language is), then it can sometimes be translated without the need for post-editing.
→ This approach is particularly cost-effective when there is a need to translate one document into many languages, as is the case for legal documents

The problem is that different languages categorize the world differently. A majority of the situations that are covered by the English word "open" are also covered by the German word "offen," but the boundaries of the category differ across languages.
In English, we extend the basic meaning of "open" to cover open markets, open questions, and open job offerings.
In German, the extensions are different. Job offerings are "freie," not open, but the concepts of loose ice, private firms, and blank checks all use a form of "offen."

8. DATABASE ACCESS
The first major success for natural language processing (NLP) was in the area of database access.
The disadvantage is that the user never knows which wordings of a query will succeed and which are outside the system's competence some commercial systems have built up large enough grammars and lexicons to handle a fairly wide variety of inputs. The main challenge for current systems is to follow the context of an interaction. The user should be able to ask a series of questions where some of them implicitly refer to earlier questions or answers:

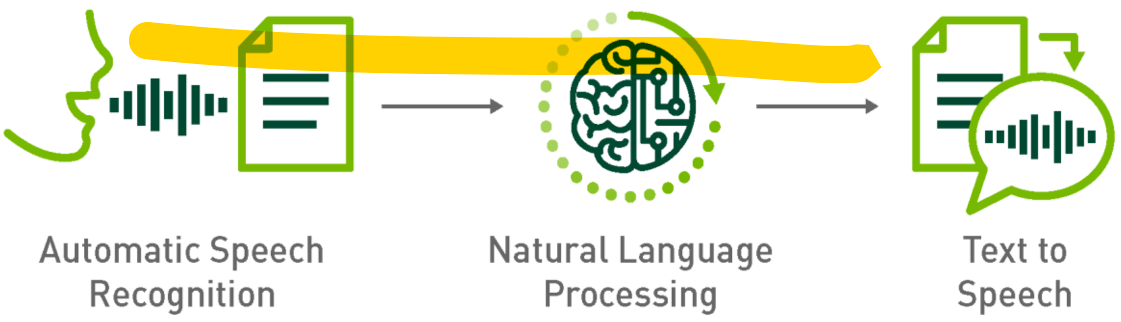
What countries are north of the equator?
How about south?
Show only the ones outside Australasia.
What is their total area?
Some systems (e.g., TEAM (Grosz *et al.*, 1987)) handle problems like this to a limited degree,

Natural Language Inc. and Symantec are still selling database access tools that use natural language, but customers are less likely to make their buying decisions based on the strength of the natural language component than on the graphical user interface or the degree of integration of the database with spreadsheets and word processing.
Natural language is not always the most natural way to communicate: sometimes it is easier to point and click with a mouse to express an idea.
The emphasis in practical NLP has now shifted away from database access to the broad **interpretation.**

In part, this is a reflection of a change in the computer industry.
In the early 1980s, most online information was stored in databases or spreadsheets. Now the majority of online information is text: email, news, journal articles, reports, books, encyclopedias. Most computer users find there is too much information available, and not enough time to sort through it. Text interpretation programs help to retrieve, categorize, filter, and extract information from text. Text interpretation systems can be split into three types: information retrieval, text categorization, and data extraction.

9. SPEECH RECOGNITION
It is the task of mapping from a digitally encoded acoustic signal to a string of words. **Speech understanding** is the task of mapping from the acoustic signal all the way to an interpretation of the meaning of the utterance.
A speech understanding system must answer three questions:

1. What speech sounds did the speaker utter?
2. What words did the speaker intend to express with those speech sounds?
3. What meaning did the speaker intend to express with those words?



To answer question 1, we have to first decide what a speech sound is. It turns out that all human languages use a limited repertoire of about 40 or 50 sounds, called **phones.** Roughly speaking, a phone is the sound that corresponds to a single vowel or consonant, but there are some complications:

Combinations of letters such as "th" and "ng" produce single phones, and some letters produce different phones in different contexts (for example, the "a" in *rat* and *rate*. Once we know what the possible sounds are, we need to characterize them in terms of features that we can pick out of the acoustic signal, such as the frequency or amplitude of the sound waves.

Question 2 is conceptually much simpler. You can think of it as looking up words in a dictionary that is arranged by pronunciation.
We get a sequence of three phones, *[k], [ce],* and *ft],* and find in the dictionary that this is the pronunciation for the word "cat."

Two things make this difficult.
The first is the existence of **homophones,** different words that sound the same, like "two" and "too."1
The second is **segmentation,** the problem of deciding where one word ends and the next begins.

Question 3 we already know how to answer—use the parsing and analysis algorithms
Some speech understanding systems extract the most likely string of words and pass them directly to an analyzer. Other systems have a more complex control structure that considers multiple possible word interpretations so that understanding can be achieved even if some individual words are not recognized correctly.

Signal processing

Sound is an analog energy source. When a sound wave strikes a microphone, it is converted to an electrical current, which can be passed to an analog-to-digital converter to yield a stream of bits representing the sound.

We have two choices in deciding how many bits to keep.
First, the **sampling rate** is the frequency with which we look at the signal. For speech, a sampling rate between 8 and 16 KHz (i.e., 8 to 16,000 times per second) is typical. Telephones deliver only about 3 KHz.
Second, the **quantization factor** determines the precision to which the energy at each sampling point is recorded. Speech recognizers typically keep 8 to 12 bits. That means that a low-end system, sampling at 8 KHz with 8-bit quantization, would require nearly half a megabyte per minute of speech.

The first step in coming up with a better representation for the signal is to group the samples together into larger blocks called ***frames.*** This makes it possible to analyze the whole frame for the appearance of speech phenomena such as a rise or drop in frequency, or a sudden onset or cessation of energy. Within each frame, we represent what is happening with a vector **of features.**
Figure 24.33 shows frames with a vector of three features. Note that the frames overlap; this prevents us from losing information if an important acoustic event just happens to fall on a frame boundary.

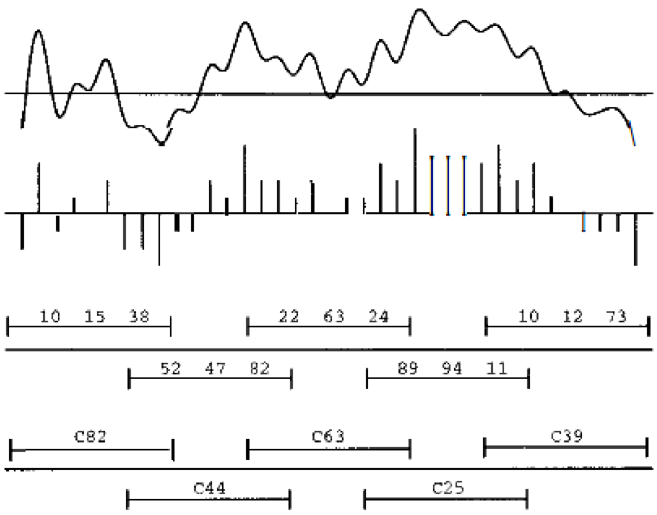


Figure 24.33 Translating the acoustic signal into a sequence of vector quantization values. (Don't try to figure out the numbers; they were assigned arbitrarily.)

The final step in many speech signal processing systems is **vector quantization.** If there are *n* features in a frame, we can think of this as an n-dimensional space containing many points.
Some information is lost in going from a feature vector to a label that summarizes a whole neighborhood around the vector, but there are automated methods for choosing an optimal quantization of the feature vector space so that little or no inaccuracy is introduced (Jelinek, 1990).

First, we end up with a representation of the speech signal that is compact. But more importantly, we have a representation that is likely to encode features of the signal that will be useful for word recognition. A given speech sound can be pronounced so many ways: loud or soft, fast or slow, high-pitched or low, against a background of silence or noise, and by any of millions of different speakers each with different accents and vocal tracts. Signal processing hopes to capture enough of the important features so that the commonalities that define the sound can be picked out from this backdrop of variation.
The dual problem, **speaker identification,** requires one to focus on the variation instead of the commonalities in order to decide who is speaking.

Defining the overall speech recognition model
Speech recognition is the diagnostic task of recovering the words that produce a given acoustic signal. It is a classic example of reasoning with uncertainty. We are uncertain about how well the microphones (and digitization hardware) have captured the actual sounds, we are uncertain about which phones would give rise to the signal, and we are uncertain about which words would give rise to the phones. As is often the case, the diagnostic task can best be approached with a causal model—the words cause the signal.

We can break this into components with Bayes' rule:

$$P(words|signal) = \frac{P(words)P(signal|words)}{P(signal)}$$

Given a *signal*, our task is to find the sequence of *words* that maximizes *P(words\signal).* Of the three components on the right-hand side, *P(signal)* is a normalizing constant that we can ignore. *P(words)* is known as the **language model.** It is what tells us, when we are not sure if we heard

"bad boy" or "pad boy" that the former is more likely. Finally, *P(signal\words)* is the **acoustic model.** It is what tells us that "cat" is very likely to be pronounced *[kcet]*.

<u>The language model: P(words)</u>

Assign a probability to each of the (possibly infinite) number of strings. Context-free grammars are no help for this task, but probabilistic context-free grammars (PCFGs) are promising.
Bigram model : it says that the probability of any given word is determined solely by the previous word in the string

Trigrammodel that provides values for />(w;|w,-_iw/_2). This is a more powerful language model, capable of determining that "ate a banana" is more likely than "ate a bandana." The problem is that there are so many more parameters in trigram models that it is hard to get enough training data to come up with accurate probability estimates.

<u>The acoustic model: P(signallwords)</u>

The acoustic model is responsible for saying what sounds will be produced when a given string of words is uttered. We divide the model into two parts. First, we show how each word is described as a sequence of phones, and then we show how each phone relates to the vector quantization values extracted from the acoustic signal.

<u>Markov model</u>

Two sources of phonetic
First, different dialects have different pronunciations.
The top of Figure. gives an example of this: for "tomato," you say [tow mey tow] and I say [tow maa tow]. The alternative pronunciations are specified as a **Markov model.**

In general, a **Markov model is a way of describing a process that goes through a series of states**. The model describes all the possible paths through the state space and assigns a probability to each one. The probability of transitioning from the current state to another one depends only on the current state, not on any prior part of the path.

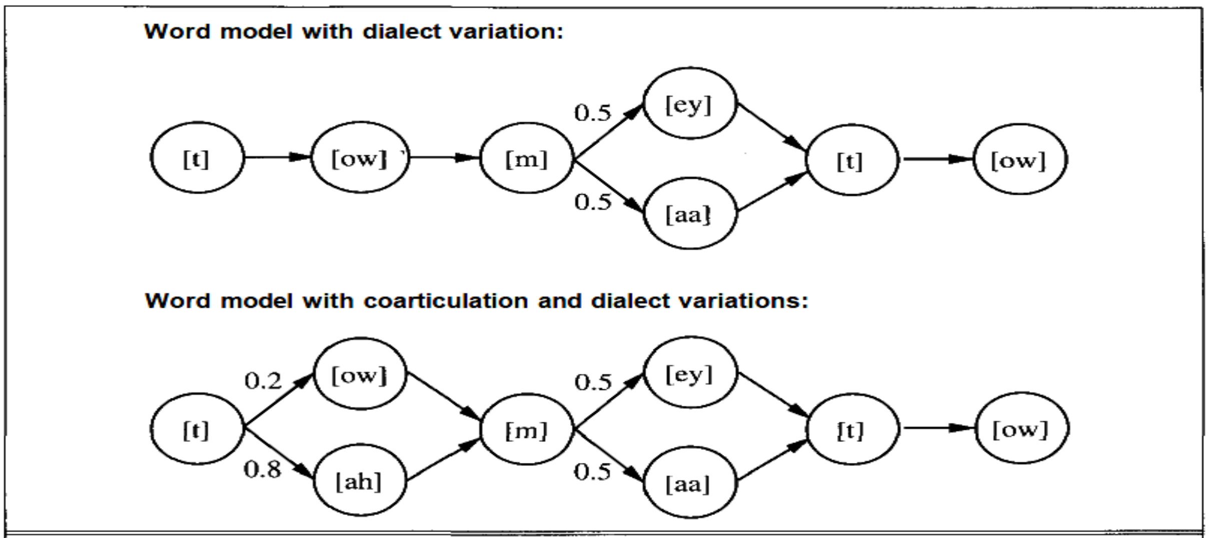

Word model with dialect variation:
[t]
[ow]
[m]
0.5
[ey]
0.5
[aa]
[t]
[ow]
Word model with coarticulation and dialect variations:
[t]
0.2
[ow]
0.8
[ah]
[m]
0.5
[ey]
0.5
[aa]
[t]
[ow]

Figure 24.35 Two pronunciation models of the word "tomato." The top one accounts for dialect differences. The bottom one does that and also accounts for a coarticulation effect.

Markov model with seven states (circles), each corresponding to the production of a phone. The arrows denote allowable transitions between states, and each transition has a probability associated with it.3 There are only two possible paths through the model, one corresponding to the phone sequence [t ow m ey t ow] and the other to [t ow m aa tow]. The probability of a path is the product of the probabilities on the arcs that make up the path. In this case, most of the arc probabilities are 1 and we have

P([towmeytow] ("tomato") = P([r«w»iaa?ow] ("tomato") = 0.5

$$P([towmeytow] | \text{“tomato”}) = P([towmaatow] | \text{“tomato”}) = 0.5$$

*The second source of phonetic variation is **coarticulation.***

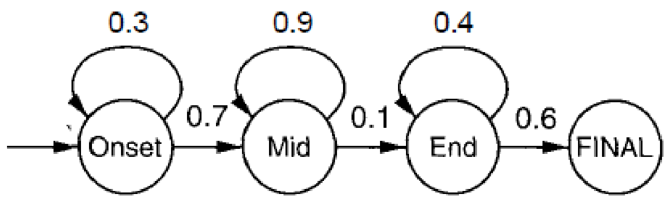
Similar models would be constructed for every word we want to be able to recognize. Now if the speech signal were a list of phones, then we would be done with the acoustic model. We could take a given input signal (e.g., [towmeytow]) and compute *P(signal\words)* for various word strings (e.g., "tomato," "toe may tow," and so on). We could then combine these with *P(words)* values taken from the language model to arrive at the *words* that maximize *P(words\signal).*

<u>hidden Markov model</u> <u>or</u> **<u>HMM</u>**

Unfortunately, signal processing does not give us a string of phones. So all we can do so far is maximize *P(words\phones).* Figure 24.36 shows how we can compute *P(signal\phone)* using a model called a **hidden Markov model** or **HMM.** The model is for a particular phone,[m], but all phones will have models with similar topology.

A hidden Markov model is just like a regular Markov model in that it describes a process that goes through a *sequence of states*. The difference is that in a regular Markov model, the output is a sequence of state names, and because each state has a unique name, the output uniquely determines the path through the model.
 In a hidden Markov model, each state has *a probability distribution of possible outputs*, and the same output can appear in more than one state.4
HMMs are called hidden models because the true state of the model is hidden from the observer. In general, when you see that an HMM outputs some symbol, you can't be sure what state the symbol came from.

Phone HMM for [m]:



Output probabilities for the phone HMM:

Onset:	Mid:	End:
C1: 0.5	C3: 0.2	C4: 0.1
C2: 0.2	C4: 0.7	C6: 0.5
C3: 0.3	C5: 0.1	C7: 0.4

Figure 24.36 An HMM for the phone [m]. Each state has several possible outputs, each with its own probability.

Actually, most phones have a duration of 50-100 milliseconds, or 5-10 frames at 10 msec/frame. So the [C1,C4,C6J sequence is unusually quick. Suppose we have a more typical speaker who generates the sequence [C1,C1,C4,C4,C6,C6J while producing the phone. It turns out there are two paths through the model that generate this sequence. In one of them both C4s come from the Mid state (note the arcs that loop back), and in the other the second C4 comes from the End state. We calculate the probability that this sequence came from the [m] model in the same way: take the sum over all possible paths of the probability of the path times the probability that the path generates the sequence.

P([Cl,C\,C4, C4, C6, C6]jLm]) =
(0.3 x 0.7 x 0.9 x 0.1 x 0.4 x 0.6) x (0.5 x 0.5 x 0.7 x 0.7 x 0.5 x 0.5) +
(0.3 x 0.7 x 0.1 x 0.4 x 0.4 x 0.6) x (0.5 x 0.5 x 0.7 x 0.1 x 0.5 x 0.5)
= 0.0001477

Putting the models together
We have described three models.

The language bigram model gives us *P(wordi\wordi-\}.*
The word pronunciation HMM gives us *P(phones\word).*
The phone HMM gives us *P(signal\phone).*

If we want to compute *P(words\signal),* we will need to combine these models in some way. Oneapproach is to combine them all into one big HMM. The bigram model can be thought of as an HMM in which every state corresponds to a word and every word has a transition arc to every other word. Now replace each word-state with the appropriate word model, yielding a bigger model in which each state corresponds to a phone. Finally, replace each phone-state with the appropriate phone model, yielding an even bigger model in which each state corresponds to a distribution of vector quantization values.

Some speech recognition systems complicate the picture by dealing with coarticulation effects at either the word/word or phone/phone level. For example, we could use one phone model for [ow] when it follows a [t] and a different model for [ow] when it follows a [g]. There are many trade-offs to be made—a more complex model can handle subtle effects, but it will be harder to train. Regardless of the details, we end up with one big HMM that can be used to compute*P(words\signal).*

10. ROBOT (HARDWARE –PERCEPTION –PLANNING –MOVING)

Robots are physical agents that perform ROBOT tasks by manipulating the physical world. To do so they are equipped with **effectors** such as legs, wheels, joints, and grippers Effectors have a single purpose: to assert physical forces on the environment.

Robots are also equipped with **sensors**, which allow them to perceive their environment. Present day robotics employs a diverse set of sensors, including cameras and lasers to measure the environment, and gyroscopes and accelerometers to measure the robot's own motion.
Manipulators, or robot arms are physically anchored to their workplace, for example in a factory assembly line or on the International Space Station.

<u>Mobile robot</u>

Mobile robots move about their environment using wheels, legs, or similar mechanisms. They have been put to use delivering food in hospitals, moving containers at loading docks, and similar tasks. **Unmanned ground vehicles**, or UGVs, drive autonomously on streets, highways, and off-road. The **planetary rover** explored Mars for a period of 3 months in 1997. Other types of mobile robots include **unmanned air vehicles** (UAVs), commonly used for surveillance, crop-spraying,

Autonomous underwater vehicles (AUVs) are used in deep sea exploration. Mobile robots deliver packages in the workplace and vacuum the floors at home.
The third type of robot combines mobility with manipulation, and is often called a **mobile manipulator**. **Humanoid robots** mimic the human torso.

The field of robotics also includes prosthetic devices (artificial limbs, ears, and eyes for humans), intelligent environments (such as an entire house that is equipped with sensors and effectors), and multibody systems, wherein robotic action is achieved through swarms of small cooperating robots. Real robots must cope with environments that are partially observable, stochastic, dynamic, and continuous.

Robotics brings together many of the concepts, including probabilistic state estimation, perception, planning, unsupervised learning, and reinforcement learning.

ROBOT HARDWARE

Sensors
Sensors are the perceptual interface between robot and environment.
Passive sensors, such as cameras, are true observers of the environment: they capture signals that are generated by other sources in the environment.
Active sensors, such as sonar, send energy into the environment.

They rely on the fact that this energy is reflected back to the sensor. Active sensors tend to provide more information than passive sensors, **Range finders** are sensors that measure the distance to nearby objects.

Sonar sensors emit directional sound waves, which are reflected by objects, with some of the sound making it back into the sensor. The time and intensity of the returning signal indicates the distance to nearby objects.

Sonar is the technology of choice for autonomous underwater vehicles. **Stereo vision** (see Section 24.4.2) relies on multiple cameras to image the environment from slightly different viewpoints, analyzing the resulting parallax in these images to compute the range of surrounding objects. For mobile ground robots, sonar and stereo vision are now rarely used, because they are not reliably accurate.

Time of flight camera :Most ground robots are now equipped with optical range finders. Just like sonar sensors, optical range sensors emit active signals (light) and measure the time until a reflection

Scanning lidars tend to provide longer ranges than time of flight cameras, and tend to perform better in bright daylight.

End of range sensing are **tactile sensors** such as whiskers, bump panels, and touch-sensitive skin. These sensors measure range based on physical contact, and can be deployed only for sensing objects very close to the robot.
<u>location sensors</u>
Most location sensors use range sensing as a primary component to determine location. Outdoors, the **Global Positioning System** (GPS) is the most common solution to the localization problem. GPS the distance to satellites that emit pulsed signals. At present, there are 31 satellites in orbit, transmitting signals on multiple frequencies. GPS receivers can recover the distance to these satellites by analyzing phase shifts. By triangulating signals from multiple satellites, GPS receivers can determine their absolute location on Earth to within a few meters.

Differential GPS involves a second ground receiver with known location, providing millimeter accuracy under ideal conditions. Unfortunately, GPS does not work indoors or underwater.

Proprioceptive sensors, which inform the robot of its own motion. To measure the exact configuration of a robotic joint, motors are often equipped with **shaft decoders** that count the revolution of motors in small increments. On mobile robots, shaft decoders that report wheel revolutions can be used for **odometry**—the measurement of distance traveled.

Inertial sensors, such as gyroscopes, rely on the resistance of mass to the change of velocity. They can help reduce uncertainty.

Other important aspects of robot state are measured by **force sensors** and **torque sensors**. These are indispensable when robots handle fragile objects or objects whose exact shape and location is unknown. Imagine a one-ton robotic manipulator screwing in a light bulb. It would be all too easy to apply too much force and break the bulb. Force sensors allow the robot to sense how hard it is gripping the bulb, and torque sensors allow it to sense how hard it is turning. Good sensors can measure forces in all three translational and three rotational directions. They do this at a frequency of several hundred times a second, so that a robot can quickly detect unexpected forces and correct its actions before it breaks a light bulb.

Effectors

Effectors are the means by which robots move and change the shape of their bodies. To understand the design of effectors, it will help to talk about motion and shape in the abstract, using the concept of a **degree of freedom** (DOF)

- For nonrigid bodies, there are additional degrees of freedom within the robot itself.
- Robot joints also have one, two, or three degrees of freedom each.
- Six degrees of freedom are required to place an object, such as a hand, at a particular point in a particular orientation.

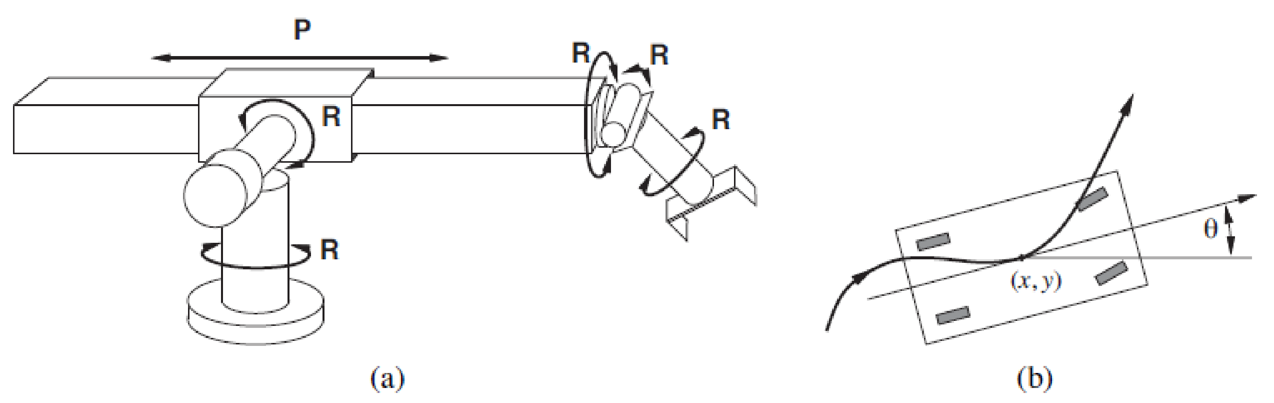


Figure 25.4 (a) The Stanford Manipulator, an early robot arm with five revolute joints (R) and one prismatic joint (P), for a total of six degrees of freedom. (b) Motion of a nonholonomic four-wheeled vehicle with front-wheel steering.

the car has three **effective degrees of freedom** but two **control** DOF **lable degrees of freedom**. **Differential drive** robots possess two independently actuated wheels (or tracks), one on each side, as on a military tank.

Legs, unlike wheels, can handle rough terrain. However, legs are notoriously slow on flat surfaces, and they are mechanically difficult to build.

A robot that can remain STABLE upright without moving its legs is called **statically stable**. A robot is statically stable if its center of gravity is above the polygon spanned by its legs.

C omplete robot also needs a source of power to drive its effectors. The **electric motor** is the most popular mechanism for both manipulator actuation and locomotion, but **pneumatic actuation** using compressed gas and **hydraulic actuation** using pressurized fluids also have their application niches.

11. ROBOTIC PERCEPTION

Perception is the process by which robots map sensor measurements into internal representations of the environment. Perception is difficult because sensors are noisy, and the environment is partially observable, unpredictable, and often dynamic.

The robot's own past actions as observed variables in the model. Figure 25.7 shows the notation used in this chapter: **X**t is the state of the environment (including the robot) at time t, **Z**t is the observation received at time t, and At is the action taken after the observation is received.

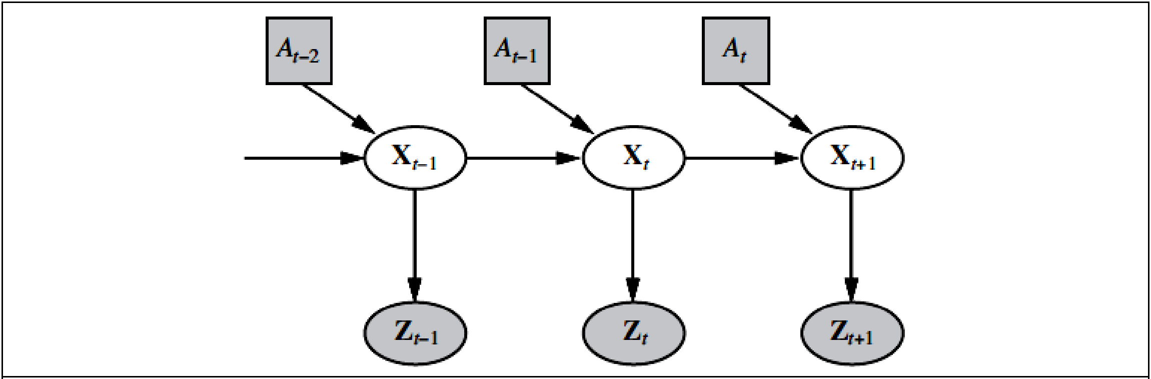


Figure 25.7 Robot perception can be viewed as temporal inference from sequences of actions and measurements, as illustrated by this dynamic Bayes network.

We would like to compute the new belief state, **P**(**X**t+1 | **z**1:t+1, a1:t), from the current belief state **P**(**X**t | **z**1:t, a1:t−1) and the new observation **z**t+1.
we condition explicitly on the actions as well as the observations, and we deal with *continuous* rather than *discrete* variables.

The probability **P**(**X**t+1 | **x**t, at) : is the **transition model** or **motion model**, and **P**(**z**t+1 | **X**MOTION MODEL t+1) : is the **sensor model**.

Localization and mapping

Localization is the problem of finding out where things are—including the robot itself. Knowledge about where things are is at the core of any successful physical interaction with the environment. For example, robot manipulators must know the location of objects they seek to manipulate; navigating robots must know where they are to find their way around.

Monte Carlo localization

Localization using particle filtering is called **Monte Carlo localization**, or MCL. The MCL algorithm is an instance of the particle-filtering algorithm. All we need to do is supply the appropriate motion model and sensor model.

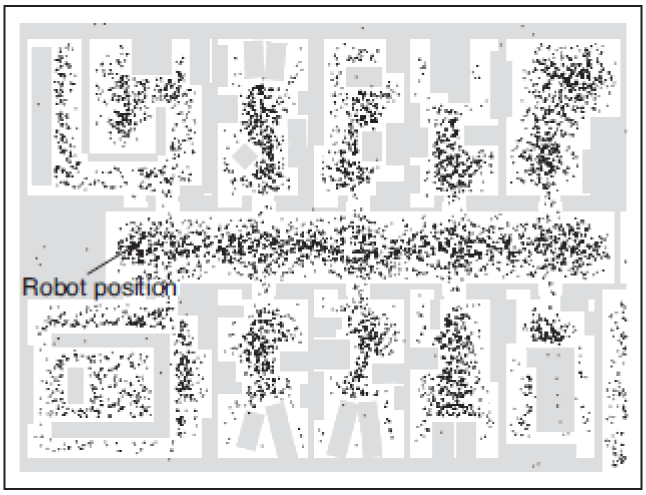
The operation of the algorithm is illustrated in as the robot finds out where it is inside an office building. In the first image, the particles are uniformly distributed based on the prior, indicating global uncertainty about the robot's position. In the second image, the first set of measurements arrives and the particles form clusters in the areas of high posterior belief. In the third, enough measurements are available to push all the particles to a single location.

Kalman filter

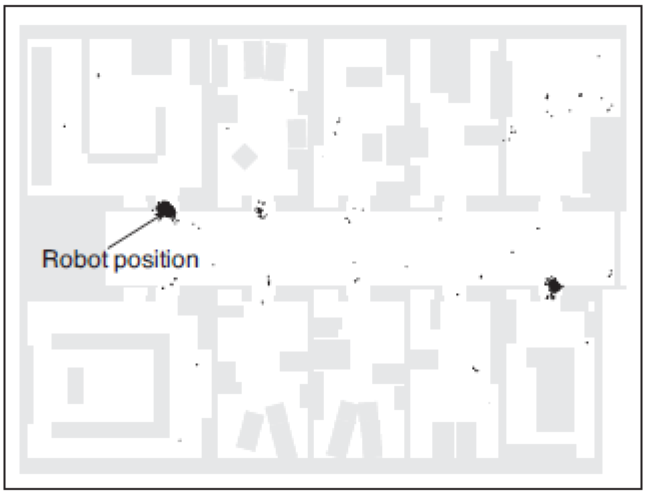
The Kalman filter is the other major way to localize. A Kalman filter represents the posterior **P**(**X**t | **z**1:t, a1:t−1) by a Gaussian. The mean of this Gaussian will be denoted **μ** t and its covariance **Σ**t. The main problem with Gaussian beliefs is that they are only closed under linear motion models f and linear measurement models h. localization algorithms using the Kalman LINEARIZATION filter **linearize** the motion and sensor models.

The problem of needing to know the identity of landmarks is an instance of the **data association** problem

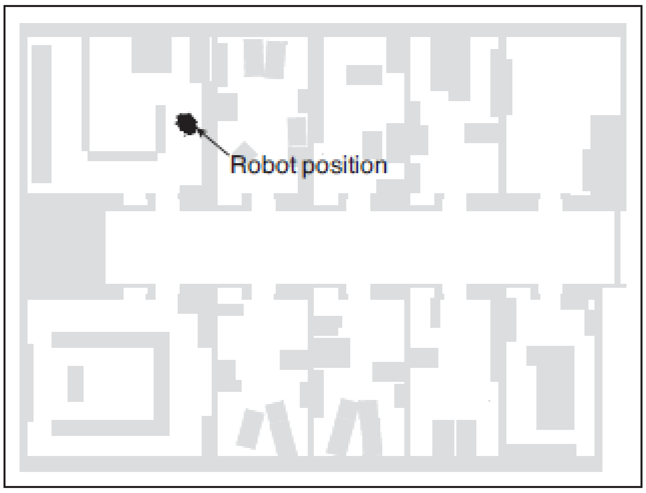
In some situations, no map of the environment is available. Then the robot will have to acquire a map. This is a bit of a chicken-and-egg problem: the navigating robot will have to determine its location relative to a map it doesn't quite know, at the same time building this map while it doesn't quite know its actual location. This problem is important for many robot applications, and it has been studied extensively under the name **simultaneous localization and mapping**, abbreviated as **SLAM**.



Monte Carlo localization, a particle filtering algorithm for mobile robot localization.
(a) Initial, global uncertainty.



(b) Approximately bimodal uncertainty after navigating in the (symmetric) corridor.



(c) Unimodal uncertainty after entering a room and finding it to be distinctive.

Machine learning in robot perception

Machine learning plays an important role in robot perception. One common approach is to map high dimensional sensor streams into lower-dimensional spaces using unsupervised machine learning methods

Another machine learning technique enables robots to continuously adapt to broad changes in sensor measurements. Adaptive perception techniques enable robots to adjust to such changes.

Methods that make robots collect their own training data (with labels!) are called **selfsupervised**. In this instance, the robot SELF-SUPERVISED uses machine learning to leverage a short-range sensor that works well for terrain classification into a sensor that can see much farther.

12. ROBOT : MOVING

All of a robot's deliberations ultimately come down to deciding how to move effectors. The **point-to-point motion** problem is to deliver the robot or its end effector to a designated target location.

We begin by finding a suitable representation in which motion-planning problems can be described and solved. It turns out that the **configuration space**—the space of robot states defined by location, orientation, and joint angles—is a better place to work than the original PATH PLANNING 3D space. The **path planning** problem is to find a path from one configuration to another in configuration space.

There are two main approaches: **cell decomposition** and **skeletonization**.

Configuration space

This raises the question of how to map between workspace coordinates and configuration space. Transforming configuration space coordinates into workspace coordinates is simple: it involves a series of straightforward coordinate transformations. These transformations are linear for prismatic joints and trigonometric for revolute joints. This chain of coordinate transformation is known as **kinematics**.

The inverse problem of calculating the configuration of a robot whose effector location is specified in workspace coordinates is known as **inverse kinematics**. The second problem with configuration space representations arises from the obstacles that may exist in the robot's workspace

Cell decomposition methods

first CELL approach to path planning uses **cell decomposition**—that is, it decomposes the free space into a finite number of contiguous regions, called cells. The path-planning problem then becomes a discrete graph-search problem, very much like the search problems Such a decomposition has the advantage that it is extremely simple to implement, but it also suffers from three limitations.

First, it is workable only for low-dimensional configuration spaces,
Second, there is the problem of what to do with cells that are "mixed"—that is, neither entirely within free space nor entirely within occupied space.
Third, any path through a discretized state space will not be smooth. It is generally difficult to guarantee that a smooth solution exists near the discrete path.

Skeletonization methods

The second major family of path-planning algorithms is based on the idea of **skeletonization**. These algorithms reduce the robot's free space to a one-dimensional representation, for which the planning problem is easier. This lower-dimensional representation is called a **skeleton** of the configuration space.

It is easy to show that this can always be achieved by a straight-line motion in configuration space.
it is a **Voronoi graph** of the free space—the set of all points that are equidistant to two or more obstacles. To do path planning with a Voronoi graph, the robot first changes its present configuration to a point on the Voronoi graph. It is easy to show that this can always be achieved by a straight-line motion in configuration space. Second, the robot follows the Voronoi graph until it reaches the point nearest to the target configuration. Finally, the robot leaves the Voronoi graph and moves to the target. Again, this final step involves straight-line motion in configuration space.

An alternative to the Voronoi graphs is the **probabilistic roadmap**, a skeletonization approach that offers more possible routes, and thus deals better with wide-open spaces.

13. PLANNING UNCERTAIN MOVEMENTS

None of the robot motion-planning algorithms discussed thus far addresses a key characteristic of robotics problems: *uncertainty*. In robotics, uncertainty arises from partial observability of the environment and from the stochastic (or unmodeled) effects of the robot's actions.

The field of robotics has adopted a range of techniques for accommodating uncertainty. If the robot faces uncertainty only in its state transition, but its state is fully observable, the problem is best modeled as a Markov decision process (MDP). The solution of an MDP is an optimal **policy**, which tells the robot what to do in every possible state.

For example, the **coastal navigation** heuristic requires the robot to stay near known landmarks to decrease its uncertainty. Another approach applies variants of the probabilistic roadmap planning method to the belief space representation. Such methods tend to scale better to large discrete POMDPs.

Robust methods

Uncertainty can also be handled using so-called **robust control** methods rather than probabilistic methods. A robust method is one that assumes a *bounded* amount of uncertainty in each aspect of a problem, but does not assign probabilities to values within the allowed interval. A robust solution is one that works no matter what actual values occur, provided they are within the assumed interval.

Fine-motion planning involves moving a robot arm in very close proximity to a static environment object.

A fine-motion plan consists of a series of **guarded motions**. The termination conditions are contact with a surface. To model uncertainty in control, we assume that instead of moving in the commanded direction, the robot's actual motion lies in the cone Cv about it.

Dynamics and control

The transition model for a dynamic state representation includes the effect of forces on this rate of change. the dynamic state has higher dimension than the kinematic space, and the curse of dimensionality would render many motion planning algorithms inapplicable for all but the most simple robots. For this reason, practical robot system often rely on simpler kinematic path planners.

A common technique to compensate for the limitations of kinematic plans is to use a CONTROLLER separate mechanism, a **controller**, for keeping the robot on track.

Potential-field control

potential fields as an additional cost function in robot motion planning, but they can also be used for generating robot motion directly, dispensing with the path planning phase altogether.

Define an attractive force that pulls the robot towards its goal configuration and a repellent potential field that pushes the robot away from obstacles.
No planning was involved in generating the potential field
The potential field can be calculated efficiently for any given configuration. Moreover, optimizing the potential amounts to calculating the gradient of the potential for the present robot configuration. These calculations can be extremely efficient, especially when compared to path-planning algorithms, all of which are exponential in the dimensionality of the configuration space (the DOFs) in the worst case.

Potential field control is great for local robot motion but sometimes we still need global planning.

Reactive control
control decisions that require some model of the environment for constructing either a reference path or a potential field.
First, models that are sufficiently accurate are often difficult to obtain, especially in complex or remote environments, such as the surface of Mars, or for robots that have few sensors. Second, even in cases where we can devise a model with sufficient accuracy, computational difficulties and localization error might render these techniques impractical.

A reflex agent architecture using **reactive control** is more appropriate
It is possible, nonetheless, to specify a controller directly without an explicit environmental model.

Variants of this simple feedback-driven controller have been found to generate remarkably robust walking patterns, capable of maneuvering the robot over rugged terrain

Reinforcement learning control
One particularly exciting form of control is based on the **policy search** form of reinforcement Learning. Policy search needs an accurate model of the domain before it can find a policy.

The controls are the manual controls of of the helicopter: throttle, pitch, elevator, aileron, and rudder. All that remains is the resulting state—how are we going to define a model that accurately says how the helicopter responds to each control? The answer is simple: Let an expert human pilot fly the helicopter, and record the controls that the expert transmits over the radio and the state variables of the helicopter. About four minutes of human-controlled flight suffices to build a predictive model that is sufficiently accurate to simulate the vehicle.