COMP90042 Web Search & Text Analysis

Workshop Week 9

Zenan Zhai

May 7, 2019

University of Melbourne

Outline

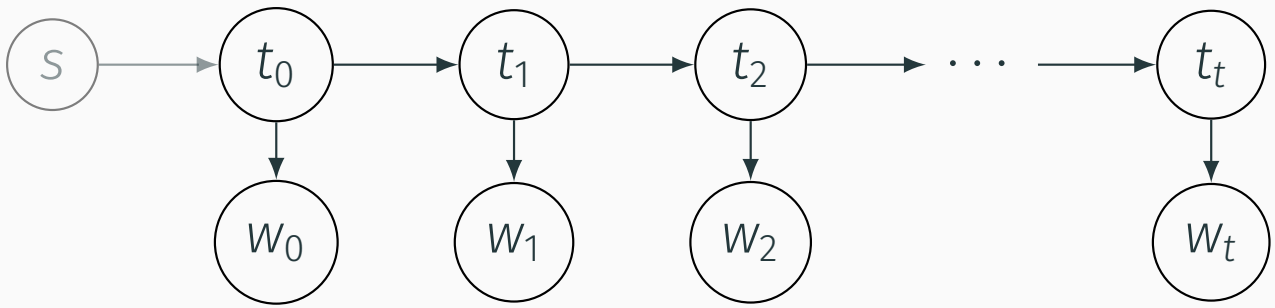
Hidden Markov Model

- Concept
- Training
- Inference

Computation Theory

- Regular Language
- Finite State Automata
- Applications

Hidden Markov Model



Probabilistic graphic model

- Hidden: Tags are not observed.
- Markov assumption
  - Transition: Current tag only depends on previous tag.
  - Emission: Current word only depends on current tag.

Parameters

- $A : P(t_i|t_{i-1})$ Transition Matrix
- $B : P(w_i|t_i)$ Emission Matrix
- $\pi : P(w_1|s)$ Initial States

HMM - Training

Maximum Likelihood Estimation (MLE)

$$P(t_i|t_{i-1}) = \frac{count(t_i, t_{i-1})}{count(t_{i-1})}$$

$$P(w_i|t_i) = \frac{count(t_i, w_i)}{count(t_i)}$$

Exercise

1. silver-JJ wheels-NNS turn-VBP
2. wheels-NNS turn-VBP right-JJ
3. right-JJ wheels-NNS turn-VBP

How to estimate A, B, π ?

HMM - Inference

$$\hat{t} = argmax_t \prod_{i-1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Viterbi Algorithm

- Dynamic programming
  - Reduce problem size by decomposing to simpler sub problems.
- Keep track of
  - Best sub-sequence probabilities α
  - Backward pointers

Exercise Worksheet 9 - question 2

- Most likely tag sequence given A, B, π

Exercise - Viterbi Algorithm

A	JJ	NNS	VBP
JJ	0.4	0.5	0.1
NNS	0.1	0.4	0.5
VBP	0.4	0.5	0.1

A	silver	wheel	turn
JJ	0.8	0.1	0.1
NNS	0.3	0.4	0.3
VBP	0.1	0.3	0.6

π[JJ, NNS, VBP] = [0.3, 0.4, 0.3]

Outline

Hidden Markov Model

- Concept
- Training
- Inference

Computation Theory

- Regular Language
- Finite State Automata
- Applications

Regular Language

Definition

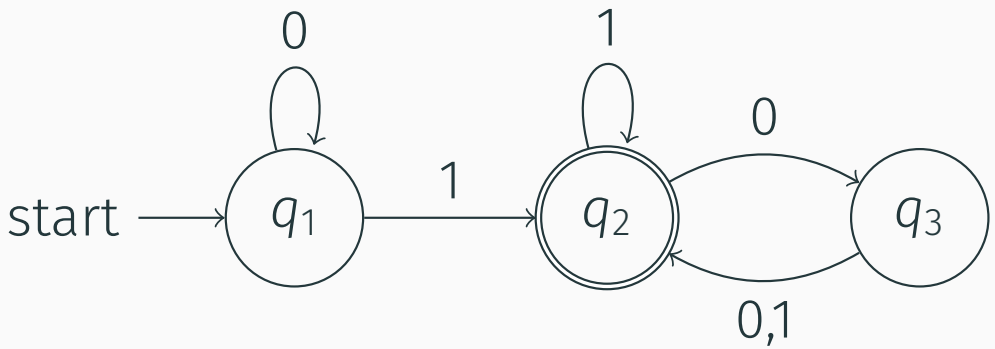
- A language is called a regular language if some finite automaton recognizes it.

Regular languages can be written in regular expressions.

Properties

- Closed under concatenation/union
- Closed under intersection
- Closed under negation

Finite State Automata



A finite state automata is a 5-tuple $M_1 = (Q, \Sigma, \delta, q_1, F)$

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. δ described as transitions
4. q_1 is the start state, and
5. $F = q_2$.

What language does M_1 accept?

Weighted FSA

Adding weight to start/final state and transition in FSA

- $\lambda : Q \rightarrow \mathbb{R}$ Assign weight to initial state
- $\rho : Q \rightarrow \mathbb{R}$ Assign weight to final state
- $\delta : (Q, \Sigma, Q) \rightarrow \mathbb{R}$ Assign weight to transitions

Weighted FSA for scoring given sequence $\pi = t_1, t_2, t_3, \cdots, t_N$

- $S(\pi) = \lambda(t_0) + \sum_{i=1}^{N}(t_i) + \ \rho(t_N)$
- Dijsktra algorithm for shortest path: $O(|V|log|V| + |E|)$

N-gram Language model as weighted FSA

- $\lambda(q_i) = -logP(w_1 = i|start)$
- $\rho(q_i) = -logP(end|w_N = i)$
- $\delta(q_i, w, q_j) = \begin{cases} -logP(w_m = j|w_{m-1} = i), & \text{if } w = j \\ \infty, & \text{otherwise} \end{cases}$

Finite State Transducer

A finite transducer T is a 6-tuple (Q, Σ, Γ, I, F, δ) such that:

1. Q Set of all states
2. Σ Input alphabet
3. Γ Output alphabet
4. I Set of initial states
5. F Set of final states
6. $\delta \subseteq Q \times \{\Sigma \cup \epsilon\} \times \{\Gamma \cup \epsilon\} \times Q$ Transitions

Added output to Weighted FSA, can be used for ”translations”.

Questions:

- Are all languages regular?
- Comparing to a von Neumann architecture machine, what is missing in FSA?