ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prof. Ryan Cotterell

Course Assignment Episode 2

14/12/2020 - 08:26h

Question 1: Distributions over Palindromes (15 pts)

In this problem, we will consider finding palindromic subsequences of a string. Let Σ be an alphabet[1] of characters. Suppose $\mathbf{x} \in \Sigma^*$ (where $*$ denotes the Kleene closure) is a string. We denote a subsequence as $\mathbf{y} = \mathbf{x}_{k(|\mathbf{x}|)}$ where $k(|\mathbf{x}|)$ is an ordered (strictly increasing) subset of $\{1, \ldots, |\mathbf{x}|\}$ and $\mathbf{x}_{k(|\mathbf{x}|)}$ is the (ordered) sequence $\{x_i\}_{i \in k(|\mathbf{x}|)}$. We denote the *set* of all palindromic subsequences of $\mathbf{x}$ by $\mathcal{P}(\mathbf{x})$.[2] In contrast to a substring, a subsequence need not be contiguous. We call a subsequence palindromic, or, simply, a palindrome if $\mathbf{y} = \mathbf{y}^{\mathrm{R}}$, i.e. if the subsequence $\mathbf{y}$ is the same when read forwards and backwards.

(a) Give a dynamic-programming algorithm that returns the maximum-length palindromic subsequence $\mathbf{x}$. Analyze its time and space complexity **A Word to the Wise:** Yes, this is a very common coding interview question. And, yes, you can just Google the answer. Please don't—you're only cheating yourself out of an education.

(b) Reformulate the dynamic program you derived in (a) to use semiring notation. Revisit the slides from Lecture 6 and determine which semiring is used in the special case you found in (a).

(c) Now, exploit your semiring generalization to give an algorithm that counts the number of palindromes in $\mathbf{x}$. Which semiring gives this result? Also, explain which semiring gives you a "recognition" algorithm, i.e. an algorithm returns "yes" if there is a palindrome in $\mathbf{x}$ and "no" otherwise.

(d) Give an algorithm with the same runtime complexity as the one in (a) that computes the average length of all palindromes in $\mathbf{x}$. Which semiring(s) did you use?

(e) Give pseudocode for a prioritized version of your algorithm in (a). **Hint**: refer to Dijkstra's algorithm for inspiration.[3] Analyze the time and space complexity of your algorithm.

(f) This class is about probabilistic modeling over structures! So, let's model some palindromes—they are language after all. For any palindromic subsequence $\mathbf{y} = \mathbf{x}_{k(|\mathbf{x}|)}$, suppose you are given a score function $\text{score}(\mathbf{y}, \mathbf{x}) : \Sigma^* \times \Sigma^* \to \mathbb{R}$. Now, consider a probabilistic model over all palindromes in $\mathbf{x}$ defined as follows:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} \exp\{\text{score}(\mathbf{y}, \mathbf{x})\} \tag{1}$$

[1] An alphabet is a non-empty, finite set.

[2] Note that this set may be exponentially large (in the size of $|\mathbf{x}|$).

[3] We do expect competence in undergraduate algorithms as this is a graduate-level computer science course. However, if you do not know Dijkstra's algorithm, please feel free to consult a friend in the course with a computer science background or a an algorithms textbook.

where we have the normalizer:

$$Z = \sum_{\mathbf{y}' \in \mathcal{P}(\mathbf{x})} \exp\{\text{score}(\mathbf{y}', \mathbf{x})\} \tag{2}$$

Give a slow, naïve algorithm to compute Z; analyze its time and space complexity. Now, explain how to construct a decomposable score$(\mathbf{y}, \mathbf{x})$ such that you can compute Z in the same runtime as your algorithm given in (a). **Hint:** There is more than one way to construct a decomposable score function that will work out—try to be creative and think about how to design a fun decomposable function.

Question 2: Am I a Semiring? (10 pts)

(a) Define the set $A = \{a+b\sqrt{3} \mid a, b \in \mathbb{Z}\}$. Can you find a $\mathbf{0}$ and $\mathbf{1}$ such that $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ is a semiring? Prove your answer.

(b) Is $\langle \mathbb{R}, \max, \times, 0, 1\rangle$ a semiring? Prove your answer.

(c) Let A be a finite set. Is $\langle 2^A, \cup, \cap, \emptyset, \emptyset\rangle$ a semiring? Prove your answer.

Question 3: CRFs as Context-Free Grammars (15 pts)

In this question, we ask you to show how to encode a conditional random field (CRF) for part-of-speech tagging as a weighted context-free grammar (CFG). Then, we ask you to manipulate the CKY algorithm to recover the forward algorithm presented in Lecture #6.

Consider the following grammar:

$$\begin{aligned} S &\to (A \mid B)\, S \\ S &\to (A \mid B) \\ A &\to a \\ B &\to b \end{aligned}$$

(a) Describe as concisely as possible the set of strings this grammar can generate.

(b) For any string $\mathbf{s} = s_1, \ldots, s_N$ in this set, what is the maximum number of parse trees under this grammar that yield $\mathbf{s}$?

(c) In this question, we denote spans as $[i, j]$; the length of $[i, j]$ is $j - i$. We say a span is admissible under the grammar if we can build a complete parse tree with a constituent from i to j. In this setting, a complete tree means we have built a span $[1, N + 1]$ rooted at non-terminal S. Suppose we are parsing a string of length N, what are the spans admissible by the grammar under consideration? In your answer, please enumerate all spans of length ℓ for every $\ell \in \{1, \ldots, N\}$.

(d) Modifying the pseudocode for CKY given in class, show that CKY on this grammar runs in $o(n^3)$ time. Give a tight worst-case runtime.

(e) Given a set of part-of-speech tags $\mathcal{T} = \{T_1, \ldots, T_{|\mathcal{T}|}\}$ and a set of words $\mathcal{W} = \{w_1, \ldots, w_{|\mathcal{W}|}\}$, construct a grammar (without weights) of a similar structure to the grammar under consideration that encodes a first-order conditional random field (CRF) for part-of-speech tagging, i.e, a CRF that scores part-of-speech bigrams. You should assume any word in $\mathcal{W}$ may take on any part of speech. **Hint**: The answers as you put down for (b) and (c) above should also be true for your grammar.

(f) Now show how to construct a score function score$(\cdot)$ for your grammar such that the unnormalized probability of a tree

$$p(\text{tree} \mid s_1, \ldots, s_N) \propto \prod_{\pi \in \text{tree}} \text{score}(\pi) \tag{3}$$

where π is a production in the grammar you designed, is the same as the unnormalized probability under a CRF:

$$p(t_1, \ldots, t_N \mid s_1, \ldots, s_N) \propto \prod_{n=1}^{N} \psi(t_n, t_{n-1}) \cdot \phi(t_n, s_n) \tag{4}$$

Note that we define $t_0 := \mathsf{BOS}$ as a beginning-of-sequence tag. Your answer for score $(\cdot)$ should be a function of ψ and ϕ.

(g) Your answer to (f) allows us to use CKY to compute the normalizer Z. Give a tight worst-case runtime bound for CKY as a function of number of part-of-speech tags $|\mathcal{T}|$ and the length of the sentence N under this specific grammar.

Question 4: Minimum Edit Distance as a Finite-State Machine (10 pts)

(a) For this problem, we define minimum edit distance as the minimum number of insertions, substitutions or deletions required to convert one string into another.

(i) Design a weighted finite-state acceptor (WFSA) that accepts (only) all strings of edit distance ≤ 1 from the string "ilovenlp."

(ii) Now consider a WFSA that accepts only strings of edit distance $\leq d$ from some target string $\mathbf{s} = s_1, \ldots, s_N$. What is the minimal number of states required in this WFSA as a function of d and N?

(b) Consider an n-gram language model, as in Lecture 5.

(i) Design a WFSA for a trigram language model that accepts only strings with probability > 0 under the model. A transition function δ, an initial weight function λ and a final weight function ρ should be defined. We expect edge cases at the beginning and end of input to be properly handled.

(ii) How many states does the n-gram language model have as a function of n and $|V|$ where V is our vocabulary?