NLU Short Questions (May 2010/2012, Aug 2012)

Bayesian Inference

i. What are features of the Bayesian approach to probabilistic modeling and why are they important?

In a non-Bayesian model, we usually maximize the likelihood of some data given some features. For example, consider the MLE formulation:

$$\theta^* = argmax_\theta \prod_i P(x_i \mid \theta)$$

We can improve on this model by giving preference to certain types of parameters over others. For example, consider the MAP formulation:

$$\theta^* = argmax_\theta \prod_i P(x_i \mid \theta)\, P(\theta)$$

Both models above give us point estimates. EM considers a point estimate of the data. MAP also considers a point estimate of the model parameters.

In a Bayesian probabilistic model, however, we consider all possible model parameter settings. We treat θ as a random variable. This means that we can either put priors on our priors (hierarchical priors) and/or consider all possible parameter settings and integrate them out (Bayesian integration). In this way, we're not biased towards some particular point estimate.

Bayesian modeling is great in NLP because it acts as some sort of smoothing (simple integration is equivalent to add-ε smoothing, hierarchical priors give rise to Kneser-Ney) which makes low-data situations more manageable. Additionally, being able to favor sparse parameter distributions (e.g. a Dirichlet with $\alpha_1=...=\alpha_n=\alpha < 1$) is a good fit for the distributions of many observations in natural language.

ii. Consider the task of unsupervised POS-tagging using an HMM. What variables, parameters, and distributions would you need to estimate if you were using the following approaches and which algorithms could you use?

A. Maximum Likelihood Estimation

Distributions:

- Transition probability: $P(t_i \mid t_{i-1})$
- Emission probability: $P(w_i \mid t_i)$

Algorithms:

- Expectation maximization (initialize model parameters, apply model to data and estimate parameters from data until converged).

B. Bayesian Inference

Parameters:

- $\alpha_1,...,\alpha_n$ - smoothing coefficients for the word emission probabilities.
- More commonly, we let $\alpha_1=...=\alpha_n=\alpha$ and have a single smoothing coefficient for all word emission probabilities.

Algorithms:

- Gibbs sampling (assume all but one model parameters are correct and sample data to estimate the other parameter).
- Bayesian integration to find an analytic solution to the inference problem (equivalent to add-ε smoothing).

Distributional Semantics

i. Why is dimensionality reduction argued to be important when using LSA?

A. For information retrieval.

- High dimensions lead to data sparsity.
- Distance metrics are less discriminative in high dimensional spaces.
- More dimensions leads to higher computational complexity i.e. slower retrieval.
- In reality, our data-set probably only has a small, limited number of topics. If we assume too many, we'll fragment the actual topics we have over multiple pseudo-topics. This means that we'll model noise and might introduce search errors.
- Not all words are equally distinctive and important (we can probably just drop low tf-idf words without losing much retrieval performance).

B. For cognitive modelling.

- Humans probably only account for a limited number of topics/words so fewer dimensions lead to a more principled model.
- Humans have an internal notion of polysemy that the LSA vector-space model does not account for. Dimensionality reduction collapses related words and therewith fixes this to a limited extent.

ii. Give three arguments in favor of LDA over LSA as a semantic representation, using specific examples when possible.

- Probabilistic.
- Generative (more principled).
- Fewer parameters to set and tune.
- LDA parameters have a well defined effect on the model (smoothing).
- Works better for small amounts of data.

iii. Assume that you are given a set of target words (e.g., fish, bowl, sea, boat, shark) for which you wish to create a spatial meaning representation.

A. Describe a simple algorithm for extracting a semantic space for the above words from a large corpus.

Simple frequency space:

1. Remove stop-words from texts (e.g. the, a, …).
2. Count number of co-occurrences of all words in a N-word window around the target words.
3. Build matrix: rows are target words, columns are context words, entries are co-occurrence counts.
4. The meaning of a word is given by its context co-occurrence statistics.

B. How would you go about measuring whether fish and shark are distributionally similar? Explain using two formulas.

The similarity between two words is given by the similarity of their context-vectors. If we imagine the semantic space as a vector space indexed by context words where target words are points, the similarity between two target words is the distance between points in this space.

We can use any distance metric to measure the similarity. The following two are popular:

- Cosine similarity: $cos(\overline{a}, \overline{b}) = \sum_i a_i b_i \,/\, [(\sum_i a_i^2)^{0.5} (\sum_i b_i^2)^{0.5}]$
- Euclidean distance: $|\overline{a}, \overline{b}| = sqrt(\sum_i (a_i - b_i)^2)$

Discourse Coherence

i. Outline the entity grid, Barzilay and Lapata's model of discourse coherence.

1. Given a corpus of documents consisting of sentences.
2. Extract entities (people, places, companies, …) and their grammatical functions (subject, object, other, not present), ignore all non-entity tokens.
3. Create a matrix where columns are entities and rows are sentences. Grid cell (i,j) is the grammatical function of entity j in sentence i.
4. The coherence of a document is given by the length of runs of the entity-role combinations.
5. We can train an SVM to rank entities according to their importance and coherence.
6. We can make the model probabilistic by collecting frequency statistics of n-length state-state transitions across the entire document.

ii. Briefly describe the data used to train this model. Is the model supervised or unsupervised?

- Data used: raw documents. We assume that the original order is highly coherent and use permutations of the original documents as examples of non-coherent text.
- The model is supervised but we don't need annotated training data.
- (for entity extraction and SOX- designation it uses a (hopefully-gold-standard) parse)

Discriminative Modeling

i. Using formulas, briefly explain the difference between a discriminative and a generative approach to probabilistic modeling.

Generative formulation: $\theta^* = argmax_\theta \prod_i P_\theta(x_i, y_i)$

Discriminative formulation: $\theta^* = argmax_\theta \prod_i P_\theta(y_i \mid x_i)$

The generative paradigm aims to explain how some feature and some data was created - it performs maximization over the entire data-set. The discriminative paradigm only aims to distinguish a feature among similar data items - it performs maximization among similar items.
The former is usually more principled whereas the latter can use fancier (non-local) features which don't fit into the generative paradigm.
Discriminative models can be applied on top of generative models. Usually done in practice (most commonly, but not exclusively, this leads to better performance).

ii. In part of speech tagging, we are normally interested in $P(\bar{t}, \overline{w})$, the joint probability of a tag sequence and a word sequence. Give a discriminative formulation of this problem (formula and explanation).

By the chain rule we can write the joint probability as $P(t, w) = P(t \mid w)\, P(w)$. This gives us a formulation for a POS-tagger:

$$P(t \mid w) = P(t, w) / P(w).$$

A POS-tagger is thus a distribution of tags given words: how often do we see a tag together with a word relative to the frequency of the word in and of itself.
We can also try to model this directly (Johnson paper):

$$P(t \mid w) = \prod_{i=1}^{n+1} P(t_i \mid t_{i-1}, w_i).$$

Here, we maximize the likelihood of a tag for a given word relative to other similar contexts (i.e. situations involving the same word and tag-history).

iii. Describe two advantages and two disadvantages of discriminative models compared to generative ones.

Advantages:

- Usually performs better.
- Can be used in addition to a generative model (e.g. as a n-best re-ranking step).
- Can use features that don't fit in the generative framework (e.g. non-local features).
- Automatically distinguishes between good and bad features. This might give us insights into what's actually important to predict a given quantity.

Disadvantages:

- Not generative i.e. can't explain the data.

- Often somewhat ad-hoc.
- Requires coming up with good features (i.e. feature engineering).
- Sometimes more difficult to estimate/train.

Incremental Parsing

i. Give a definition of incremental parsing, and list two applications of an incremental parser.

An incremental parser processes a sentence word-by-word and produces a parse tree + probability for any prefix of a sentence (strong incrementality always produces a tree rooted at a S-node).

Applications:

- Psycholinguistic modeling (e.g. explaining garden path sentences) - we parse meaning on-the-fly (we don't wait until the end of an utterance to start understanding it) so incremental parsing is a better model of what goes on in a human brain than traditional approaches.
- Language modeling (e.g. in automatic speech recognition). The probability of a (partial) sentence is the sum over all the possible (partial) derivations of the words. Since we update the language model probability with every new word, we always have an estimate for the likelihood of a string of words. This is useful in fields like ASR where we need to perform real-time language modeling.

ii. Describe the distinctive features of Roark's incremental parser in terms of parsing strategy, central data structure, probability model, and features.

Parsing strategy:

- Top-down.

Central data structure:

- Priority queue.

Probability model:

- $P(w_{i+1} \mid w_0^i) = P(w_0^{i+1}) \,/\, P(w_0^i) = \sum_{d \in D_{w_0}^{w_{i+1}}} P(d) \,/\, \sum_{d' \in D_{w_0}^{w_i}} P(d')$
- The probability of the next word given the previous words is the sum of all derivations of the new prefix relative to the old prefix.

Features:

- Lexical/headword annotation.
- Parent/grandparent/sibling annotation.
- POS tag.

iii. The Roark's parser computes a quantity called lookahead probability (LAP).

How is it computed and what is it used for?

The lookahead probability is:

$$LAP(S, w_i) = \sum_{\alpha \in V \cup T} P(A_0^n \rightarrow w_i \alpha)$$

How likely are we to be able to parse the next word with the current hypotheses and one or more grammar rules applications.
The quantity is used for pruning the priority queue S: any derivation that falls below a certain LAP is discarded and no longer pursued.

Word Senses

i. What is the main difference between word sense disambiguation and discrimination?

- Disambiguation
    - Given an inventory of known senses of a word (e.g. from a thesaurus like WordNet), select the most likely one for some context.
- Discrimination
    - Cluster words according to their context to find terms that relate to the same concept (i.e. induce the possible senses of words).

ii. Describe an example model for each paradigm (i.e., disambiguation or discrimination).

- Disambiguation
    - Graph-based WSD
        - The nodes of the graph are all the senses of all the words in some context. Connect the sense by walking through the WordNet taxonomy and adding edges between related words. Rank the nodes according to some graph importance measures (e.g. PageRank, degree centrality or edge density) - the highest ranking sense is the correct sense for the word.
    - Standard supervised setting
        - Use features like POS tag, context words, etc to learn a model that classifies a word according to its senses.
- Discrimination
    - Yarowski algorithm
        - Given some starting knowledge, train a classifier, use the classifier to label the remaining data and retrain the classifier until convergence.
    - Bayesian WSD
        - A word is a distribution over possible senses, use LDA to model the distribution and cluster words according to senses.