Problem Set 2: Solutions

Problem 1: Earnest Spelling

a. We can define the weighted CSP as follows:

Variables: we will create *n* variables $t_1, \ldots, t_n$ that represent the sequence of corrected words.

Domains: $t_i \in D$, i.e. each t_i is a valid dictionary word

Constraints:

1. We have *n* unary constraints with weight $Edit(t_i, w_i)$. (w_i is not a variable as the value is given/fixed).
2. We have *n-2* ternary constraints with weight $-Fluency(t_i, t_{i+1}, t_{i+2})$.

Note that the actual (boolean) constraints are unimportant to this formulation. If we want, we can define "dummy" constraints that can never be satisfied in order to ensure that we always pay the constraint cost.

The solver will find an assignment to the variables with minimum total weight -- thus, we are penalizing edit distance and rewarding fluency.

b. We can add *n* new binary variables to our model from part (a):

$v_i \in \{True, False\}$ where $1 \leq i \leq n$

Conceptually, v_i will represent whether any of the first *i* words is a verb. To achieve this, we can add constraints "recursively" as follows:

1. $v_1 = True$ iff t_1 is a verb.
2. $v_i = True$ if (v_{i-1} is *True* or t_i is a verb), and *False* otherwise.

We also need a constraint to ensure that at least one verb is present:

3. $v_n = True$

These are hard constraints, so there is no associated weight.

c.

There are many good answers to this question. Here is a selection:

CSP pros/DSP cons:

- The language of variables and constraints can offer a more natural way to model this problem. For a DSP, you might have to add in extra data to your states in order to keep track of certain quantities, e.g. the fluency over the last three words. This can get ugly

very quickly.
- Negative weights limit the algorithms that we can apply to DSPs (Bellman-Ford), which is not an issue for CSPs.
- CSP solvers can use general-purpose optimizations (arc consistency, most/least constrained variables, etc.) to find solutions more efficiently than DSP search. CSP can assign variables in any order, which is a limitation of the sequential DSP search.
- CSP solvers can sometimes take advantage of the structure of the constraint graph to factorize the problem and solve it much more efficiently. Such a technique could solve the the spell checker problem in $O(nd^3)$ time, where *d* is the size of the dictionary.
  NOTE: Although DSP search may be slower and/or intractable depending on your problem setup, $O(d^n)$ [exponential] is *not* the inevitable runtime of DSPs unless you define your states very poorly. In fact, a reasonable DSP formulation for this problem using Bellman-Ford will also find the optimal solution in $O(nd^3)$ time.

CSP cons/DSP pros:
- Other types of problems can be naturally expressed as a DSP (e.g. driving directions), especially in cases where there are well defined goals.
- DSP search algorithms tend to be much simpler than CSP solvers. [This is less of an issue if you are using off-the-shelf solvers]. With good heuristics and sensible state definitions, DSP performance can be quite good.
- Both are NP-hard in the general case. A poor CSP formulation can be just as bad (or worse) than a simple DSP search.

Problem 2: Cancer Warning Signs

a. modelC. Model C is the only model in which all three nodes, G_{mother}, G_{father}, and G_{child} are parent nodes and not children of any other nodes.

b. modelA. Model A represents the relationship of genes influencing phenotype (genes influencing whether an individual has cancer) as well as the relationship between the parent and child gene transfer. Model B is not correct because it depicts a relationship between whether the mother or father has cancer and the child's genotype.

c. The transfer of alleles from the mother is independent of that from the father. Each parent can give either a B or b, and each allele has ½ chance of being transferred. The possible combinations for the child's genotype are then BB, Bb, bB, bb, where the first allele is from the father, and the second allele is from the child. Thus, we can calculate the following probabilities.

Genotype	Probability
BB	0.25
Bb	0.5
bb	0.25

d.
i. $P(C_{father} = true) = P(C_{father}|G_{father} = BB)P(G_{father} = BB) + P(C_{father}|G_{father} = Bb)P(G_{father} = Bb) + P(C_{father}|G_{father} = bb)P(G_{father} = bb) = \theta_B*\phi_{BB} + \theta_B*\phi_{Bb} + \theta_b*\phi_{bb} = P(C)$

Using shorthand notation: $P(C) = P(C_{father} = true)$, $P(BB) = P(G_{father} = BB)$, etc
$P(BB|C) = P(C|BB)P(BB)/P(C) = \theta_B * \phi_{BB} / P(C)$
$P(Bb|C) = \theta_B * \phi_{Bb} / P(C)$
$P(bb|C) = \theta_b * \phi_{bb} / P(C)$

ii.
If the mother's genotype is bb, the child will always have at least 1 b, so the child can never have genotype BB.
P(child = BB|mom = bb, dad's distribution) = 0

The mother will always pass the b allele, which means in order for the child to be genotype Bb, the father must pass a B. There are two ways the father can pass on a B: if the father is of genotype BB, he will pass on a B with probability 1. If the father is of genotype Bb, he will pass on a B with probability ½:

P(child = Bb | mom = bb, dad's distribution) = $(\theta_B * \phi_{BB} + ½ * \theta_B * \phi_{Bb}) /$ P(C)

In this case, in order for the child to be genotype bb, the father must pass a b. There are two ways the father can pass on a b: if the father is of genotype bb, he will pass on a b with probability 1. If the father is of genotype Bb, he will pass on a b with probability ½:

P(child = bb | mom = bb, dad's distribution) = $(½ * \theta_B * \phi_{Bb} + \theta_b * \phi_{bb}) /$ P(C)

iii. The father and the child have the same emission probabilities for breast cancer, so using the probabilities calculated in part ii,
P(Cchild | mom = bb, Cfather) = $[\theta_B * (\theta_B * \phi_{BB} + ½ * \theta_B * \phi_{Bb}) + \theta b (½ * \theta_B * \phi_{Bb} + \theta_b * \phi_{bb})]$/P(C) =
$(\theta_B^2 * \phi_{BB} + ½ \theta_B^2 * \phi_{Bb} + ½ \theta_b * \theta_B * \phi_{Bb} + \theta_b^2 * \phi_{bb}) / (\theta_B * \phi_{BB} + \theta_B * \phi_{Bb} + \theta_b * \phi_{bb})$

Problem 3: Jazz

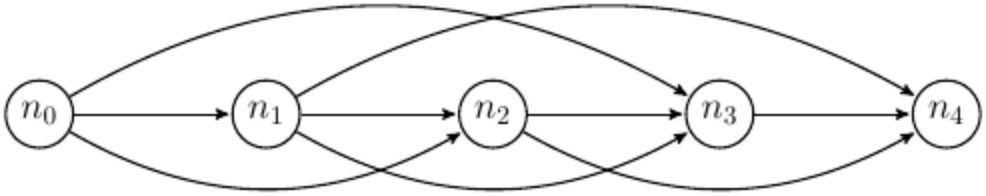
(a) The conditional probability table (CPT) will be a 7x7 table in which the rows correspond to the state at time t and the columns correspond to the state at time t+1, and thus the entry in row i, column j corresponds to the probability of playing note j at time t+1 given that note i was played at time t. A visualization of the table with a sample cell filled in is:

t \ t+1	**C**	**E**	**Eb**	**G**	**G#**	**B**	**rest**
C							
E				P(play G at time t+1 | played G at time t)			
Eb							
G							
G#							
B							
rest							

The rows of the table will need to sum to 1 (in pretentious math terms, it needs to be a right stochastic matrix :3)

(b) There were two ways of approaching this problem. One (call it (i)) was to simply extend the model from (a) so that the current state depends on the past 3 states (mathily, this is called a Markov chain of order 3). Another (call it (ii)) was to create a new model in which the state contained a 3-tuple of notes N_t = (n_t, n_{t-1}, n_{t-2}), where n_t represents the last note played, n_{t-1} represents the note played before that, and n_{t-2} represents the note played before n_{t-1}.

If you took approach (i), your CPT would be a 343x7 table in which the rows correspond to all 7^3 sequences of 3 notes that could occur and the columns correspond to the next note played. The beginning of the directed graph visualizing your model would look like



whereas if you took approach 2, your CPT would be a 343x7 table where each row corresponds to one of the 7^3 possible N-tuples of notes (aka states) that you could be in, and each column corresponds to the note which will become the new first entry in the 3-tuple in the next state. The beginning of your directed graph would look like

(c) Since the two models from part (b) are equivalent, we won't worry about their details and just say that we're trying to use the hundreds of hours of songs to learn the probabilities P(n_{t+1} | n_t, n_{t-1}, n_{t-2}). That is, the probability of some note being played at time t+1 given the notes played at times t, t-1, and t-2. To do this, we take a frequentist approach and use normalized counts as our probability distribution. In particular, we set the aforementioned probabilities as

$$P(n_{t+1} = x_1 | n_t = x_2, n_{t-1} = x_3, n_{t-2} = x_4)$$

$$= \frac{\#\text{ of times we hear } x_1, x_2, x_3, \text{ and } x_4 \text{ played in sequence}}{\#\text{ of times we hear } x_2, x_3, \text{ and } x_4 \text{ played in sequence}}$$

(d) There was a simple solution and a crazy alignment-based solution to this problem. The simple solution was a standard HMM in which our observed variables were the observed notes m_1, …, m_T and the hidden variables were the score notes n_1, …, n_L. The CPT for emissions is a 7x7 table where the rows correspond to a score note and the columns correspond to a played note, such that the cell at row i column j gives the probability of the musician playing note j given that they are on score note i. The CPT for transitions is an LxL table where the rows and columns correspond to score positions, and the cell at row i column j gives the probability that the musician who was previously on score position i moves to score position j. Since the musician always either stays on a score position or jumps to a subsequent position, any cell such that j < i will contain a 0 (more mathanese: it will be an upper triangular matrix).

The crazy solution was to model it as a sequence alignment problem and use a pair HMM to

solve it. I could rant about this for a long time since I think it's awesome, but since people much better at explaining things than I have already done it I'll just post these slides: https://edit.ethz.ch/bsse/cbg/education/spring2012/smcb/lectures/04_Hidden_Markov_models_for_sequence_alignment.pdf
Just realize that we have a sequence of score notes and a sequence of observed notes that the musician actually played, and we are trying to find the best alignment between the two sequences. Therefore our pair HMM can emit (note, note), (note, gap), or (gap, note), and we assign probabilities to these emissions and the transitions between the states that generate them and voila!

(e) There were again many ways to approach this problem, but it needed to be set up such that it matched the specification of your model from part (d). In our case, going with the simple model, an individual particle would represent a position on the score, the particles would holistically represent our belief about the musician's position on the score. We could initialize the particles uniformly at random throughout the score, or we could be a bit more smart and initialize them via a distribution that was biased towards notes at the beginning of the score (i.e., a Geometric disctribution http://en.wikipedia.org/wiki/Geometric_distribution). The update and transition steps would be the standard methods taught in class, that is, the update is based on your emission probabilities while the transition is based on your transition probabilities.

Perhaps your HMM will define the shape of jazz to come! Either way this is an excuse for me to plug one of my favorite jazz albums that you should totally check out if you're about that life