Organizing Web Information (CS 728) **Quiz**
Computer Science and Engineering **2017-04-03 Monday**
Indian Institute of Technology Bombay **17:30–19:30 SIC 305**

NAME _____ ROLL _____

This exam has 8 printed page/s. Write your name and roll number on **EVERY SIDE (and not just sheet)**, because we may take apart your answer book and/or xerox it for correction. Write your answer clearly within the spaces provided and on any last blank page. Do not write inside the rectangles to be used for grading. **If you need more space than is provided, you probably made a mistake in interpreting the question.** Start with rough work elsewhere, but you need not attach rough work. Use the marks alongside each question for time management. **Illogical or incoherent answers are worse than wrong answers or even *no* answer, and may fetch negative credit.** You may not use any computing or communication device during the exam. You may use textbooks, class notes written by you, approved material downloaded **prior to the exam** from the course Web page, course news group, or the Internet, or notes made available by me for xeroxing. If you use class notes from other student/s, you must obtain them **prior to the exam** and **write down his/her/their name/s and roll number/s** here.

1. PCFG potpourri.

1.a Suppose we train a HMM and a PCFG with suitable states on a large natural English corpus, and then calculate

- $\Pr_{\text{HMM}}(\text{john decided to bake a})$
- $\Pr_{\text{PCFG}}(\text{john decided to bake a})$

Which one is expected to be larger and why? | | | 1 |

(This is obviously a highly qualitative statement subject to lots of details.) $\Pr_{\text{HMM}}$ is expected to be larger, because this sequence of words, or something similar, may be quite frequent, whereas a PCFG for whole sentences is highly unlikely to end without the object of *bake*.

1.b Consider the following PCFG with corresponding rule probabilities:

Rule	Probability
$S \to S\,S$	p
$S \to$ hello	$1-p$

Here S is the only nonterminal and hello the only terminal. What are the probabilities of these strings as functions of p?

- hello hello _____
- hello hello hello _____

| | | 2 |

- hello hello $p(1-p)^2$
- hello hello hello: This can be produced in two ways: $(S\ S)\ S$ and $S\ (S\ S)$. Therefore the probability is $2p^2(1-p)^3$.

1.c With the same 2-rule PCFG specified above, let L_h be the total probability of all parse trees with depth at most h. For example, with the smallest tree with $h = 2$ has the single production $S \to$ hello with probability $L_2 = 1 - p$. To complete the induction for L_{h+1}, condition on the first production: either it is $S \to$ hello, or it is $S \to S\,S$ and each subtree has to terminate within height h. Therefore (complete as a function of p):

$$L_{h+1} = \_\_\_\_\_ + \_\_\_\_\_ L_h^{\_\_\_\_}.$$

		2

$$L_{h+1} = 1 - p + pL_h^{2}.$$

1.d As $h \to \infty$, L_h converges to (complete showing steps)

$$\min\left\{1, \frac{1 - \_\_\_\_}{\_\_\_\_}\right\}.$$

Do you see a problem for some values of p? Which values and why?

		3

Denoting $L = \lim_{h\to\infty} L_h$ and $q = 1 - p$, we get $pL^2 - L + q = 0$, for which the roots are $(1 \pm \sqrt{1-4pq})/(2p)$. $\sqrt{1-4pq} = \sqrt{1-4p(1-p)} = \sqrt{1-4p+4p^2} = \sqrt{(1-2p)^2} = 1 - 2p$. So one case is $\frac{1-(1-2p)}{2p} = 1$ and the other case is $\frac{1+(1-2p)}{2p} = \frac{1-p}{p}$. Testing with a couple of specific values of p, it can be verified (also see, e.g., *The Convergence of Sequences Defined by Quadratic Recurrence-Formulae* by T. W. Chaundy and Eric Phillips, 1935) that L_h will converge to the smaller value, which is a problem if $p > 1/2$ — it means the PCFG defines an improper distribution. (But this may not matter for some inference tasks.)

2. We study various notions of graph proximity and how to encode them in node embeddings. We are given an unweighted, undirected adjacency matrix $\mathbf{A} \in \{0,1\}^{N\times N}$ betweeen N nodes.

Katz index: Let $0 < \beta < 1$ be a decay parameter. Suppose a path from node i to j has k edges. Then this path contributes β^k toward the Katz index of proximity between i and j. The overall Katz index is the sum of contributions from all paths.

Random walk with restart (RWR): The probability of walking from node u to a neighbor v is the reciprocal of d_u, the degree of u. Starting at i, at each step we reset to i with probability $0 < 1 - \alpha < 1$, and with probability α/d_u we walk to a neighbor of u. The proximity from u to v is the steady state visit probability of v.

Common neighbors (CN): The proximity between i and j is simply the number of common neighbors they have.

Adamic-Adar (AA): Like CN, except neighbor k common to i and j is counted with a weight $1/d_k$.

2.a Which of the above proximity measures is/are symmetric?

☐ Katz ☐ RWR ☐ CN ☐ AA

		2

Katz, CN and AA are symmetric.

2.b All four measures can be summarized as $\mathbf{S} = \mathbf{M}_g^{-1}\mathbf{M}_\ell \in \mathbb{R}_+^{N\times N}$, where S_{ij} is the proximity from i to j and $\mathbf{M}_g, \mathbf{M}_\ell$ are suitable matrix functions of A or other constant matrices. Fill the following table.

Proximity	$\mathbf{M}_g$	$\mathbf{M}_\ell$
Katz		
RWR		
CN		
AA		

		4

Proximity	$\mathbf{M}_g$	$\mathbf{M}_\ell$
Katz	$\mathbb{I} - \beta\mathbf{A}$	$\beta\mathbf{A}$
RWR	$\mathbb{I} - \alpha\mathbf{P}$	$(1-\alpha)\mathbb{I}/N$
CN	$\mathbb{I}$	$\mathbf{A}^2$
AA	$\mathbb{I}$	$\mathbf{A}\mathbf{D}\mathbf{A}$

Here $\mathbf{P}$ is a matrix derived from $\mathbf{A}$ with each column scaled to sum to 1. $\mathbf{D}$ is a diagonal matrix with D_{ii} being the reciprocal of the degree of node i, i.e., reciprocal of the ith row or column sum of $\mathbf{A}$.

2.c Now consider a **directed** acyclic graph (DAG), perhaps modeling types and entities belonging to those types in a knowledge base. Also focus on RWR proximity. If i is a type, then S_{ij} can be nonzero for which nodes j?

		1

Only nodes j reachable along directed paths from i can get a positive score.

2.d Continuing the DAG and RWR setting, if $v \subset u$ are two types, and

$$R(u) = \{i : S_{ui} > 0\} \quad \text{and} \quad R(v) = \{j : S_{vj} > 0\},$$

are two node index sets, is there a simple relation between $R(u)$ and $R(v)$? If so, what is it?

		1

$R(v) \subset R(u)$

2.e Depending on your previous answer, one might use S_{u*} as an "embedding" to represent node u. Give one reason why this may not be very useful or efficient. (Extra credit: propose a solution to the limitation.)

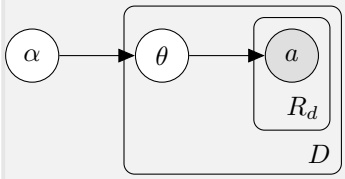
		1

To test for $\subset$ and $\in$, allocating a vector of N real numbers to represent each node takes excessive space and will usually overfit the training data. It would be better to reduce the dimensionality.

3. We develop some formulations for resolving author ambiguity from their mentions in papers. There are D papers and A authors. For each paper, a Dirichlet distribution with hyperparameters α is invoked to create a multinomial topic distribution with parameters θ. Note that here "topic" is not about the subject matter of the paper but a way to choose authors (although it is desirable to couple the two).

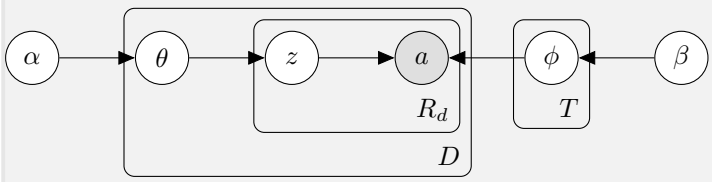
3.a In the first (preliminary) model, the dth paper lists R_d disambiguated author IDs. $\alpha \in \mathbb{R}^A$ and $\theta \in \Delta_A$ is in the A-dimensional multinomial simplex (i.e., $0 \leq \theta_a \leq 1$ and $\sum_a \theta_a = 1$). A multinomial distribution with parameters θ is invoked R_d times, each giving an author ID $z_r \in [1, A]$. Draw the plate diagram for this setting. Make sure you distinguish latent and observed variables.

		1



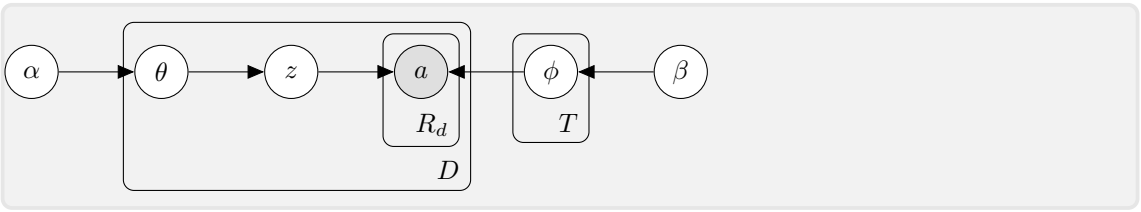
3.b Authors form collaborative teams and papers tend to repeat such teams. We create T "team generators" to model this effect, and change $\alpha \in \mathbb{R}^T$ and $\theta \in \Delta_T$. We also invoke a Dirichlet distribution (with hyperparameters $\beta \in \mathbb{R}^A$) T times to create T multinomial distributions $\phi_1, \ldots, \phi_T$, where each $\phi_t \in \Delta_A$. We expect each team generator ϕ_t to be sparse. ♣ Now, for each paper d, after sampling θ, for each author name $r = 1, \ldots, R_d$, we first sample (instead of an author) a team generator index $z \in [1, T]$, and then use ϕ_z to sample one author. Draw a plate diagram for this setting.

		3



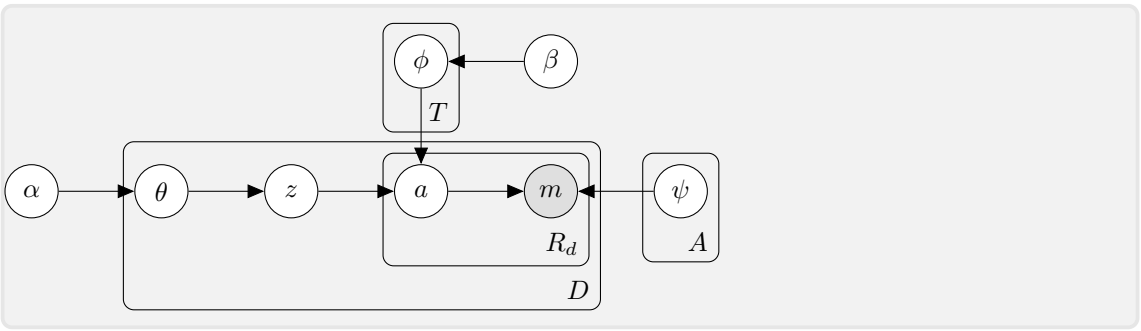
3.c It may be argued that choosing a different team for each author defeats the purpose of teams. Modify the process described after the ♣ above to generate only one team per paper. Write down the generative story in words, and draw a modified plate diagram.

		3

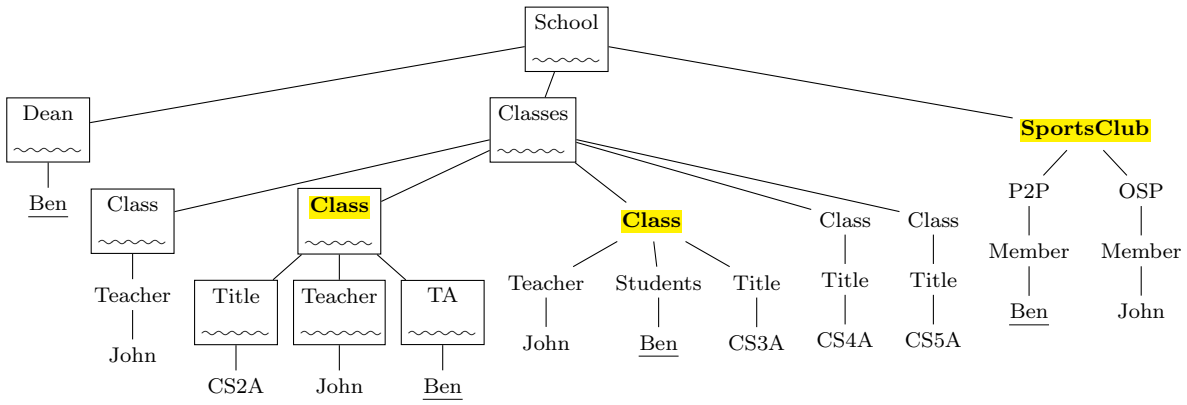


3.d Finally, the assumption that we see disambiguated author IDs a mentioned in papers is unrealistic. For simplicity, assume each author has M mention variants, like "John Smith", "J. Smith", "Smith, J. E." etc. Now a becomes latent, and m is the observed mention variable. Each author (ID) is therefore associated with a multinomial distribution with parameters $\psi_a \in \Delta_M$. Complete the plate diagram for this final generative story.

		2



4. Suppose we are given a large XML document (stored in RAM) that has a tree-structured tag hierarchy and nodes have associated text. A query is a set of words. A response to such a query is an enumeration of all minimal (i.e. as deep as possible) internal nodes such that the subtree under each node contains all query words.



For example, given query words Ben and John, the highlighted nodes (but not their ancestors) are the correct roots of answer subtrees. Suppose d is the maximum depth of the tree, the query keyword set is $w_1, \ldots, w_K$ and corresponding matching node sets are $S_1, \ldots, S_K$. I.e., the set of nodes that contain word w_k is S_k. For example, if $w_1 =$ Ben, then S_1 is shown as the four underlined (leaf) nodes; only three of them contribute toward minimal answer subtrees.

4.a lca(v_1, v_2) denotes the least common ancestor of nodes v_1, v_2. If each node includes a parent pointer, how would you compute lca(v_1, v_2) and how much time would it

take at most?

		1

See next answer. The time taken is $O(Kd)$.

4.b How would you extend the above solution to more than two nodes, i.e., $\text{lca}(v_1, \ldots, v_K)$?

		2

Let *count* be a map from a node ID to an integer. Assume parent(root) $= \perp$.

```
1: for k = 1, ..., K do
2:     n ← v_k
3:     while n ≠ ⊥ do
4:         count[n] += 1
5:         if count[n] = K then return n
6:         n ← parent(n)
```

4.c Given node sets $S_1, \ldots, S_K$, we say node v belongs to $\text{lca}(S_1, \ldots, S_K)$ if there exists a $v_k \in S_k$ for each $k = 1, \ldots, K$ such that $v = \text{lca}(v_1, \ldots, v_k)$. v is said to belong to the smallest lca or $\text{slca}(S_1, \ldots, S_K)$ if $v \in \text{lca}(S_1, \ldots, S_K)$ and v is not a proper ancestor of any $u \in \text{lca}(S_1, \ldots, S_K)$. A brute-force approach to compute $\text{lca}(S_1, \ldots, S_K)$ would take time proportional to (complete with some function of $K, d, \{S_k\}$ giving reasons):

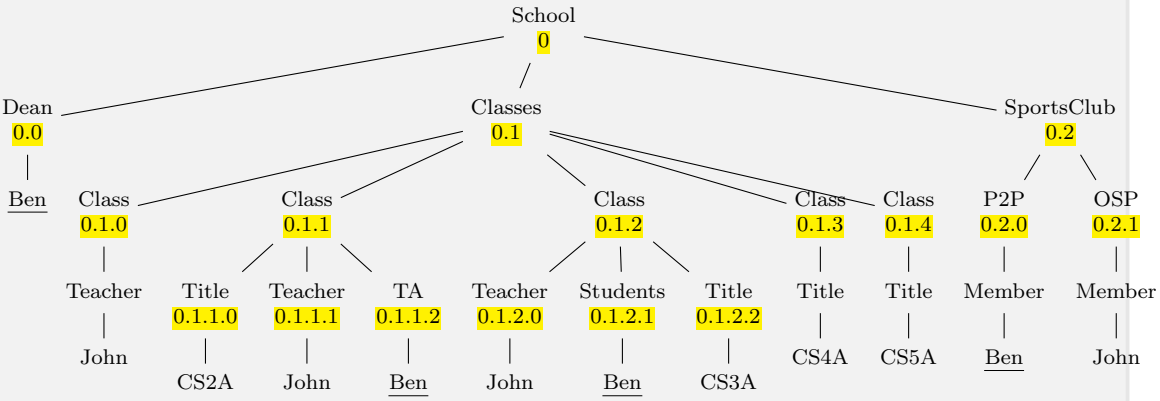
$$\underbrace{\sim\sim\sim}_{\text{function of } K}\ \underbrace{\sim\sim\sim}_{\text{function of } d}\ \prod_{k=1}^{K}\ \underbrace{\sim\sim\sim}_{\text{function of } S_k}$$

		3

Assuming "brute-force" means, for each S_k we choose, in turn, a node $v_k \in S_k$ where the kth word occurs, then we compute $\text{lca}(v_1, \ldots, v_K)$, the time needed is $Kd \prod_{k=1}^{K} |S_k|$.

4.d For our purposes, an alternative to parent pointers is the Dewey code: the root is numbered 0, its leftmost child 0.0, next child 0.1, etc. The first child of 0.1 would be 0.1.0. Fill Dewey codes into each node above that has a blank.

		2



We have not labeled nodes that are the single child of its parent.

4.e If nodes are presented as Dewey codes, how would you compute the lca of two nodes, and K nodes? Give brief informal pseudocode.

		2

Finding the lca of some nodes presented as Dewey codes is equivalent to computing the longest common prefix of those codes. Suppose the code of node v_k is $c_k = (c_{k0}, c_{k1}, \ldots)$ which has $|c_k|$ path components.

```
1: for i = 0, 1, ... do
2:     c ← c_{1i}
3:     for k = 2, ..., K do
4:         if c ≠ c_{ki} then return i − 1
```

The time taken remains $O(Kd)$.

4.f We also need an inverted list from words to sorted Dewey codes of nodes that directly contain the words. Write down the lists for John and Ben:

John: _____

Ben: _____

		1

John: 0.1.0, 0.1.1.1, 0.1.2.0, 0.2.1

Ben: 0.0, 0.1.1.2, 0.1.2.1, 0.2.0

4.g Complete the following pseudocode to report all slcas containing all query words. Note: The blank lines may need more than one action. Additional data structures may be needed for the test in line 9.

```
1: initialize stack s ← ()                          /* left is base, right is top, |s| is size */
2: open cursors on posting lists of all query words
3: while some cursor is not fully consumed do
4:     consume ____________ (smallest/largest) Dewey ID ℓ from some cursor
5:                          /* using lexicographic order over head items of cursors */
6:     p ← lca(____________, ____________)
7:     while |s| > ____________ do
8:         t ← pop(s)
9:         if subtree of t contains all query words then
10:            ____________________________________________________________
11:        else
12:            ____________________________________________________________
13:    ____________________________________________________________
14: ____________________________________________________________
```

		4

In line 4, whatever choice is made, will also work on a laterally inverted tree (imagine executing the code from the other side of the paper), so the choice does not matter, i.e., either smallest or largest is correct. Following one of the two orders is important, though.

```
1: initialize stack s ← ()
2: open cursors on posting lists of all query words
3: while some cursor is not fully consumed do
4:     consume lexicographically smallest, say Dewey ID ℓ from some cursor
5:     p ← lca( s , ℓ )
6:     while |s| > |ℓ| do
7:         t ← pop(s)
8:         if subtree of t contains all query words then
9:             add t to result pool
10:        else
11:            WordSet(top(s)) ← WordSet(top(s)) ∪ WordSet(t)
12:    push all popped ts back on s last-popped first-in
13: add to result pool the longest prefix of s that contains all query words
```

Given a Dewey ID, we need to access the set of query words contained in the node corresponding to the ID. When ℓ is first retrieved from a cursor, we can attach the matching query keyword. Any popping action transfers all present keywords to the parent node on stack. For an lca operation, we take the union of the keyword sets of the input nodes.

Total: 41