UNIT V APPLICATIONS

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

1. LANGUAGE MODELS

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 **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 signaled.

1. Natural languages, such as English or Spanish, cannot be characterized as a definitive set of sentences.
Example: Everyone agrees that "Not to be invited is sad" is a sentence of English, but people disagree on the grammaticality of "To be not invited is sad." Therefore, it is more fruitful to define a natural language model as a probability distribution over sentences rather than a definitive set. That is, rather than asking if a string of *words* is or is not a member of the set defining the language, we instead ask for P(S =words)—what is the probability that a random sentence would be *words*. Natural languages are also **ambiguous**. "He saw her duck" can mean either that he saw a waterfowl belonging to her, or that he saw her move to evade something. Thus, again, we cannot speak of a single meaning for a sentence, but rather of a probability distribution over possible meanings.
2. Finally, natural languages are difficult to deal with because they are very large, and constantly changing. Thus, our language models are, at best, an approximation. We start with the simplest possible approximations and move up from there.

1.1.1 *N*-gram character models

1) An n-gram model is defined as a **Markov chain** of order n − 1. In Markov chain the probability of character ci depends only on the immediately preceding characters, not on any other characters. So in a trigram model (Markov chain of order 2) we have

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

We can define the probability of a sequence of characters $P(c_{1:N})$ under the trigram model by first factoring with the chain rule and then using the Markov assumption:

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

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*.

2) **language identification**: Given a text, determine what natural language it is written in. This is a relatively easy task; even with short texts such as "Hello, world" or "Wie geht es dir," it is easy to identify the first as English and the second as German. Computer systems identify languages with greater than 99% accuracy; occasionally, closely related languages, such as Swedish and Norwegian, are confused.
3) One approach to language identification is to first build a trigram character model of each candidate language, P (ci | ci−2: i−1, _), where the variable _ ranges over languages. For each _ the model is built by counting trigrams in a corpus of that language. (About 100,000 characters of each language are needed.) That gives us a model of **P** (Text | Language), but we want to select the most probable language given the text, so we apply Bayes' rule followed by the Markov assumption to get the most probable language:

$$\begin{aligned}\ell^* &= \operatorname*{argmax}_{\ell} P(\ell \mid c_{1:N}) \\ &= \operatorname*{argmax}_{\ell} P(\ell) P(c_{1:N} \mid \ell) \\ &= \operatorname*{argmax}_{\ell} P(\ell) \prod_{i=1}^{N} P(c_i \mid c_{i-2:i-1}, \ell)\end{aligned}$$

3) Other tasks for character models include spelling correction, genre classification, and named-entity recognition. 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).

4) **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.

1.1.2 Smoothing *n*-gram models

1. The major complication of n-gram models is that the training corpus provides only an estimate of the true probability distribution.
2. For common character sequences such as " th" any English corpus will give a good estimate: about 1.5% ofall trigrams
3. On the other hand, " ht" is very uncommon—no dictionary words start with ht. It is likely that the sequence would
4. have a count of zero in a training corpus of standard English. Does that mean we should assign P(" th")=0? If we did, then the text "The program issues an http request" would have an English probability of zero, which seems wrong
5. The process adjusting the probability of low-frequency counts is called **smoothing**.
6. 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. It defines the probability estimate as

   * $P(c_i|c_{i-2:i-1}) = \lambda_3 P(c_i|c_{i-2:i-1}) + \lambda_2 P(c_i|c_{i-1}) + \lambda_1 P(c_i)$, where $\lambda_3 + \lambda_2 + \lambda_1 = 1$.

It is also possible to have the values of λ_i depend on the counts: if we have a high count of trigrams, then we weigh them relatively more; if only a low count, then we put more weight on the bigram and unigram models

1.1.3 Model evaluation

1. The evaluation can be a task-specific metric, such as measuring accuracy on language identification.
2. Alternatively, we can have a task-independent model of language quality: calculate the probability assigned to the validation corpus by the model; the higher the probability 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

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

3. Perplexity can be thought of as the reciprocal of probability, normalized by sequence length.
4. It can also be thought of as the weighted average branching factor of a model. Suppose there are 100 characters in our language, and our model says they are all equally likely. Then for a sequence of any length, the perplexity will be 100. If some characters are more likely than others, and the model reflects that, then the model will have a perplexity less than 100.

1.1.4 *N*-gram word models

1. n-gram models over words rather than characters

2. All the same mechanism applies equally to word and character models.
3. The main difference is that the **vocabulary**— the set of symbols that make up the corpus and the model—is larger.
4. There are only about 100 characters in most languages, and sometimes we build character models that are even more restrictive, for example by treating "A" and "a" as the same symbol or by treating all punctuation as the same symbol. But with word models we have at least tens of thousands of symbols, and sometimes millions. The wide range is because it is not clear what constitutes a word.
5. Word n-gram models need to deal with **out of vocabulary** words.
6. 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.
7. This can be done by adding just one new word to the vocabulary: <UNK>, standing for the unknown word.
8. Sometimes multiple unknown-word symbols are used, for different classes. For example, any string of digits might be replaced with <NUM>, or any email address with <EMAIL>.

2.INFORMATION RETRIEVAL

DEFINITION: Information retrieval is the task of documents that are relevant to a user's need for information. The best-known examples of information retrieval systems are search engines on the WorldWideWeb. A Web user can type a query such as [AI book] into a search engine and see a list of relevant pages. In this section, we will see how such systems are built.
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.

The earliest IR systems worked on a **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.

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. Users find it unintuitive that when they want to know about farming in the states of Kansas *and* Nebraska they need to issue the query [farming (Kansas OR Nebraska)].
3. It can be hard to formulate an appropriate query, even for a skilled user. Suppose we try [information AND retrieval AND models AND optimization] and get an empty result set. We could try [information OR retrieval OR models OR optimization], but if that returns too many results, it is difficult to know what to try next.

2.1 IR scoring functions

1. Most IR systems have abandoned the Boolean model and use models based on the statistics of word counts.

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

5. The BM25 function takes all three of these into account

$$BM25(d_j, q_{1:N}) = \sum_{i=1}^{N} IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k+1)}{TF(q_i, d_j) + k \cdot (1 - b + b \cdot \frac{|d_j|}{L})} ,$$

where $|d_j|$ is the length of document d_j in words, and L is the average document length in the corpus: $L = \sum_i |d_i|/N$. We have two parameters, k and b, that can be tuned by cross-validation; typical values are $k = 2.0$ and $b = 0.75$. $IDF(q_i)$ is the inverse document

6. systems 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.

2.2 IR system evaluation

Imagine that an IR system has returned a result set for a single query, for which we know which documents are and are not relevant, out of a corpus of 100 documents. The document counts in each category are given in the following table

	In result set	Not in result set
Relevant	30	20
Not relevant	10	40

1. **Precision** measures the proportion of documents in the result set that are actually relevant. In our example, the precision is 30/(30 + 10)=.75. The false positive rate is 1 − .75=.25.
2. **Recall** measures the proportion of all the relevant documents in the collection that are in the result set. In our example, recall is 30/(30 + 20)=.60. The false negative rate is 1 −.60=.40.
3. In a very large document collection, such as theWorldWideWeb, recall is difficult to compute, because thereis no easy way to examine every page on the Web for relevance.

2.3 IR refinements

1. One common refinement is a better model of the effect of document length on relevance.
2. The BM25 scoring function uses a word model that treats all words as completely independent, but we know that some words are correlated: "couch" is closely related to both "couches" and "sofa." Many IR systems attempt to account for these correlations.
3. For example, if the query is [couch], it would be a shame to exclude from the result set those documents that mention "COUCH" or "couches" but not "couch." 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.

4. The next step is to recognize **synonyms**, such as "sofa" for "couch." As with stemming, this has the potential for small gains in recall, but can hurt precision. Synonyms and related words can be found in dictionaries or by looking for correlations in documents or in queries.
5. 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.

2.4 PAGERANK ALGORITHM

PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google:

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

```
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 ← Σ_i INLINK_i(p).HUB
      p.HUB ← Σ_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).

we will see that the recursion bottoms out properly. The PageRank for a page p is defined as:

$$PR(p) = \frac{1-d}{N} + d\sum_i \frac{PR(in_i)}{C(in_i)},$$

where $PR(p)$ is the PageRank of page p, N is the total number of pages in the corpus, in_i are the pages that link in to p, and $C(in_i)$ is the count of the total number of out-links on page in_i. The constant d is a damping factor. It can be understood through the **random surfer model**: imagine a Web surfer who starts at some random page and begins exploring.

2.5 The HITS algorithm

1. The Hyperlink-Induced Topic Search algorithm, also known as "Hubs and Authorities" or HITS, is another influential link-analysis algorithm.
2. HITS differ from PageRank in several ways. First, it is a query-dependent measure: it rates pages with respect to a query.
3. HITS first find a set of pages that are relevant to the query. It does that by intersecting hit lists of queries

4. 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.
5. 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.
6. 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.
7. Thus, as with PageRank, we iterate a process that updates the authority score of a page to be the sum of the hub scores of the pages that point to it, and the hub score to be the sum of the authority scores of the pages it points to.
8. Both PageRank and HITS played important roles in developing our understanding of Web information retrieval. These algorithms and their extensions are used in ranking billions of queries daily as search engines steadily develop better ways of extracting yet finer signals of search relevance.

3.IMAGE EXTRACTION

Information extraction is the process of acquiring INFORMATION 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. In a limited domain, this can be done with high accuracy.

3.1 Finite-state automata for information extraction

1. 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.
2. One step up from attribute-based extraction systems are **relational extraction** systems, which deal with multiple objects and the relations among them.
3. A typical relational-based extraction system is FASTUS, which handles news stories about corporate mergers and acquisitions.
4. A relational extraction system can be built as a series of **cascaded finite-state transducers**.
5. That is, the system consists of a series of small, efficient finite-state automata (FSAs), where each automaton receives text as input, transduces the text into a different format, and passes it along to the next automaton. FASTUS consists of five stages:
    1. Tokenization
    2. Complex-word handling
    3. Basic-group handling
    4. Complex-phrase handling
    5. Structure merging

6. FASTUS's first stage is **tokenization**, which segments the stream of characters into tokens (words, numbers, and punctuation). For English, tokenization can be fairly simple; just separating characters at white space or punctuation does a fairly good job. Some tokenizers also deal with markup languages such as HTML, SGML, and XML.

7. The second stage handles **complex words**, including collocations such as "set up" and "joint venture," as well as proper names such as "Bridgestone Sports Co." These are recognized by a combination of lexical entries and finite-state grammar rules.

8. The third stage handles **basic groups**, meaning noun groups and verb groups. The idea is to chunk these into units that will be managed by the later stages.

9. The fourth stage combines the basic groups into **complex phrases**. Again, the aim is to have rules that are finite-state and thus can be processed quickly, and that result in unambiguous (or nearly unambiguous) output phrases. One type of combination rule deals with domain-specific events.

10. The final stage **merges structures** that were built up in the previous step. If the next sentence says "The joint venture will start production in January," then this step will notice that there are two references to a joint venture, and that they should be merged into one. This is an instance of the **identity uncertainty problem.**

3.2 Probabilistic models for information extraction

1. The simplest probabilistic model for sequences with hidden state is the hidden Markov model, or HMM.
2. 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.

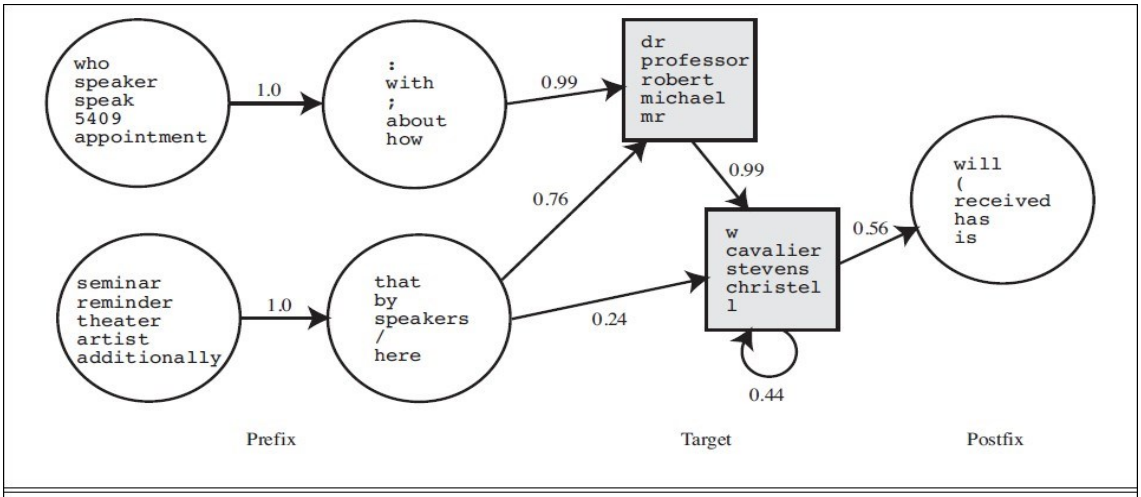

Figure 22.2 Hidden Markov model for the **speaker** of a talk announcement. The two square states are the target (note the second target state has a self-loop, so the target can match a string of any length), the four circles to the left are the prefix, and the one on the right is the postfix. For each state, only a few of the high-probability words are shown. From Freitag and McCallum (2000).

3. Once the HMMs have been learned, we can apply them to a text, using the Viterbi algorithm to find the most likely path through the HMM states. One approach is to apply each attribute HMM separately; in this case you would expect most of the HMMs to spend most of their time in background states. This is appropriate when the extraction is sparse— when the number of extracted words is small compared to the length of the text.
4. The other approach is to combine all the individual attributes into one big HMM, which would then find a path that wanders through different target attributes, first finding a speaker target, then a date target, etc. Separate HMMs are better when we expect just one of each attribute in a text and one big HMM is better when the texts are more free-form and dense with attributes.
5. **HMMs have the advantage of supplying probability numbers** that can help make the choice. If some targets are missing, we need to decide if this is an instance of the desired relation at all, or if the targets found are false positives. A machine learning algorithm can be trained to make this choice.

3.3 Ontology extraction from large corpora

1. A different application of extraction technology is building a large knowledge base or ontology of facts from a corpus. This is different in three ways:
   - First it is open-ended—we want to acquire facts about all types of domains, not just one specific domain.
   - Second, with a large corpus, this task is dominated by precision, not recall—just as with question answering on the Web
   - Third, the results can be statistical aggregates gathered from multiple sources, rather than being extracted from one specific text.
2. Here is one of the most productive templates:

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

3. Here the bold words and commas must appear literally in the text, but the parentheses are for grouping, the asterisk means repetition of zero or more, and the question mark means optional.
4. NP is a variable standing for a noun phrase
5. 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.
6. Similar templates can be constructed with the key words "including," "especially," and "or other." Of course these templates will fail to match many relevant passages, like "Rabies is a disease." That is intentional.
7. The "NP is a NP" template does indeed sometimes denote a subcategory relation, but it often means something else, as in "There is a God" or "She is a little tired." With a large corpus we can afford to be picky; to use only the high-precision templates.
8. 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.

3.4 Automated template construction

Clearly these are examples of the author–title relation, but the learning system had no knowledge of authors or titles. The words in these examples were used in a search over a Web corpus, resulting in 199 matches. 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.

1. Each template has the same seven components as a match.
2. The Author and Title are regexes consisting of any characters (but beginning and ending in letters) and constrained to have a length from half the minimum length of the examples to twice the maximum length.
3. The prefix, middle, and postfix are restricted to literal strings, not regexes.
4. The middle is the easiest to learn: each distinct middle string in the set of matches is a distinct candidate template. For each such candidate, the template's *Prefix* is then defined as the longest common suffix of all the prefixes in the matches, and the *Postfix* is defined as the longest common prefix of all the postfixes in the matches.
5. If either of these is of length zero, then the template is rejected.
6. The *URL* of the template is defined as the longest prefix of the URLs in 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.

3.5 Machine reading

1. 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**.
2. A representative machine-reading system is TEXTRUNNER. TEXTRUNNER uses cotraining to boost its performance, but it needs something to bootstrap from.
3. Because TEXTRUNNER 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.

TEXTRUNNER achieves a precision of 88% and recall of 45% (F1 of 60%) on a large Web corpus. TEXTRUNNER has extracted hundreds of millions of facts from a corpus of a half-billion Web pages.

4. NATURAL LANGUAGE PROCESSING

Machine translation, sometimes referred to by the abbreviation MT (not to be confused with computer-aided translation, machine-aided human translation (MAHT) or interactive translation) is a sub-field of computational linguistics that investigates the use of software to translate text or speech from one language to another.

On a basic level, MT performs simple substitution of words in one language for words in another, but that alone usually cannot produce a good translation of a text because recognition of whole phrases and their closest counterparts in the target language is needed. Solving this problem with corpus statistical, and neural techniques is a rapidly growing field that is leading to better translations, handling differences in linguistic typology, translation of idioms, and the isolation of anomalies.

4.1 Machine translation systems

All translation systems must model the source and target languages, but systems vary in the type of models they use. 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: creating a complete knowledge representation of everything; parsing into that representation; and generating sentences from that representation. Other systems are based on a **transfer model**.

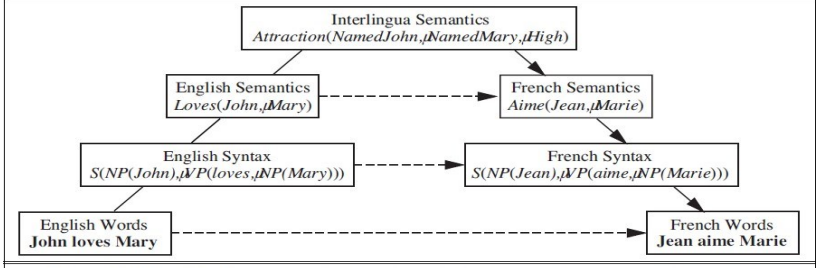

Figure 23.12 The Vauquois triangle: schematic diagram of the choices for a machine translation system (Vauquois, 1968). We start with English text at the top. An interlingua-based system follows the solid lines, parsing English first into a syntactic form, then into a semantic representation and an interlingua 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.

4.2 Statistical machine translation

1. **Find parallel texts**: First, gather a parallel bilingual corpus. For example, a **Hansard** is a record of parliamentary debate. Canada, Hong Kong, and other countries produce bilingual Hansards, the European Union publishes its official documents in 11 languages, and the United Nations publishes multilingual documents. Bilingual text is

also available online; some Web sites publish parallel content with parallel URLs, for example, /en/ for the English page and /fr/ for the corresponding French page. The leading statistical translation systems train on hundreds of millions of words of parallel text and billions of words of monolingual text.
2. **Segment into sentences**: The unit of translation is a sentence, so we will have to break the corpus into sentences. Periods are strong indicators of the end of a sentence, but consider "Dr. J. R. Smith of Rodeo Dr. paid $29.99 on 9.9.09."; only the final period ends a sentence. One way to decide if a period ends a sentence is to train a model that takes as features the surrounding words and their parts of speech. This approach achieves about 98% accuracy.
3. **Align sentences**: For each sentence in the English version, determine what sentence(s) it corresponds to in the French version. Usually, the next sentence of English corresponds to the next sentence of French in a 1:1 match, but sometimes there is variation: one sentence in one language will be split into a 2:1 match, or the order of two sentences will be swapped, resulting in a 2:2 match. By looking at the sentence lengths alone (i.e. short sentences should align with short sentences), it is possible to align them (1:1, 1:2, or 2:2, etc.) with accuracy in the 90% to 99% range using a variation on the Viterbi algorithm.
4. **Align phrases**: Within a sentence, phrases can be aligned by a process that is similar to that used for sentence alignment, but requiring iterative improvement. When we start, we have no way of knowing that "qui dort" aligns with "sleeping," but we can arrive at that alignment by a process of aggregation of evidence.
5. **Extract distortions**: Once we have an alignment of phrases, we can define distortion probabilities. Simply count how often distortion occurs in the corpus for each distance.
6. **Improve estimates with EM**: Use expectation–maximization to improve the estimates of P(f | e) and P(d) values. We compute the best alignments with the current values of these parameters in the E step, then update the estimates in the M step and iterate the process until convergence.

4.3 Speech recognition

Definition: Speech recognition is the task of identifying a sequence of SPEECH words uttered by a speaker, given the acoustic signal. It has become one of the mainstream applications of AI.

1. **Example:** The phrase "recognize speech" sounds almost the same as "wreck a nice beach" when spoken quickly. Even this short example shows several of the issues that make speech problematic.
2. First, **segmentation**: written words in English have spaces between them, but in fast speech there are no pauses in "wreck a nice" that would distinguish it as a multiword phrase as opposed to the single word "recognize."
3. Second, **coarticulation**: when speaking quickly the "s" sound at the end of "nice" merges with the "b" sound at the beginning of "beach," yielding something that is close to a "sp."Another problem that does not show up in this example is **homophones**—words like "to, "too," and "two" that sound the same but differ in meaning.

$$\underset{word_{1:t}}{\operatorname{argmax}} P(word_{1:t} \mid sound_{1:t}) = \underset{word_{1:t}}{\operatorname{argmax}} P(sound_{1:t} \mid word_{1:t}) P(word_{1:t}) \ .$$

Here $P(sound_{1:t}|word_{1:t})$ is the **acoustic model**. It describes the sounds of words—that "ceiling" begins with a soft "c" and sounds the same as "sealing." $P(word_{1:t})$ is known as the **language model**. It specifies the prior probability of each utterance—for example, that "ceiling fan" is about 500 times more likely as a word sequence than "sealing fan."

4. Once we define the acoustic and language models, we can solve for the most likely sequence of words using the Viterbi algorithm.

4.3.1 Acoustic model

1. An analog-to-digital converter measures the size of the current—which approximates the amplitude of the sound wave at discrete intervals called the **sampling rate.**
2. The precision of each measurement is determined by the **quantization factor**; 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.
3. A **phoneme** is the smallest unit of sound that has a distinct meaning to speakers of a particular language.

For example, the “t” in “stick” sounds similar enough to the “t” in “tick” that speakers of English consider them the same phoneme.
Each frame is summarized by a vector of **features. Below Picture represents phone model**

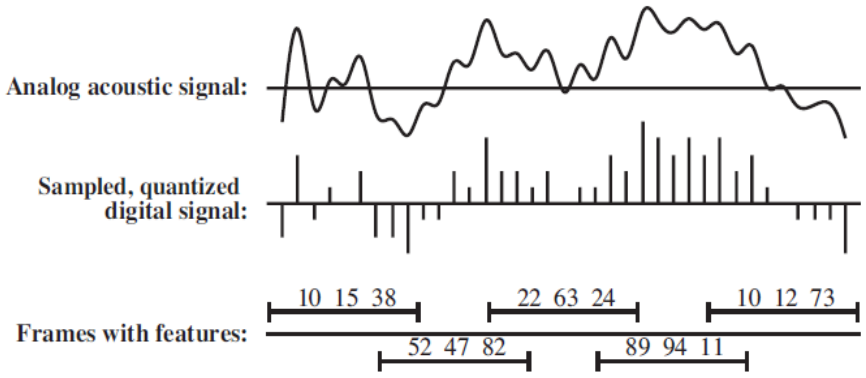

Figure 23.15 Translating the acoustic signal into a sequence of frames. In this diagram each frame is described by the discretized values of three acoustic features; a real system would have dozens of features.

4.3.2 Building a speech recognizer

1. The quality of a speech recognition system depends on the quality of all of its components— the language model, the word-pronunciation models, the phone models, and the signal processing algorithms used to extract spectral features from the acoustic signal.
2. The systems with the highest accuracy work by training a different model for each speaker, thereby capturing differences in dialect as well as male/female and other variations. This training can require several hours of interaction with the speaker, so the systems with the most widespread adoption do not create speaker-specific models.
3. The accuracy of a system depends on a number of factors. First, the quality of the signal matters: a high-quality directional microphone aimed at a stationary mouth in a padded room will do much better than a cheap microphone transmitting a signal over phone lines from a car in traffic with the radio playing. The vocabulary size matters: when recognizing digit strings with a vocabulary of 11 words (1-9 plus “oh” and “zero”), the word error rate will be below 0.5%, whereas it rises to about 10% on news stories with a 20,000-word vocabulary, and 20% on a corpus with a 64,000-word vocabulary. The task matters too: when the system is trying to accomplish a specific task—book a flight or give directions to a restaurant—the task can often be accomplished perfectly even with a word error rate of 10% or more.

5. ROBOT

1. **Robots** are physical agents that perform tasks by manipulating the physical world.
2. To do so, they are equipped with **effectors** such as legs, wheels, joints, and grippers.
3. Effectors have a single purpose: to assert physical forces on the environment.
4. Robots are also equipped with **sensors**, which allow them to perceivetheir environment.
5. 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.
6. Most of today’s robots fall into one of three primary categories. **Manipulators**, or robot arms are physically anchored to their workplace, for example in a factory assembly line or on the International Space Station.

5.1 Robot hardware

1. Sensors are the perceptual interface between robot and environment.
2. **Passive sensors**, such as cameras, are true observers of the environment: they capture signals that are generated by other sources in the environment.

3. **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, but at the expense of increased power consumption and with a danger of interference when multiple active sensors are used at the same time. Whether active or passive, sensors can be divided into three types, depending on whether they sense the environment, the robot's location, or the robot's internal configuration.
4. **Range finders** are sensors that measure the distance to nearby objects. In the early days of robotics, robots were commonly equipped with **sonar sensors**. Sonar sensors emit directional sound waves, which are reflected by objects, with some of the sound making it.
5. **Stereo vision** 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.
6. Other range sensors use laser beams and special 1-pixel cameras that can be directed using complex arrangements of mirrors or rotating elements. These sensors are called **scanning lidars** (short for *light detection and ranging*).
7. Other common range sensors include radar, which is often the sensor of choice for UAVs. Radar sensors can measure distances of multiple kilometers. On the other extreme 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.
8. A second important class of sensors is **location sensors**. 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.
9. The third important class is **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.
10. **Inertial sensors**, such as gyroscopes, rely on the resistance of mass to the change of velocity. They can help reduce uncertainty.
11. 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.

5.2 Robotic perception

1. 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. In other words, robots have all the problems of **state estimation** (or **filtering)**
2. As a rule of thumb, good internal representations for robots have three properties: they contain enough information for the robot to make good decisions, they are structured so that they can be updated efficiently, and they are natural in the sense that internal variables correspond to natural state variables in the physical world.

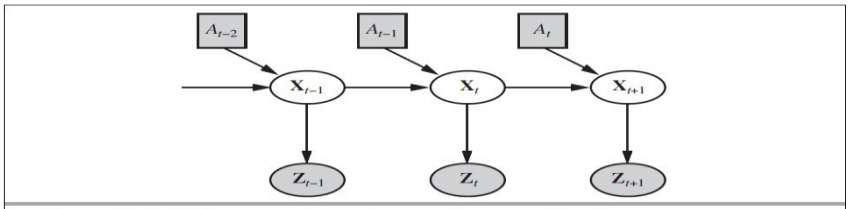

Figure 25.7 Robot perception can be viewed as temporal inference from sequences of actions and measurements, as illustrated by this dynamic Bayes network.

We would like to compute the new belief state, $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{z}_{1:t+1}, a_{1:t})$, from the current belief state $\mathbf{P}(\mathbf{X}_t \mid \mathbf{z}_{1:t}, a_{1:t-1})$ and the new observation $\mathbf{z}_{t+1}$. We did this in Section 15.2, but here there are two differences: we condition explicitly on the actions as well as the observations, and we deal with *continuous* rather than *discrete* variables. Thus, we modify the recursive filtering equation (15.5 on page 572) to use integration rather than summation:

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{z}_{1:t+1}, a_{1:t}) = \alpha \mathbf{P}(\mathbf{z}_{t+1} \mid \mathbf{X}_{t+1}) \int \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, a_t)\, P(\mathbf{x}_t \mid \mathbf{z}_{1:t}, a_{1:t-1})\, d\mathbf{x}_t \,. \quad (25.1)$$

MONTE CARLO ALGORITHM

```
function MONTE-CARLO-LOCALIZATION(a, z, N, P(X'|X, v, ω), P(z|z*), m) returns
a set of samples for the next time step
  inputs: a, robot velocities v and ω
          z, range scan z_1, ..., z_M
          P(X'|X, v, ω), motion model
          P(z|z*), range sensor noise model
          m, 2D map of the environment
  persistent: S, a vector of samples of size N
  local variables: W, a vector of weights of size N
                   S', a temporary vector of particles of size N
                   W', a vector of weights of size N

  if S is empty then        /* initialization phase */
     for i = 1 to N do
        S[i] ← sample from P(X_0)
     for i = 1 to N do      /* update cycle */
        S'[i] ← sample from P(X'|X = S[i], v, ω)
        W'[i] ← 1
        for j = 1 to M do
           z* ← RAYCAST(j, X = S'[i], m)
           W'[i] ← W'[i] · P(z_j | z*)
     S ← WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S', W')
  return S
```

Figure 25.9 A Monte Carlo localization algorithm using a range-scan sensor model with independent noise.

1. 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**.
2. SLAM problems are solved using many different probabilistic techniques, including the extended Kalman filter discussed above.

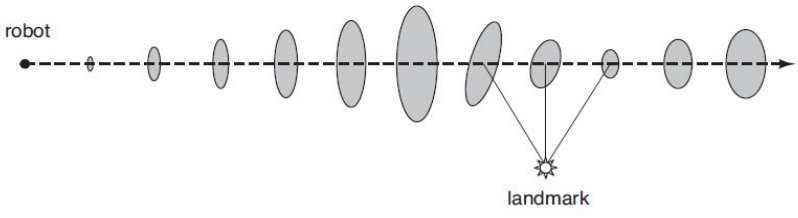

Figure 25.12 Example of localization using the extended Kalman filter. The robot moves on a straight line. As it progresses, its uncertainty increases gradually, as illustrated by the error ellipses. When it observes a landmark with known position, the uncertainty is reduced.

5.3 Machine learning in robot perception

1. Machine learning plays an important role in robot perception. This is particularly the case when the best internal representation is not known. One common approach is to map high dimensional sensor streams into lower-dimensional spaces using unsupervised machine learning methods Such an approach is called **low-dimensional embedding.**

2. Another machine learning technique enables robots to continuously adapt to broad changes in sensor measurements.
3. Adaptive perception techniques enable robots to adjust to such changes.Methods that make robots collect their own training data (with labels!) are called **self-supervised**. In this instance, the robot machine learning to leverage a short-range sensor that works well for terrain classification into a sensor that can see much farther.

5.4 Planning to move

1. All of a robot's deliberations ultimately come down to deciding how to move effectors.
2. The **point-to-point motion** problem is to deliver the robot or its end effector to a designated target location.
3. A greater challenge is the **compliant motion** problem, in which a robot moves while being in physical contact with an obstacle.
4. An example of compliant motion is a robot manipulator that screws in a light bulb, or a robot that pushes a box across a table top. 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 3D space.
5. The **path planning** problem is to find a path from one configuration to another in configuration space. We have already encountered various versions of the path-planning problem throughout this book; the complication added by robotics is that path planning involves *continuous* spaces. T
6. here are two main approaches: **cell decomposition** and **skeletonization**. Each reduces the continuous path-planning problem to a discrete graph-search problem. In this section, we assume that motion is deterministic and that localization of the robot is exact. Subsequent sections will relax these assumptions.
7. 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.

5.4.1 Configuration space

1. It has two joints that move independently. Moving the joints alters the (x, y) coordinates of the elbow and the gripper. (The arm cannot move in the z direction.) This suggests that the robot's configuration can be described by a four-dimensional coordinate: (xe, ye) for the location of the elbow relative to the environment and (xg, yg) for the location of the gripper. Clearly, these four coordinates characterize the full state of the robot. They constitute what is known as **workspace representation.**
2. configuration spaces have their own problems. The task of a robot is usually expressed in workspace coordinates, not in configuration space coordinates. This raises the question of how to map between workspace coordinates and configuration space.
3. These transformations are linear for prismatic joints and trigonometric for revolute joints. This chain of coordinate transformation is known as **kinematics**.
4. The inverse problem of calculating the configuration of a robot whose effector location is specified in workspace coordinates is known as **inverse kinematics.**

The configuration space can be decomposed into two subspaces: the space of all configurations that a robot may attain, commonly called **free space**, and the space of unattainable configurations, called **occupied space.**

5.4.2 Cell decomposition methods

1. The simplest cell decomposition consists of a regularly spaced grid.
2. Grayscale shading indicates the *value* of each free-space grid cell—i.e., the cost of the shortest path from that cell to the goal.
3. Cell decomposition methods can be improved in a number of ways, to alleviate some of these problems. The first approach allows *further subdivision* of the mixed cells—perhaps using cells of half the original size. This can be continued recursively until a path is found that lies entirely within free cells. (Of course, the method only works if there is a way to decide if a given cell is a mixed cell, which is easy only if the configuration space boundaries
4. have relatively simple mathematical descriptions.) This method is complete provided there is a bound on the smallest passageway through which a solution must pass. One HYBRID A* algorithm that implements this is **hybrid A***.

5.4.3 Modified cost functions

1. A potential field is a function defined over state space, whose value grows with the distance to the closest obstacle.
2. The potential field can be used as an additional cost term in the shortest-path calculation.
3. This induces an interesting tradeoff. On the one hand, the robot seeks to minimize path length to the goal. On the other hand, it tries to stay away from obstacles by virtue of minimizing the potential function.
4. There exist many other ways to modify the cost function. For example, it may be desirable to *smooth* the control parameters over time.

5.4.4 Skeletonization methods

1. The second major family of path-planning algorithms is based on the idea of **skeletonization**.
2. These algorithms reduce the robot's free space to a one-dimensional representation, for which the planning problem is easier.
3. This lower-dimensional representation is called a **skeleton** of the configuration space.
4. 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.
5. 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.

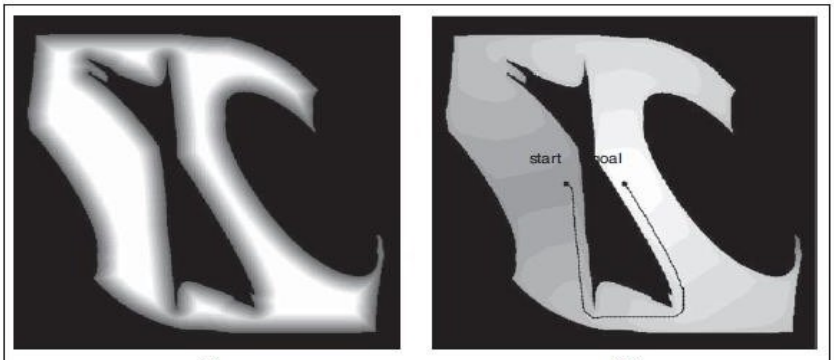

Figure 25.17 (a) A repelling potential field pushes the robot away from obstacles. (b) Path found by simultaneously minimizing path length and the potential.

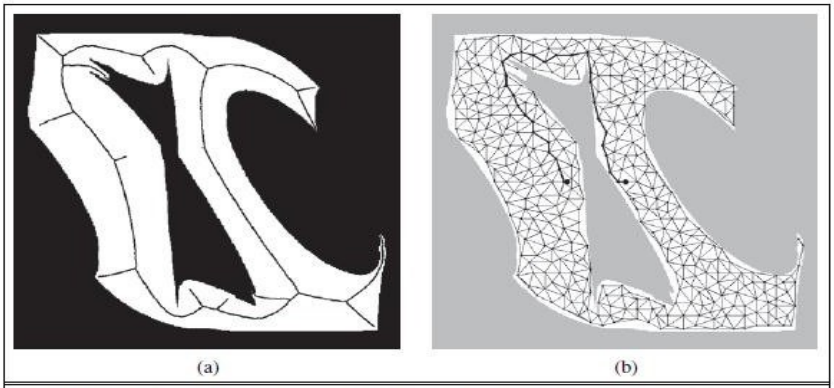

Figure 25.18 (a) The Voronoi graph is the set of points equidistant to two or more obstacles in configuration space. (b) A probabilistic roadmap, composed of 400 randomly chosen points in free space.

5.4.5 Robust methods

1. 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.
2. A robust solution is one that works no matter what actual values occur, provided they are within the assumed interval.
3. An extreme form of robust method is the **conformant planning** approach.

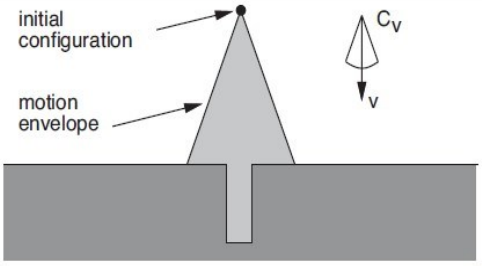

Figure 25.19 A two-dimensional environment, velocity uncertainty cone, and envelope of possible robot motions. The intended velocity is v, but with uncertainty the actual velocity could be anywhere in C_v, resulting in a final configuration somewhere in the motion envelope, which means we wouldn't know if we hit the hole or not.

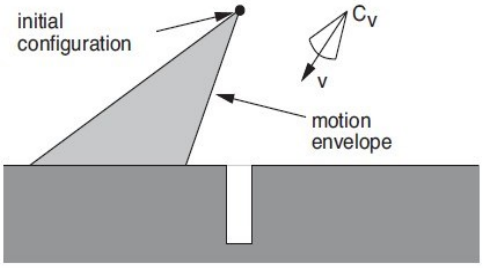

Figure 25.20 The first motion command and the resulting envelope of possible robot motions. No matter what the error, we know the final configuration will be to the left of the hole.