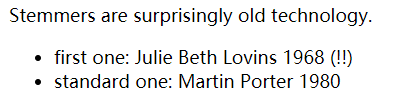
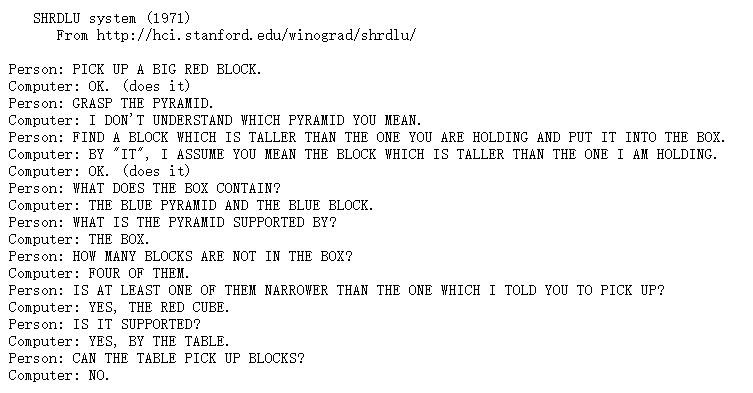
**Midterm 2**

Julie Lovins



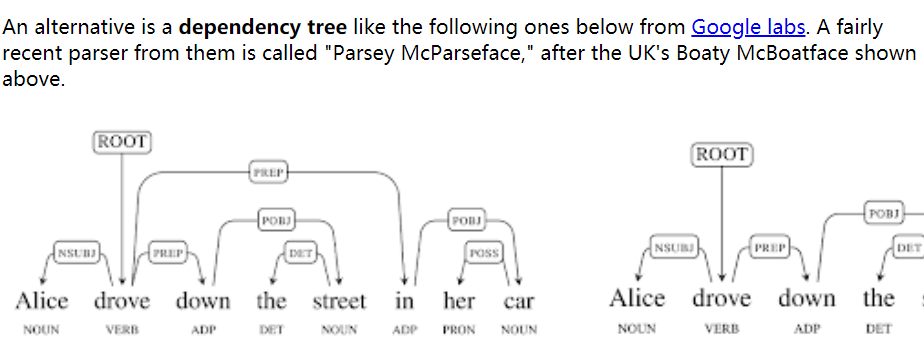
SHRDLU



Pattern Playback

Very early speech synthesis (1940's, Haskins Lab) was done by painting pictures of formants onto a sheet of plastic and using a physical transducer called the Pattern Playback to convert them into sounds.

Parsey McParseface



**Modelling text data**

* Word types vs. word tokens

word type: a dictionary entry such as "cat"

word token: a word in a specific position in the text

* The Bag of Words model

A bag of words model determines the class of a document based on how frequently each individual word occurs. **It ignores the order in which words occur, which ones occur together, etc.** **So it will miss some details,** **e.g. the difference between "poisonous" and "non-poisonous,"**

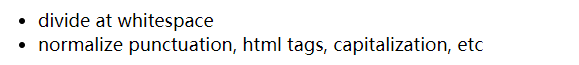
* Bigrams, ngrams

The Bag of Words model from last class used individual words as features, ignoring the order in which they appeared. Text classification is often improved by using pairs of words (bigrams) as features. Some applications (notably speech recognition) use trigrams (triples of words) or even longer n-grams. E.g. "the big cat" conveys more topic information than just "big."

**For n words, there are n2 possible bigrams. But our amount of training data stays the same. So the training data is even less adequate for bigrams than it was for single words.**

* Tokenization

Our first job is to produce a clean string of words, suitable for use in the classifier.



**Format features such as punctuation and capitalization may be helpful or harmful, depending on the task.**

So "normalize" may either involve throwing away the information or converting it into a format that's easier to process.

* stemming (Porter stemmer)

Stemming converts inflected forms of a word to a single standardized form. It removes prefixes and suffixes, leaving only the content part of the word (its "stem")

i.e. help, helping, helpful -> help

* making units of useful size: dividing words or grouping characters

Chinese: no spaces between charactesr. So we run a "word segmenter" that groups strings of 1-3 related characters into words.

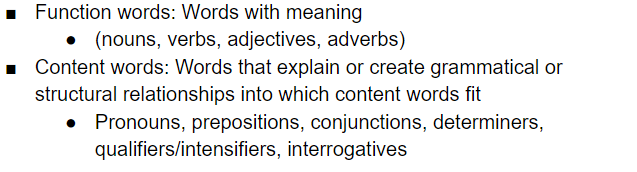
German: complex words (esp. compounds) would be divided. E.g. "Wintermantel" might be divided into "winter" (winter) and "mantel" (coat).

* stop words & rare words

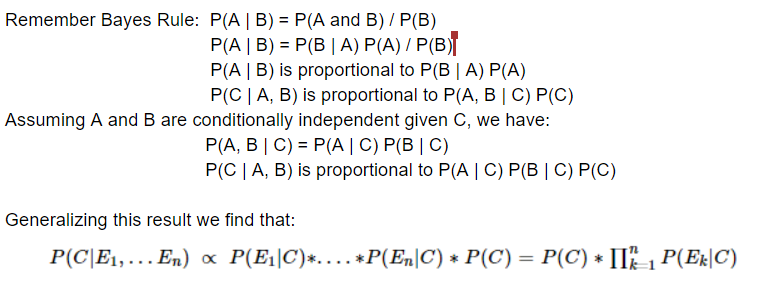
Very frequent words ("stop words") often convey very little information about the topic. So they are typically deleted before doing a bag of words analysis.

Rare words may be deleted. More often, all rare words are mapped into a single placeholder value (e.g. **UNK**). This allows us to treat them all as a single item, with a reasonable probability of being observed.

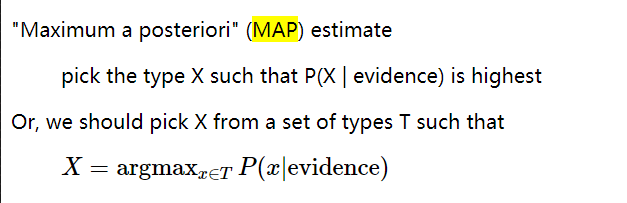
* Filler: Delete them as they convey no information
* Backchannel: occurs when one participant is speaking and another participant interjects responses to the speaker (“hmm”, “uh-huh”, “yeah”)
* function vs. content

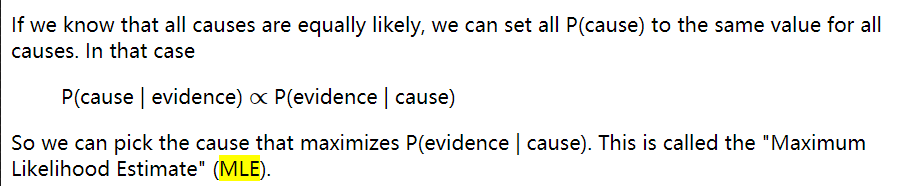


**Naive Bayes**



* MAP and MLE versions of the estimation equations





**The MLE estimate can be very inaccurate if the probabilities of different causes are actually different and not the same value as was assumed. On the other hand, it can be a sensible choice if we have no information about the prior probabilities or if there is reason the believe P(Cause) should be constant across all possible causes.**

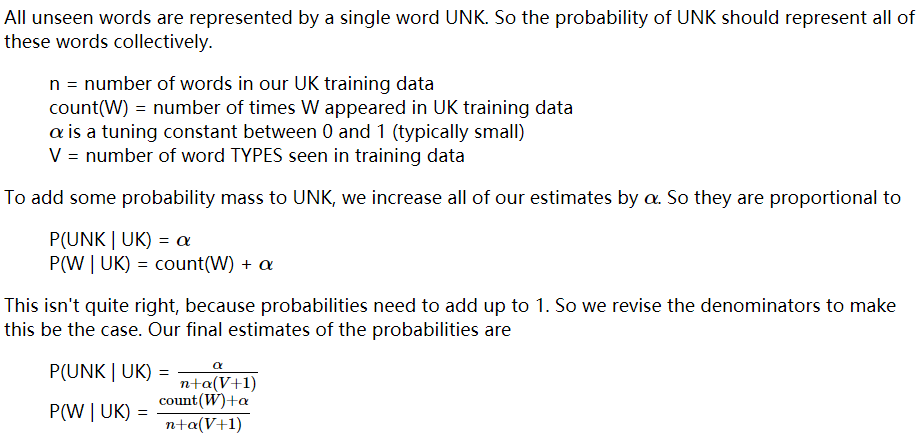
* Estimating probabilities from data
* Avoiding underflow (log transforms)

Use log

* Avoiding overfitting

for uncommon words: Words that didn't appear in the training data get estimated zero probability. Words that were uncommon in the training data get inaccurate estimates.

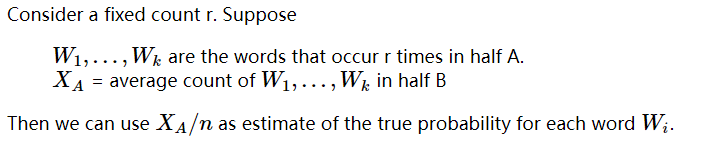
* Smoothing



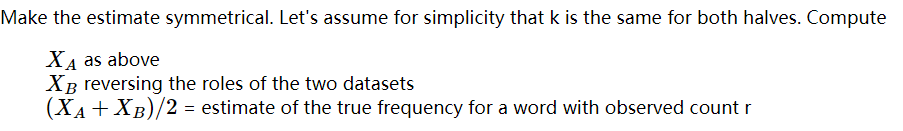
**overestimates probability of unseen words**

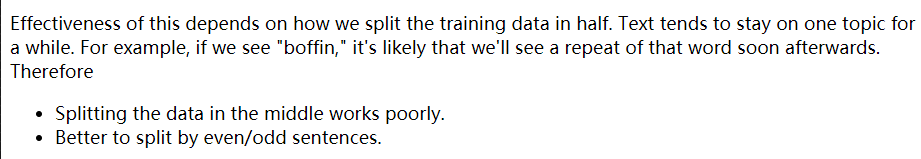
**underestimates probability of common words**

* Deleted estimation: **Also called cross-validation or held-out estimation.**



**Tweaks on deleted estimation:**





***Performance of Deleted Estimation:***

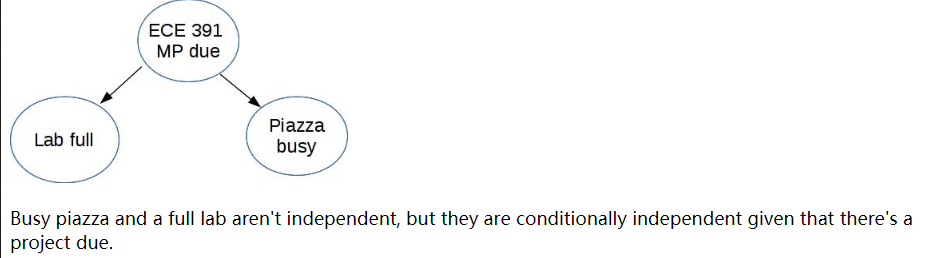
**Infrequent words overestimate their true probability. Estimates for common words are usually accurate.**

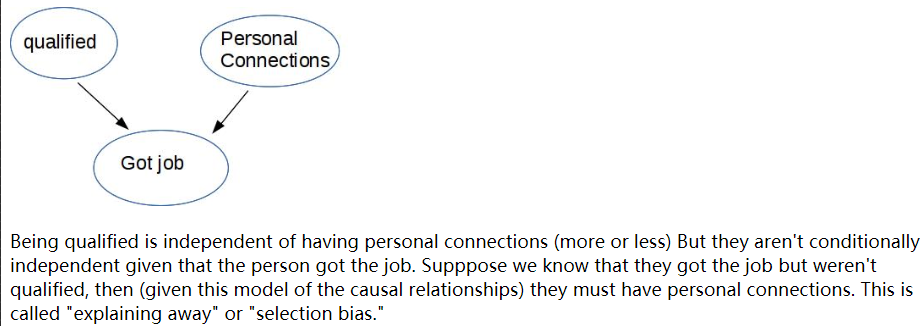
* N-gram smoothing

Idea 1: If we haven't seen an ngram, **guess its probability from the probabilites of its prefix (e.g. "the angry") and the last word ("armadillo").**

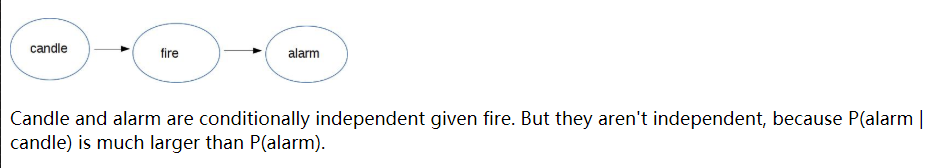
Idea 2: **Guess that an unseen word is more likely in contexts where we've seen many different words. I.e. some contexts are much more predictable (“hand me the …” can be followed by anything but “I went to mcdonalds and bought a …” is usually followed by “hamburger”.**

**Bayes nets**



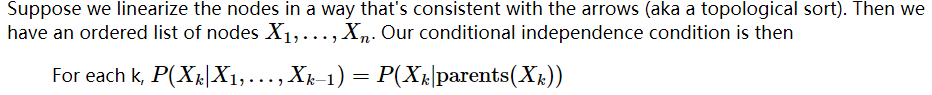


**A high-level point: conditional independence isn't some weakened variant of independence. Neither property implies the other.**

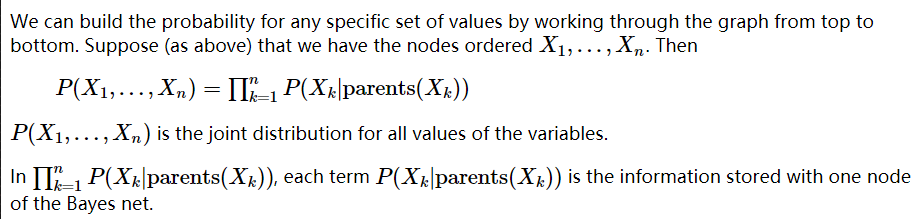


**A Bayes net is a directed acyclic (no cycles) graph (called a DAG for short). So nodes are partially ordered by the ancestor/descendent relationships.**

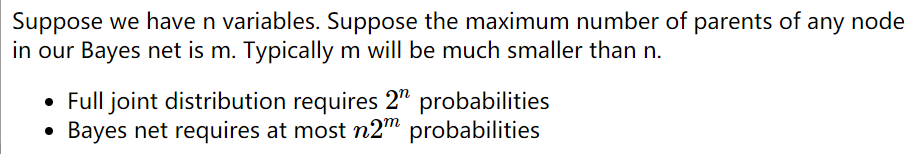
**Each node is conditionally independent of its ancestors, given its parents. That is, the ancestors can ONLY influence node A by working via parents(A)**

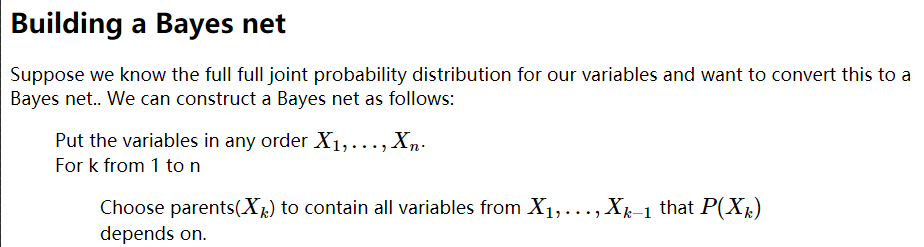


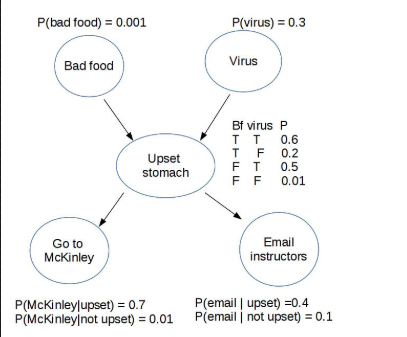
* Reconstructing the joint distribution



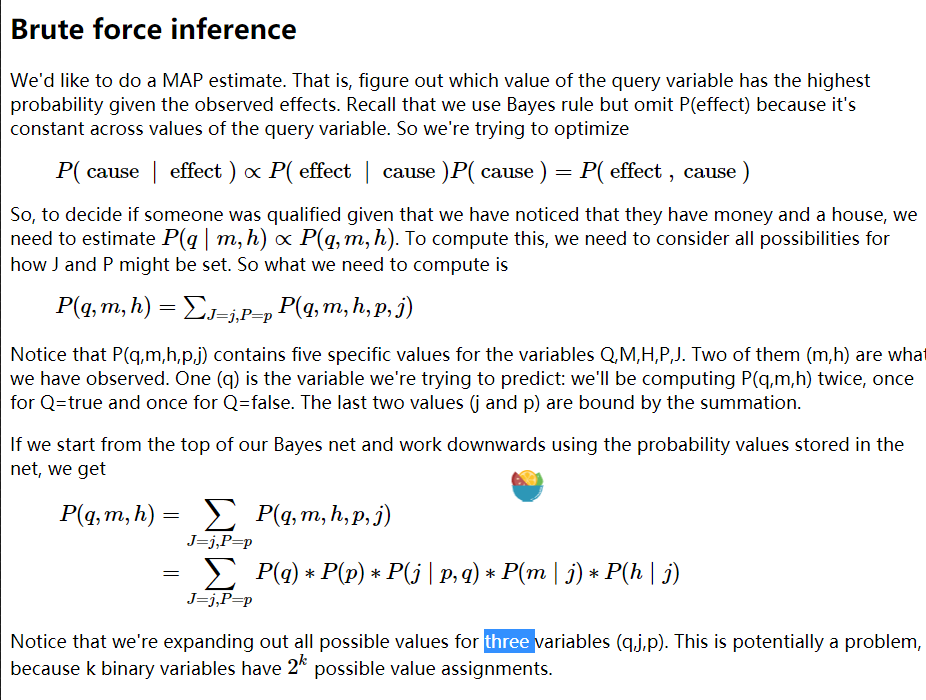
* Comparing size of model to full joint probability table



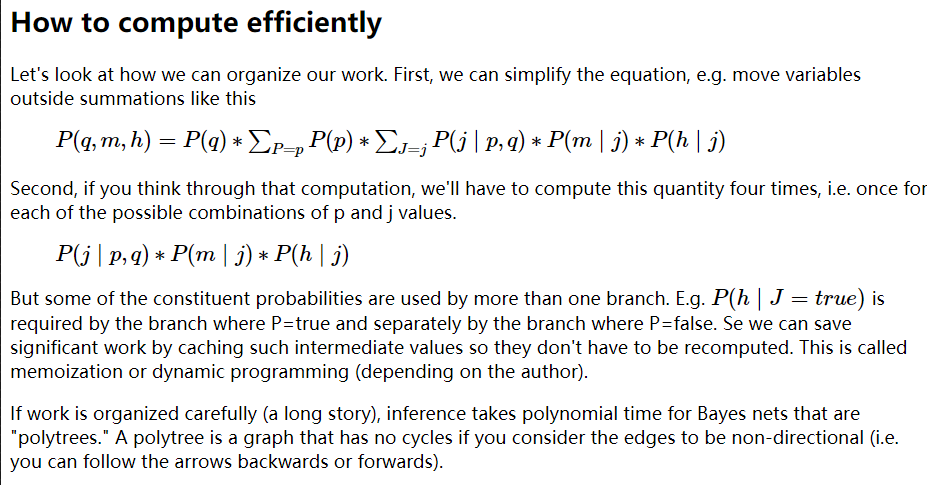




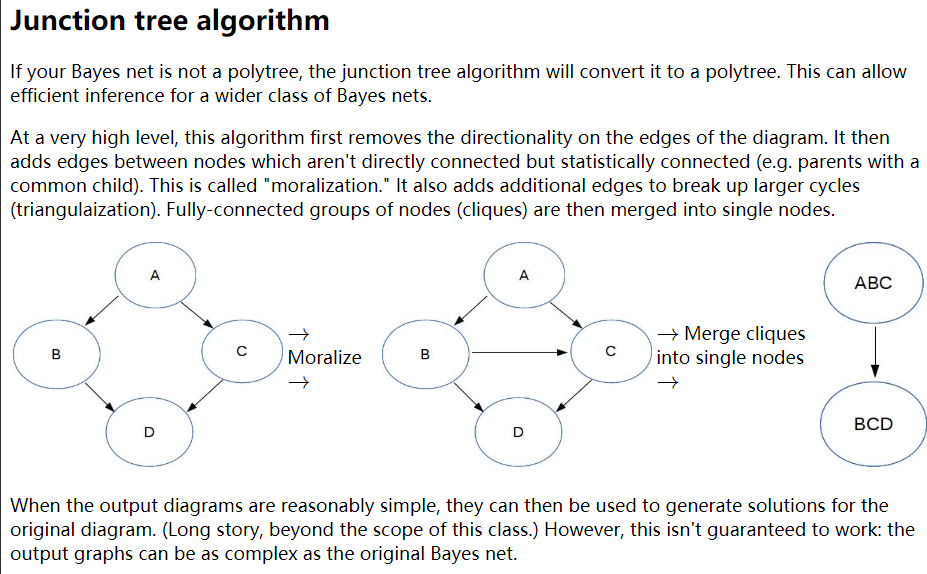
**So the choice of variable order affects whether we get a better or worse Bayes net representation. We could imagine a learning algorithm that tries to find the best Bayes net, e.g. try different variable orders and see which is most compact. In practice, we usually depend on domain experts to supply the causal structure.**



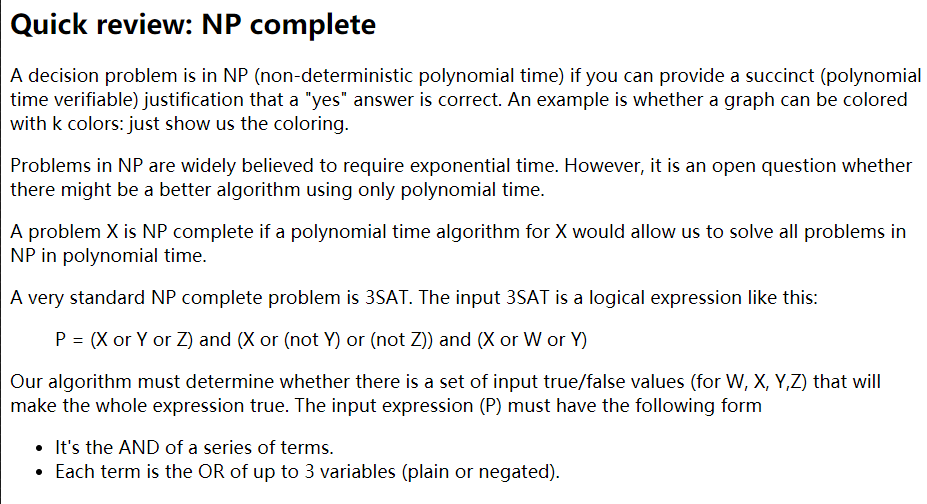
**Notice that we're expanding out all possible values for three variables (q,j,p). This is potentially a problem, because k binary variables have 2k possible value assignments.**



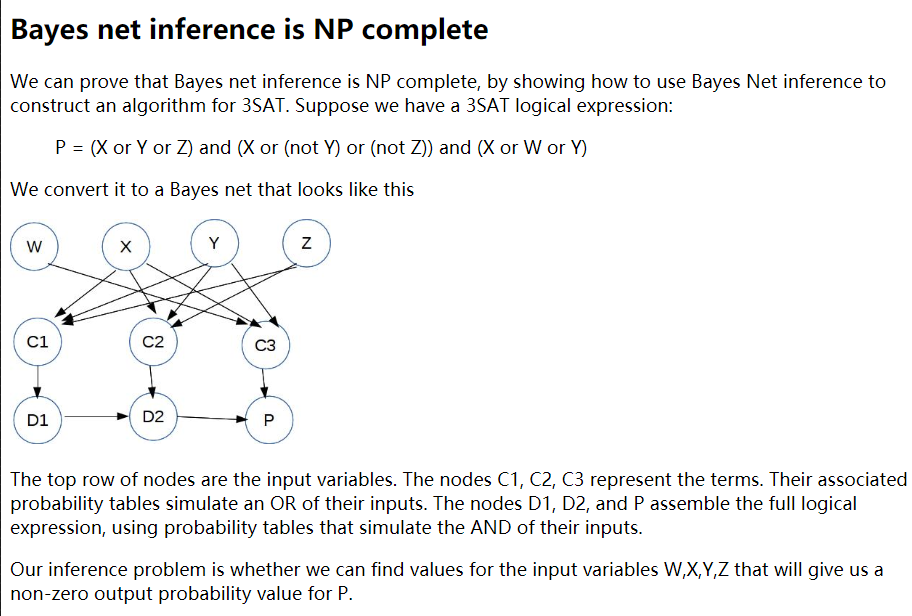
**Bayes nets can be computed in polynomial time when there are polytrees - which means that when you remove the direction from the graph you’re still left with no cycles**

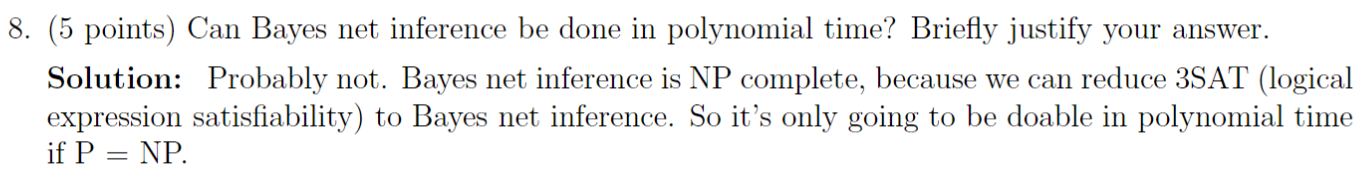


**At a very high level, this algorithm first removes the directionality on the edges of the diagram. It then adds edges between nodes which aren't directly connected but statistically connected (e.g. parents with a common child). This is called "moralization." It also adds additional edges to break up larger cycles (triangulaization). Fully-connected groups of nodes (cliques) are then merged into single nodes.**



**Bayes net inference is NP complete**





**NLP**

Speech or natural language based HCI (human-computer interaction)

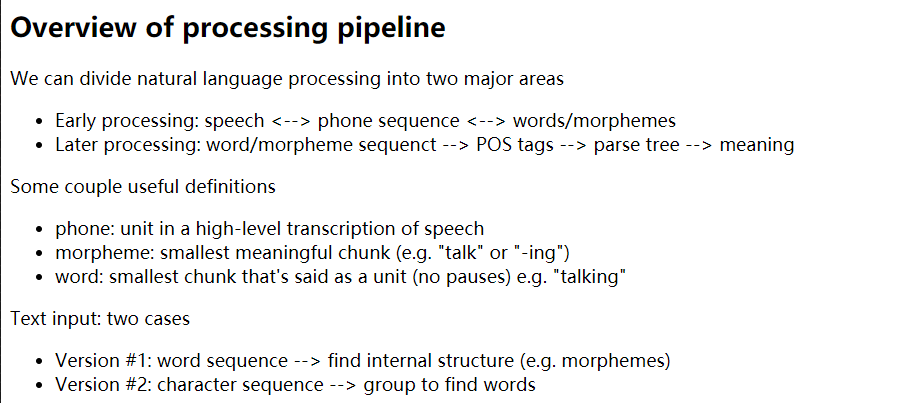
Answering questions, providing information

Transcription/captioning

Evaluating/correcting

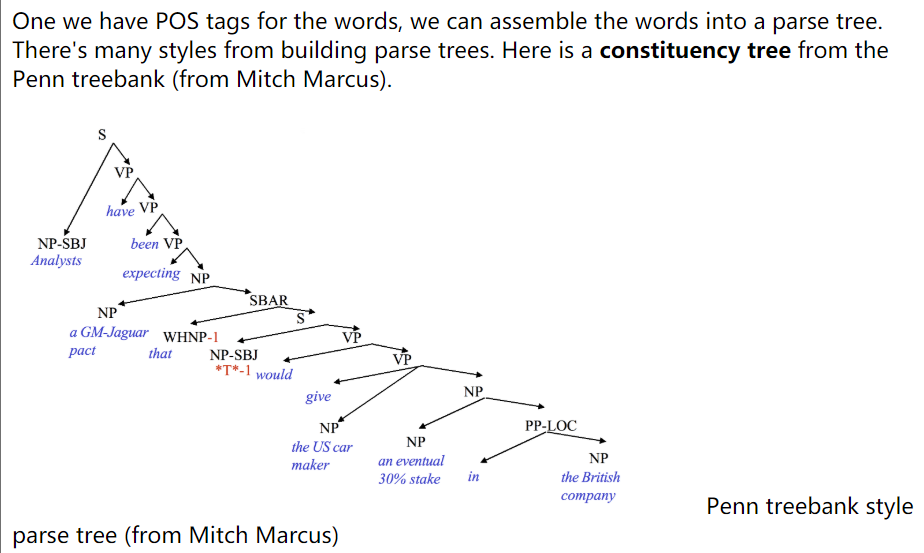
Translation

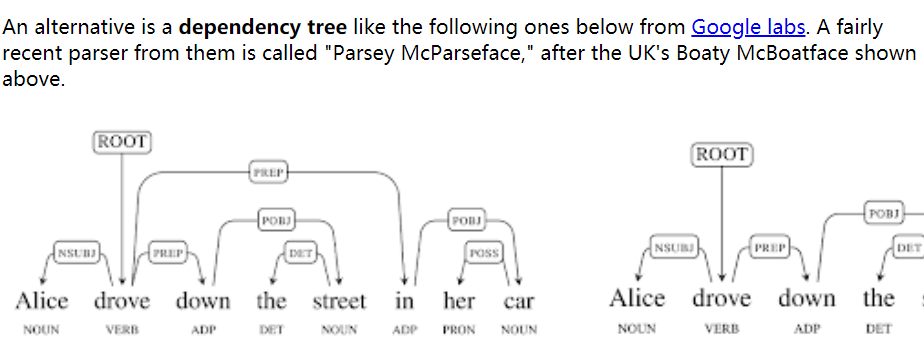
Summarizing information



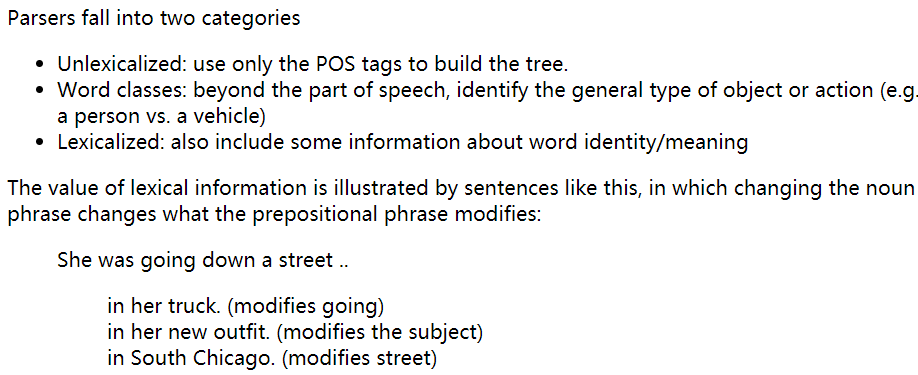
Phonology 音系学

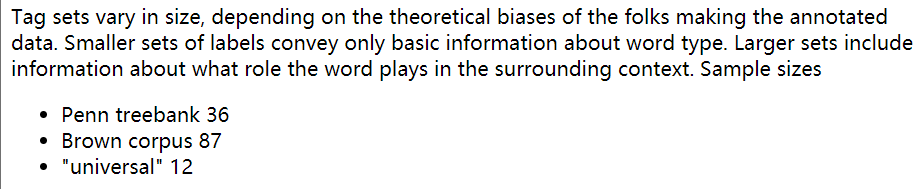
POS tagging

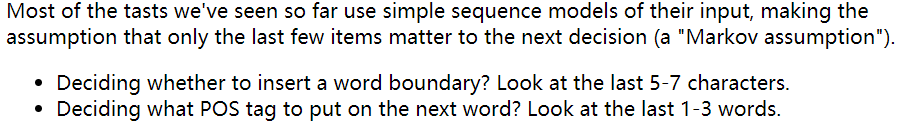




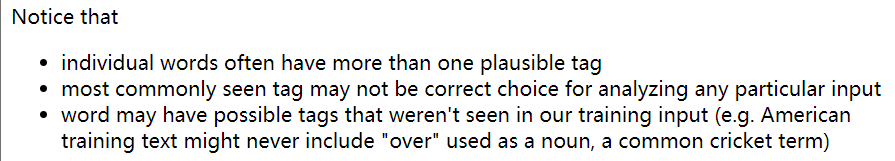
**constituency tree & dependency tree**







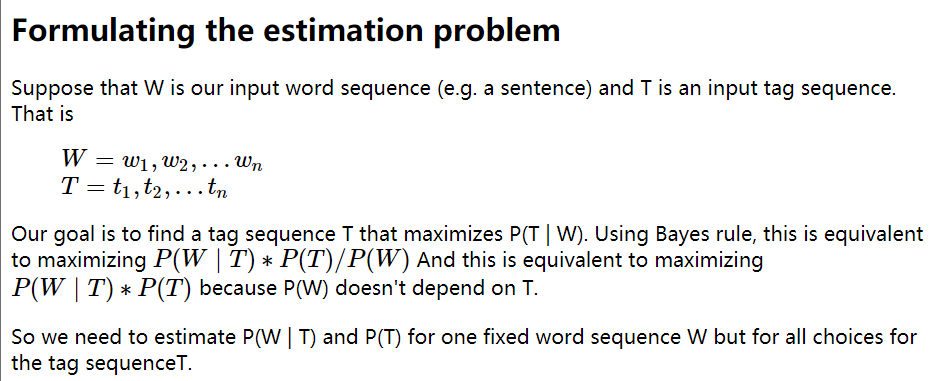
**POS tagging**

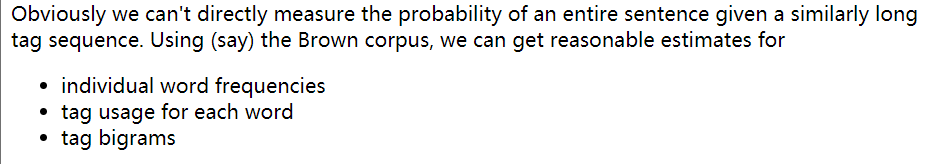


Baseline algorithm

Most testing starts with a "baseline" algorithm, i.e. a simple method that does do the job, but typically not very well. In this case, our baseline algorithm might pick the most common tag for each word, ignoring context. In the table above, the most common tag for each word is shown in bold: it made one error out of six words.

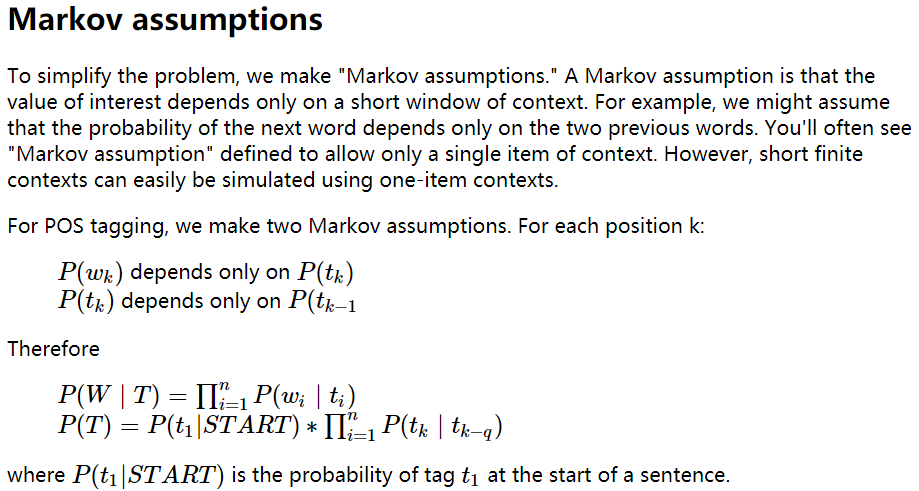
**The test/development data will typically contain some words that weren't seen in the training data. The baseline algorithm guesses that these are nouns, since that's the most common tag for infrequent words.** An algorithm can figure this out by examining new words that appear in the development set, or by examining the set of words that occur only once in the training data ("hapax") words.

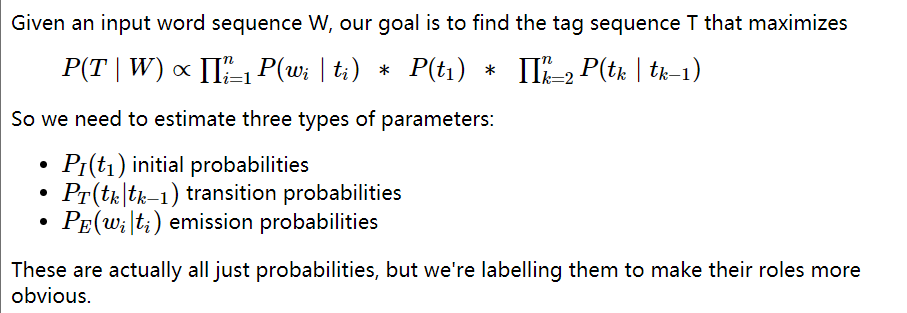


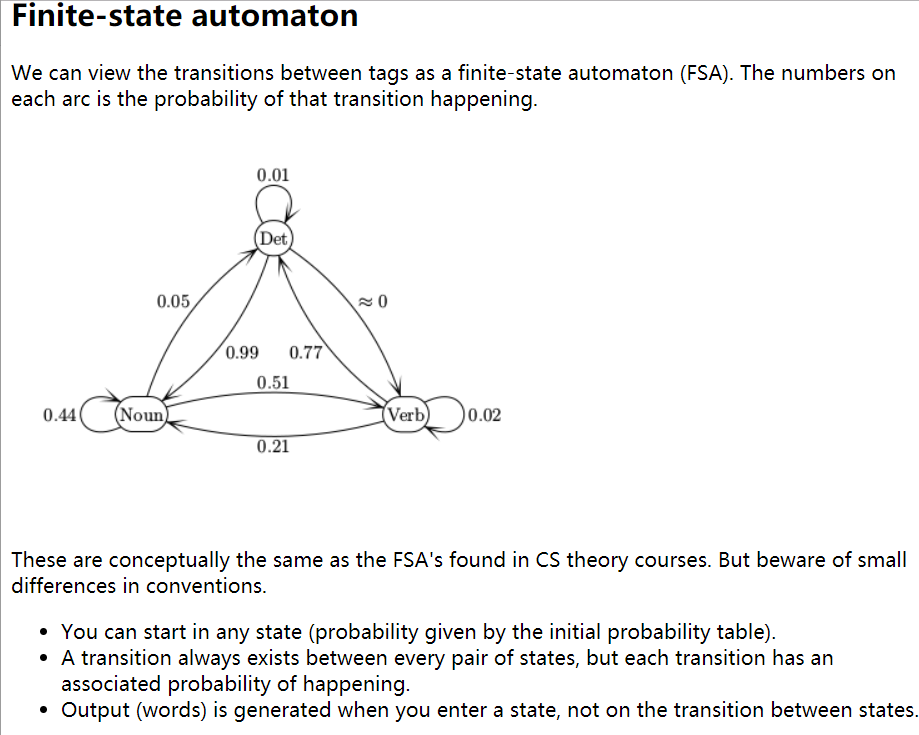


If we have enough tagged data, we can also estimate tag trigrams. **Tagged data is hard to get in quantity, because human intervention is needed for checking and correction.**

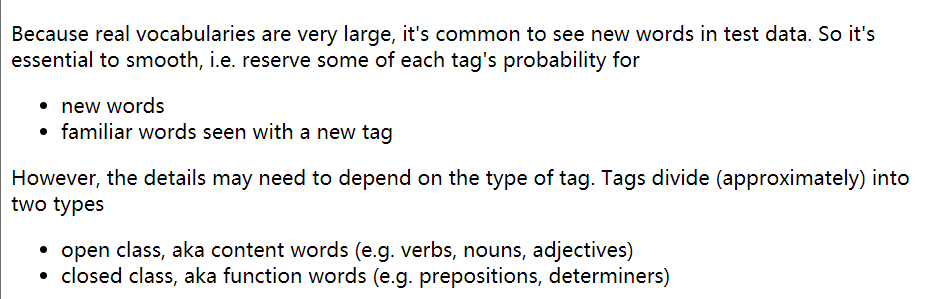
**A Markov assumption is that the value of interest depends only on a short window of context.**



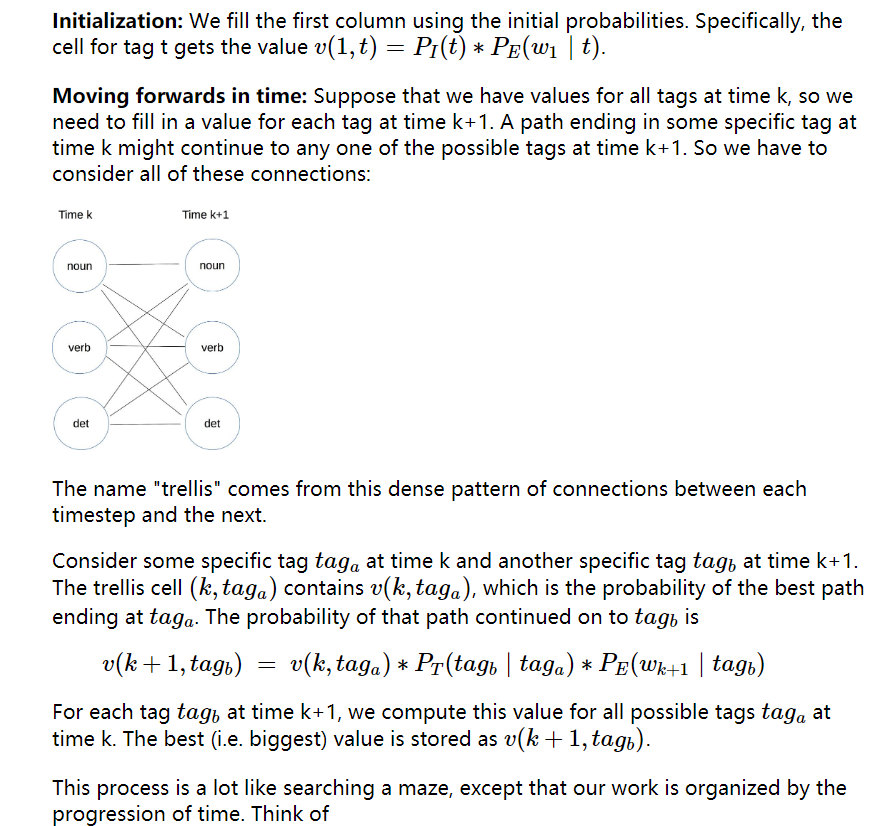


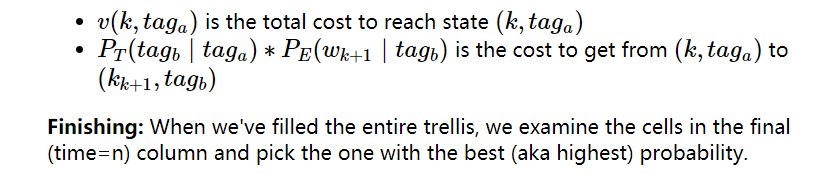


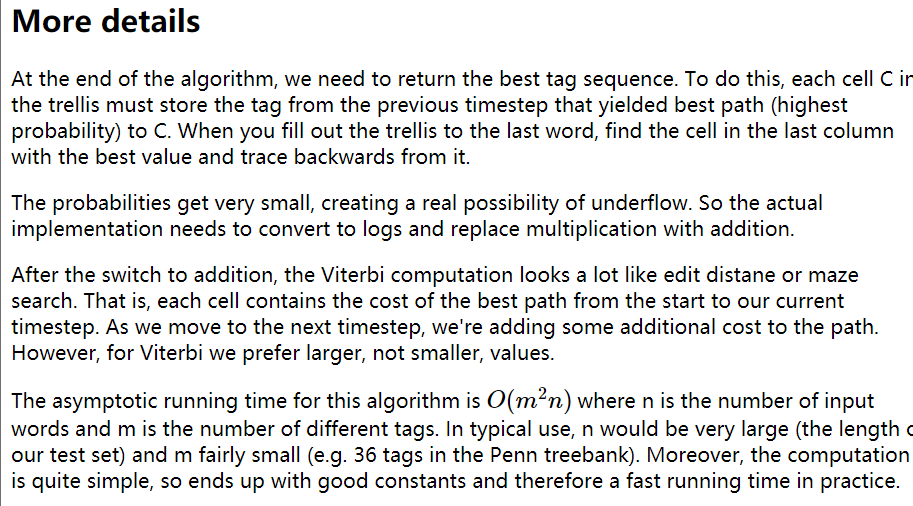
**Output (words) is generated when you enter a state, not on the transition between states.**



**Smoothing for emission probabilities needs to take these differences into account. Suppose that use Laplace smoothing, separately for each tag, to fill in missing emission probabilities. Then each tag will need a different Laplace constant, i.e. a larger Laplace constant for tags that are more likely to generate unseen words.**





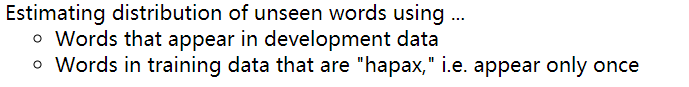


**The asymptotic running time for this algorithm is O(m2n) where n is the number of input words and m is the number of different tags.**

**At what points does an HMM need smoothing?**

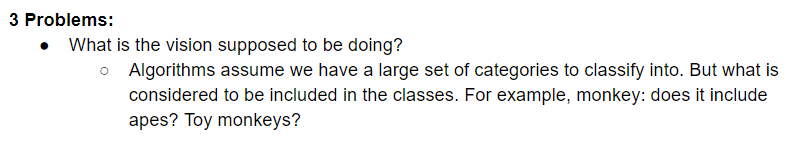
near-zero probability of a transition between tags e.g. from Det to Verb

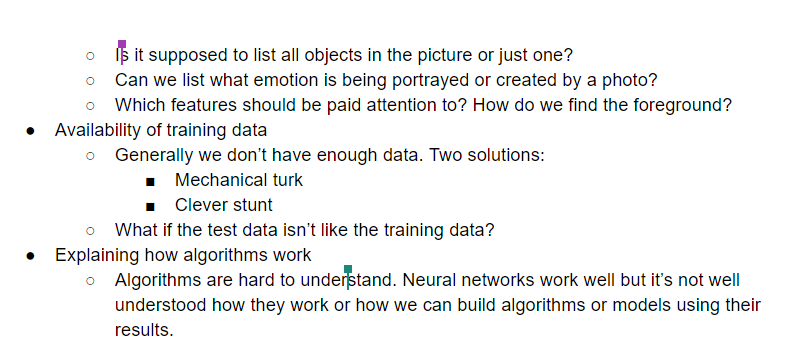
emission probabilities, also need consider open class & closed class



**CV ？**

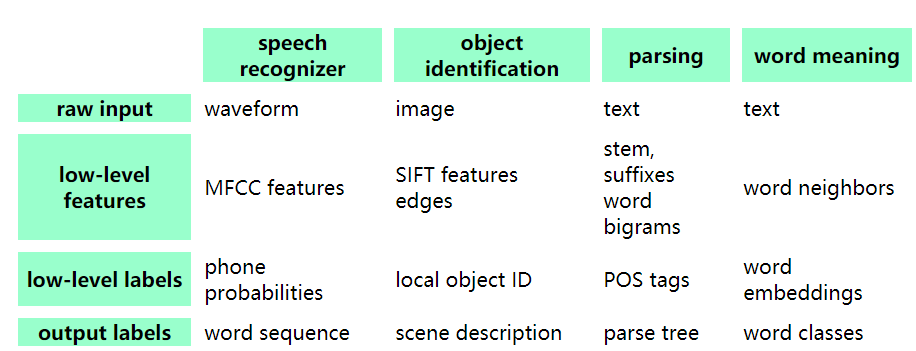
We're good at classification

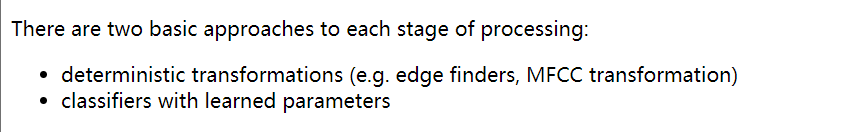


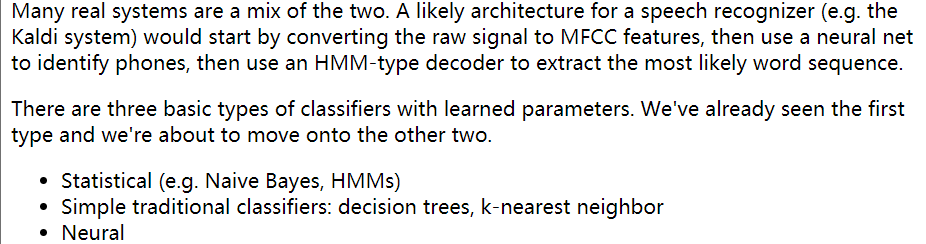


**Classifiers**

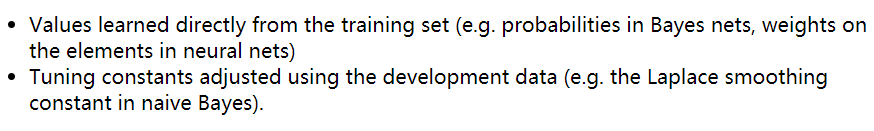
Uses for classification: labelling objects, making decisions

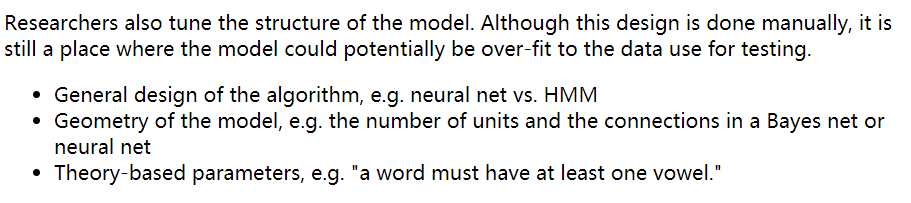


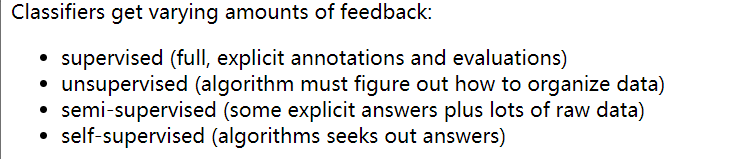


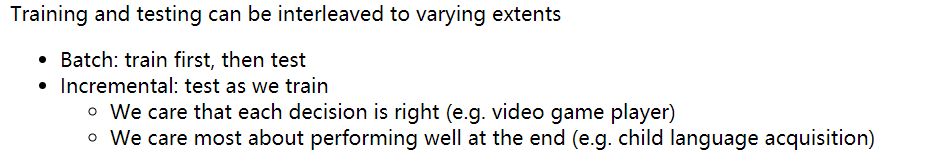


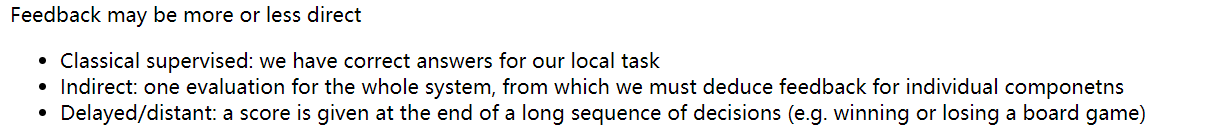
What can we tune?







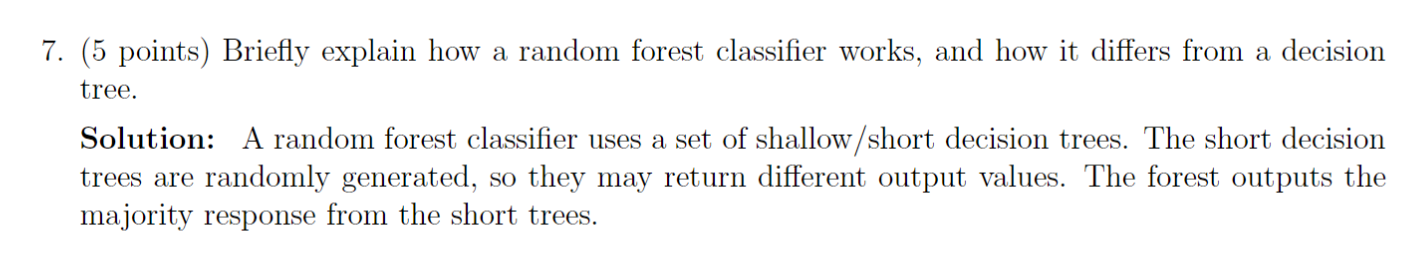




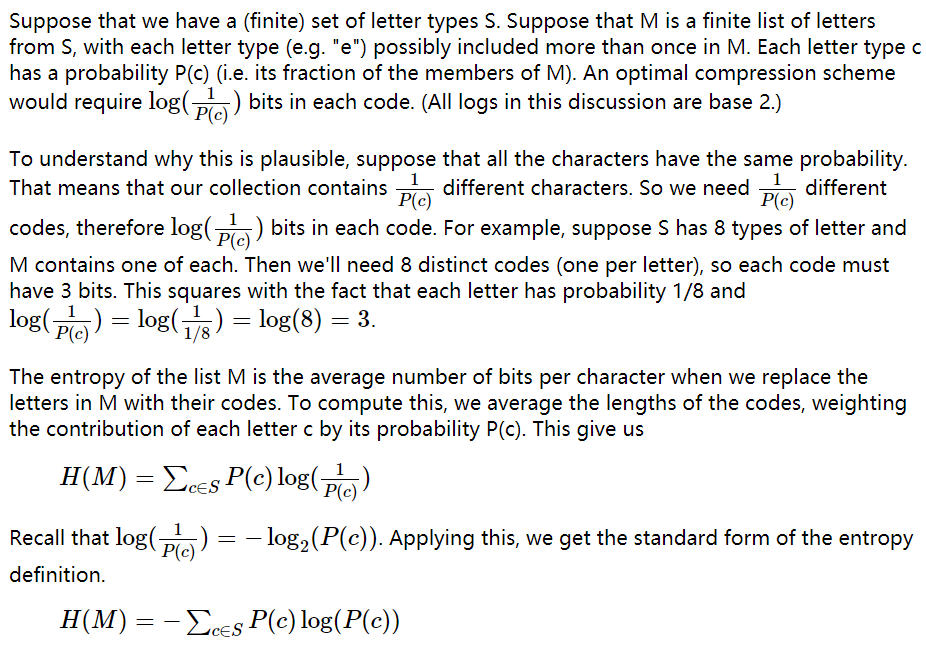
**Decision trees**

**each node in the tree represents a pool of examples.** Ideally, each leaf node contains only objects of the same type. However, that isn't always true if the classes aren't cleanly distinguished by the available features (see the vowel picture above). So, when we reach a leaf node, the algorithm returns the most frequent label in its pool.

**If we have enough decision levels in our tree, we can force each pool to contain objects of only one type. However, this is not a wise decision. Deep trees are difficult to construct (see the algorithm in Russell and Norvig) and prone to overfitting their training data.** It is more effective to create a "forest" of shallow trees (bounded depth) and have these trees vote on the best label to return. **The set of shallow trees is created by choosing variables and possible split locations randomly.**



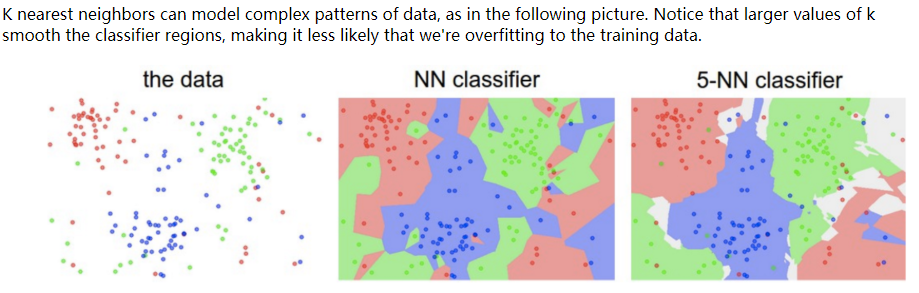
**A good split should produce subpools that are less diverse than the original pool.** If we're choosing between two candidate splits (e.g. two thresholds for the same numerical variable), it's better to pick the one that produces subpools that are more uniform. **Diversity/uniformity is measured using entropy.**



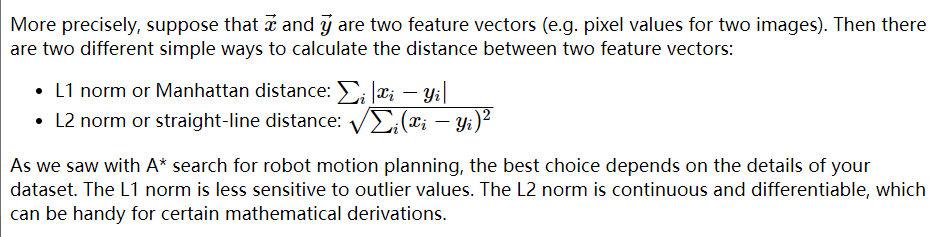
**K nearest neighbors**

Another way to classify an input example is to find the most similar training example and copy its label.

**This "nearest neighbor" method is prone to overfitting, especially if the data is messy and so examples from different classes are somewhat intermixed. However, performance can be greatly improved by finding the k nearest neighbors for some small value of k (e.g. 5-10).**



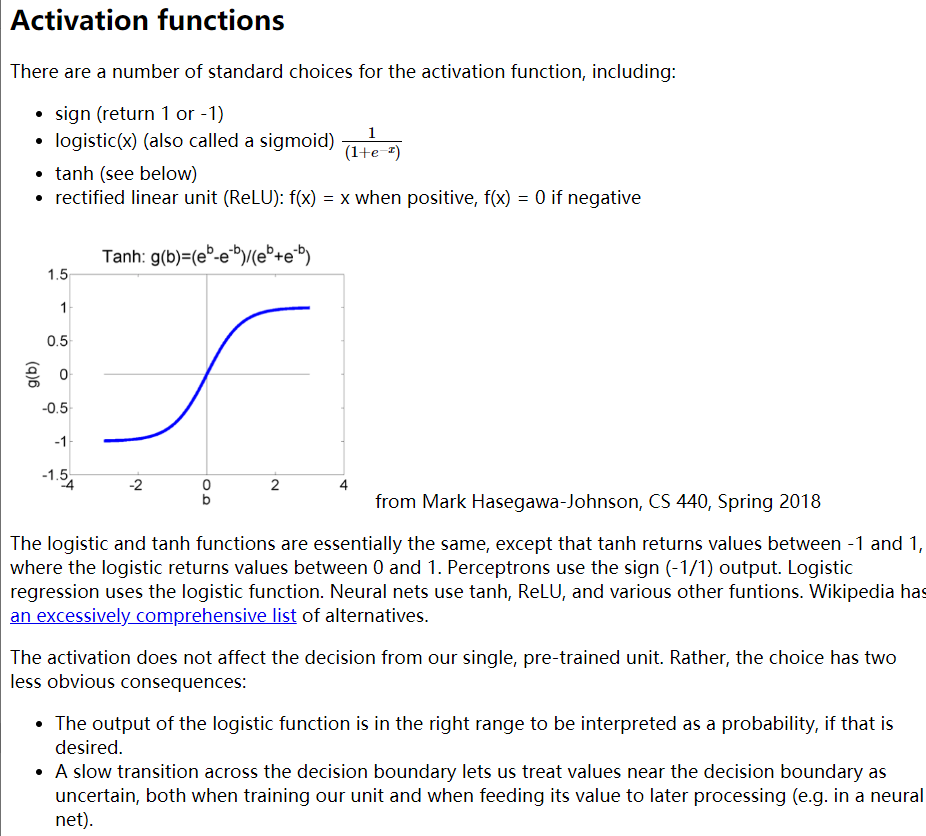
**A limitation of k nearest neighbors is that the algorithm requires us to remember all the training examples, so it takes a lot of memory if we have a lot of training data.** In low dimensions, a k-D tree can be used to find efficiently find the nearest neighbors for an input example. However, efficiently handling higher dimensions is more tricky and beyond the scope of this course.



**The L1 norm is less sensitive to outlier values. The L2 norm is continuous and differentiable, which can be handy for certain mathematical derivations.**

**Classification**

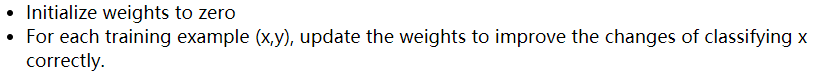
**To make a yes/no decision, we compare the weighted sum to a threshold. The bias term adjusts the output so that we can use a standard threshold: 0 for neural nets and 0.5 for logistic regression**

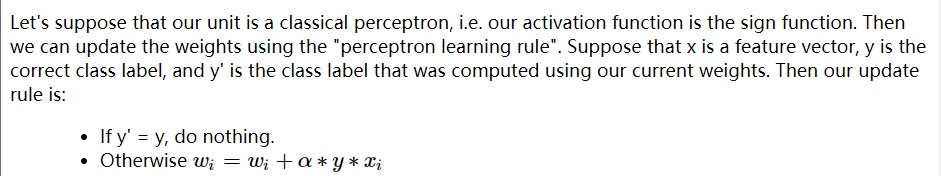


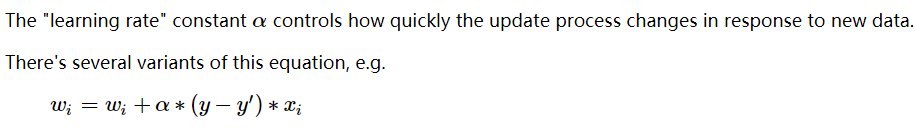
**Perceptron: sign(-1/1) logistic regression: logistic function(sigmoid)**

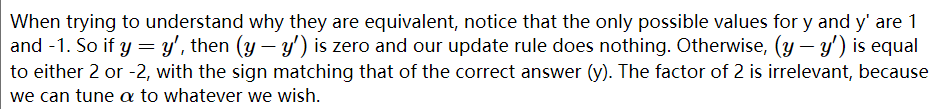
**Neural nets: tanh, ReLU**

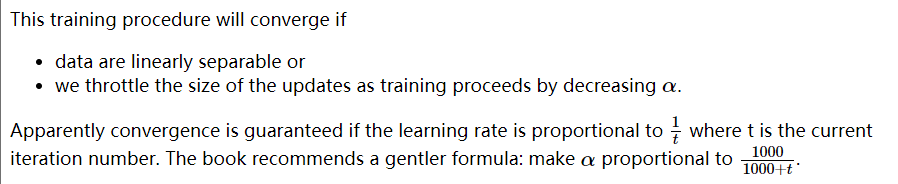
training





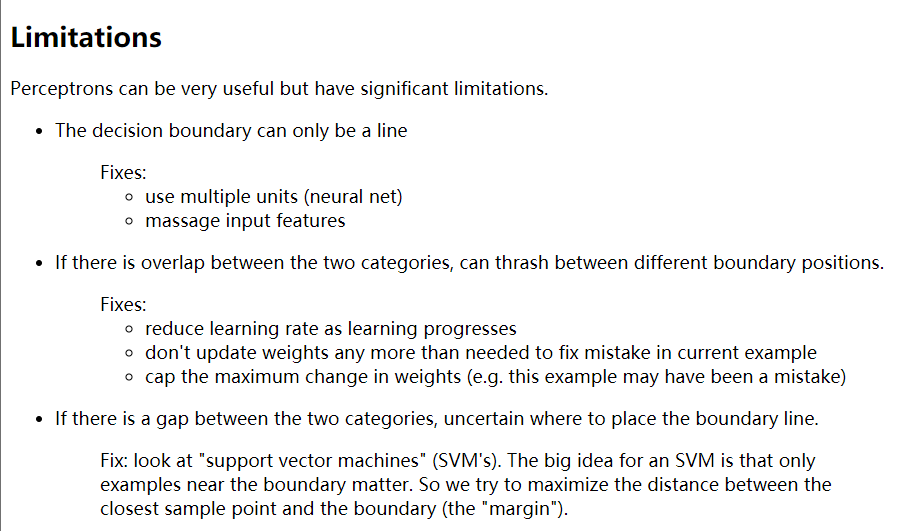






**One run through training data through the update rule is called an "epoch." One generally uses multiple epochs, i.e. the algorithm sees each training pair several times.**

Datasets often have strong local correlations, because they have been formed by concatenating sets of data (e.g. documents) that each have a strong topic focus. **So it's often best to process the individual training examples in a random order.**



Multi-class perceptrons

