Natural Language for Communication

Michael Rose

In which we see how humans communicate with one another in natural language, and how computer agents might join in on the conversation.

This chapter looks at language models for communication. We start with grammatical models of the phrase structure of sentences, add semantics to the model, and then apply it to machine translation and speech recognition.

Phrase Structure Grammars

A big issue for n-gram models of language is **data sparsity**. For example, with 10^5 words, we get 10^{15} trigram probabilities to estimate. We can adress the problem of sparsity through generalization.

Generative Capacity

Grammatical formalisms can be classified by their **generative capacity**; the set of languages they can represent. Chomsky decscribes four classes of grammatical formalisms that differ only in the form of the rewrite rules. Here is the hierarchy, most powerful class first:

- **Recursively enumerable** grammars use unrestricted rules. Both sides can use any number of terminal and nonterminal symbols (for example $ABC \rightarrow DE$). These grammars are equivalent to Turing machines in their expressive power.
- **Context Sensitive Grammars** - These are restricted only in that the right side must contain at least as many symbols as the left hand side. For example, $ABC \rightarrow AYB$, or $a^n b^n c^n$.
- **Context Free Grammars** - The left hand consists of a single nonterminal symbol. Thus each rule licenses rewriting the nonterminal as the right hand side in any context.
- **Regular Grammars** - The most restricted class. Every rule has a single nonterminal on the left hand side and a terminal symbol optionally followed by a nonterminal on the right hand side. Regular grammars are equivalent in power to finite state machines.

The grammars higher in the hierarchy have more expressive power, but the algorithms for dealing with them are less efficient.

A popular model of phrase structure is the **probabilistic context free grammar**, or PCFG. A **grammar** is a collection of rules that defines a language as a set of allowable strings of words.

Here is a PFCG rule:

$$\begin{aligned} VP &\rightarrow Verb \quad [0.70] \\ &\mid \quad VP \quad NP \quad [0.30] \end{aligned}$$

Here VP (verb phrase) and NP (noun phrase) are **nonterminal symbols**. The grammar also refers to actual words, which are called **terminal symbols**.

Syntactical Analysis (Parsing)

Parsing is the process of analyzing a string of words to uncover its phrase structure, according to the rules of a grammar. In order to avoid inefficiency through excessive backtracking in our algorithms, we use dynamic programming. Every time we analyze a substring, store the results so we won't have to reanalyze it later. We store our built up lexicon of phrases in a chart, and algorithms that deal with this are called **chart parsers**. An example of a bottom up parser is the **CYK algorithm**.

The CYK algorithm requires that the grammar is in **Chomsky Normal Form**. This requires that the grammar be in one of two formats: lexical rules of the form $X \rightarrow word$, and syntactic rules of the form $X \rightarrow YZ$. Any context free grammar can be automatically transformed into Chomsky normal form.

The CYK algorithm uses space of $O(n^2m)$ for the P table, where n is the number of words in the sentence, and m is the number of nonterminal symbols in the grammar. It has a time complexity of $O(n^3m)$. No algorithm can do better for general context free grammars, although there are faster algorithms for more restricted grammars. The algorithm manages to process the 2^c parse trees in $O(c^3)$ time because it doesn't need to examine all the parse trees, it just needs to find the most probable tree.

In practice we are often only after just the best or a few trees. In this case, we can think of the CYK algorithm as defining the complete state space defined by the "apply grammar rule" operator. It is possible to search just part of this space using the A* search. With the A* algorithm we don't have to search the entire state space, and we are guaranteed that the first parse found will be the most probable.

23.2.1 | Learning Probabilities for PCGFs

A PFCG has many rules, with a probability for each rule. This suggests that learning the grammar from datda might be better than a knowledge engineering approach. Learning is easiest if we are given a corpus of correctly parsed sentences, commonly called a **treebank**. Given a corpus of trees, we can create a PCFG just by counting and smoothing.

If a treebank is not available, but we have a corpus of raw, unlabeled sentences then we can still learn grammar from the corpus but it is more difficult. We then have two problems : learning the structure of the grammar rules, and learning the probabilities associated with each rule.

We'll assume that we're given the lexical and syntactic category names. Then we can assume that every grammar includes every possible $X \rightarrow YZ$ or $X \rightarrow word$ rule. We can then use an expectation maximization approach just as we did with Hidden Markov Models. The parameters we are trying to learn are the rule probabilities; we start off with random or uniform values. The hidden variables are the parse trees: we don't know whether a string of words $w_i, ..., w_j$ is or is not generated by a rule $X \rightarrow ....$ The E step estimates the probability that each subsequence is generated by each rule. The M step then estimates the probability of each rule. The whole computation can be done in a dynamic programming fashion with an algorithm called the **inside out algorithm** in analogy to the forward-backward algorithm for HMMs.

The insideout algorithm has some drawbacks:

1. The parses that are assigned by the induced grammars are often difficult to understand and unsatisfying to linguists. This makes it hard to combine handcrafted knowledge with automated induction.
2. It is slow: $O(n^3m^3)$, where n is the number of words in a sentence and m is the number of grammar categories.
3. The space of probability assignments is very large, and empirically it seems that getting stuck in local maxima is a severe problem.

Alternatives such as simulated annealing get closer to global maximums at the cost of more computation. Currently the error rates for automatically learned grammars are still about 50% higher than for hand constructed grammar, but the gap is decreasing.

23.3 | Augmented Grammars and Semantic Interpretation

In this section we see how to extend context-free grammars – to say that, for example, not every noun phrase is independent of context, but rather, certain noun phrases are more likely to appear in one context, and others in another context.

23.3.1 | Lexicalized PCFGs

To get at the relationships between words in a noun phrase, we can use a **lexicalized PCFG**, in which the probabilities for a rule depend on the relationship between words in the parse tree, not just on the adjacency of words in a sentence. For this, it is useful to introduce the notion of the head of a phrase - the most important word. We use the notation $VP(v)$ to denote a phrase with category VP whose head word is v. We say that the category VP is augmented with the head variable v.

23.3.2 | Formal Definition of Augmented Grammar Rules

Augmented rules can be formalized by showing how an augmented rule can be translated into a logical sentence. These sentences have the form of a definite clause, so the result is called a **definite clause grammar**. As an example, here is a version of a rule from the lexicalized grammar for NP:

$NP(n) \rightarrow Article(a)Adjs(j)Noun(n)\{constraint\}$

The translation from grammar rule to definite clause allows us to talk about parsing as logical inference. Then we can do bottom up parsing using forward chaining or top down parsing using backward chaining.

23.3.3 | Case Agreement and Subject Verb Agreement

In order to deal with noun phrases that have different meanings in a subjective and objective case, we can split NP into two categories, NP_S and NP_o, to stand for the noin phrases in the subjective and objective case, respectively. Similarly, we need to deal with cases regarding subject. The context of a sentence changes for third person singular subjects and all other combinations of person and number.

Complications

Disambiguation is the process of recovering the most probable intended meaning of an utterance. To do this properly, we need to combine four models:

1. The world model: the likelihood that a proposition occurs in the world.
2. The mental model: the likelihood that the speaker forms the intention of communicating a certain fact to the hearer.
3. The language model: the likelihood 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: for spoken communication, the likelihood that a particular sequence of sounds will be generated, given that the speaker has chosen a given string of words.

23.4 | Machine Translation

All translation systems must model the source and target languages, but systems vary in the type of models they use.

Some systems attempt to analyzw 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**. They keep a database of translation rules and whenever the rule matches, they translate it directly.

23.4.2 | Statistical Machine Translation

Most successful machine translation systems are built by training a probabilistic model gathered from a large corpus of text. For example, if we wished to translate a sentence in English (e) into French (f)m we find the string of words that maximizes:

$$f^* = \arg\max_f P(f|e) = \arg\max P(e|f)P(f)$$

where $P(f)$ is the target language model for French, $P(e|f)$ is the translation model, which shows how probable an english sentence is as a translation for a given french sentence, and $P(f|e)$ vice versa.

The procedure is as follows:

1. Find parallel texts
2. Segment into sentences
3. Align sentences
4. Align phrases
5. Extract distortions - where $P(f, d|e)$ is the probability that the sequence of phrases f with distortions d is a translation of the sequence of phrases e -
$$P(f, d|e) = \prod_i P(f_i|e_i)P(d_i)$$
6. Improve estimates with EM

23.5 | Speech Recognition

Speech Recognition is the task of identifying a sequence of words uttered by a speaker, given the acoustic signal. Some of the difficulties involved in speech recognition are:

1. segmentation - written words in english have spaces between them, but this is not the case when dealing with fast speech in audio form.
2. coarticulation - when speaking quickly, the ends and beginnings of words tend to meld into each other.
3. homophones - there exist words like to, too, two, which all sound alike but have different meanings.

We can view speech recognition as a problem in most likely sequence explanation. This is the problem of finding the most likely sequence of state variables, given a sequence of observations. In this case, the state variables are the words, and the observations are sounds. More precisely, an observation is a vector of features extracted from the audio signal. We can compute this as such:

$$\arg\max_{word_{1:t}} P(word_{1:t}|sound_{1:t}) = \arg\max_{word_{1:t}} P(sound_{1:t}|word_{1:t})P(word_{1:t})$$

where $P(sound_{1:t}|word_{1:t})$ is the **acoustic model**. This approach is called the **noisy channel model** by Claude Shannon. Once we define the acoustic and language models, we can solve for the most likely sequence of words using the Viterbi algorithm.

23.5.1 | Acoustic Model

Each input frame is defined by a vector of features. A Fourier transform is used to determine the amount of acoustic energy at about a dozen frequencies. Then we compute a measure called the **mel frequency cepstral coefficient** for each frequency. We also compute the total energy for the frame, giving us 13 features. For each one we compute the difference between this frame and the previous frame, and the difference between differences, for a total of 39 frames.

Then we can take our features and discretize them for use in a hidden Markov model. Next we need to describe the unobservable states of the HMM and define the transition model and the sensor model. Then we can finally string together our HMMs and form a **pronunciation model**.

23.5.2 | Language Model

For general purpose speech recognition, the language model can be an n-gram model of text learned from a corpus of written sentences. However, spoken language has different characteristics than written language, so it is better to have a corpus of transcripts of spoken language. Part of the design of a voice user interface is to coerce the user into saying things from a limited set of options, so the speech recognizer will have a tighter probability distribution to deal with.

20.6 | Summary

- Formal language theory and phrase structure grammars (in particular context free grammar) are useful tools for dealing with some aspects of natural language. The probabilistic context free grammar formalism is widely used.
- Sentences in a context free language can be parsed in $O(n^3)$ time by a chart parser such as the CYK algorithm, which requires grammar rules to be in Chomsky normal form.
- A treebank can be used to learn grammar. It is also possible to learn a grammar from an unparsd corpus of sentences, but this is less successful.
- A lexicalized PCFG allows us to represent that some relationships between words are more common than others.
- It is conventient to augment a grammar to handle such problems as subject verb agreement and pronoun case. Definite clause grammar is a formalism that allows for augmentations. With DCG, parsing and semantic interpretation (and even generation) can be done using logical inference.
- Semantic interpretation can also be handled by augmented grammar.
- Ambiguity is a very important problem in natural language understanding; most sentences have many possible interpretations, but usually only one is appropriate. Disambiguation relies on knowledge about the world, about the current situation, and about language use.
- Machine translation systems have been implemented using a range of techniques, from full syntactic and semantic analysis to statistical techniques based on phrase frequencies. Currently the statistical models are most popular and most sucessful.
- Speech recognition systems are also primarily based on statistical principles. Speech systems are popular and useful, albeit imperfect.
- Together, machine translation and speech recognition are two of the big successes of natural language technology. One reason that the models perform well is that large corpora are available - both translation and speech tasks that are performed in the wild by people every day. In contrast, tasks like parsing sentences have been less successful, in part because no large corpora of parsed sentences are available in the wild and in part because parsing is not useful in and of itself.