Research Plan

Myeong-Jang Pyeon Yu-Min Kim Hak-Su Lim

September 13, 2018

Team Name: Practical PFA (PPFA)
Advisor: Prof. Yo-Sub Han
TA: Marco Cognetta

1 Probabilistic Finite-State Automata

There are DFA/NFA that represent the accepting model to verify the acceptance of the strings, but if the probabilities is added to the transitions of the automata [6,7]. A=$\langle\Sigma, Q, \mathbb{I}_\mathbb{P}, \mathbb{F}_\mathbb{P}, \delta_\mathbb{P}\rangle$ this is called PFA and represents the string generating model. The PFA satisfies the following conditions.

- Q is a finite set of states; these will be labelled $q_1, \ldots, q_{|Q|}$ unless otherwise stated
- Σ is the alphabet
- $\mathbb{I}_\mathbb{P}$: Q $\rightarrow$ $\mathbb{Q}^+ \cap$ [0, 1] *(initial-state probabilities)*
- $\mathbb{F}_\mathbb{P}$: Q $\rightarrow$ $\mathbb{Q}^+ \cap$ [0, 1] *(final-state probabilities)*
- $\delta_\mathbb{P}$: Q $\times$ ($\Sigma \cup \{\lambda\}$) $\times$ Q $\rightarrow$ $\mathbb{Q}^+$ *is a transition function; the function is complete:* $\delta_\mathbb{P}$ *(q,a,q') = 0 can be interpreted as "no transition from* q *to* q' *labelled with a". We will also denote* (q, a, q', P) *instead of* $\delta_\mathbb{P}(q, a, q') = P$ *where* P *is a probability.*

$\mathbb{I}_\mathbb{P}$, $\delta_\mathbb{P}$ and $\mathbb{F}_\mathbb{P}$ are functions such that:

$$\sum_{q \in Q} \mathbb{I}_\mathbb{P}(q) = 1,$$

$$\textit{and } \forall q \in Q, \mathbb{F}_\mathbb{P}(q) + \sum_{a \in \sum \cup \{\lambda\}, q' \in Q} \delta_\mathbb{P}(q, a, q') = 1.$$

If the PFA generates a string, the string has a probability of being generated by the PFA. We can also find the distributions in which the strings are generated through string sampling. The distribution satisfies the following conditions.

-$Pr_D(x)$ is the probability of a string $x \in \Sigma^*$ under the distribution D
-$\sum_{x \in \Sigma^*} Pr_D(x) = 1$.
-$\forall q \in Q$, $\mathbb{F}_{\mathbb{P}}(q) + \sum_{a \in \Sigma \cup \{\lambda\}, q' \in Q} \delta_{\mathbb{P}}(q, a, q') = 1$.

If λ-transitions are added to the PFA it is called a λ-PFA. The PFA can be represnted with graph. Figure 1 is an example of the graph of the PFA [2]. The numbers on the left of the state name are initial probabilities and the numbers on the right of the state name are final probabilities. The number on the transitions are transition probabilities. For example, the probability of parsing string "ab" with this PFA is $Pr_A(ab)$ that

$$\begin{aligned}
Pr_A(ab) &= \mathbb{I}_{\mathbb{P}}(q_1) \cdot \delta_{\mathbb{P}}(q_1, a, q_2) \cdot \delta_{\mathbb{P}}(q_2, b, q_4) \cdot \mathbb{F}_{\mathbb{P}}(q_4) \\
&\quad + \mathbb{I}_{\mathbb{P}}(q_2) \cdot \delta_{\mathbb{P}}(q_2, a, q_3) \cdot \delta_{\mathbb{P}}(q_3, b, q_3) \cdot \mathbb{F}_{\mathbb{P}}(q_3) \\
&\quad + \mathbb{I}_{\mathbb{P}}(q_2) \cdot \delta_{\mathbb{P}}(q_2, a, q_3) \cdot \delta_{\mathbb{P}}(q_3, b, q_4) \cdot \mathbb{F}_{\mathbb{P}}(q_4) \\
&= 0.4 \cdot 0.5 \cdot 0.4 \cdot 0.3 \\
&\quad + 0.6 \cdot 0.5 \cdot 0.2 \cdot 0.4 \\
&\quad + 0.6 \cdot 0.5 \cdot 0.4 \cdot 0.3 \\
&= 0.084
\end{aligned}$$

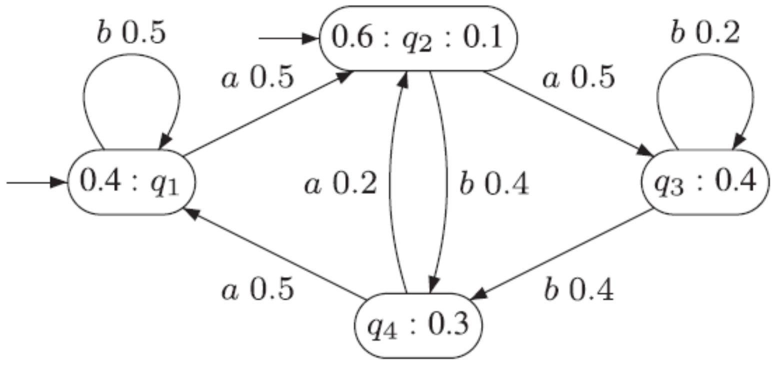

Figure 1: Graphical representation of a PFA [2]

2 Problems

2.1 Related Problems

In probabilistic automata theory, an important question is what the most probable string is. Without any constraint, the problem is called as the *consensus string* problem. The formal definition is as below.
Name: Consensus string (Cs)

Instance: A probabilistic machine $\mathcal{M}$
Question: Find in Σ^* a string x such that $\forall y \in \Sigma^*, Pr_{\mathcal{M}}(x) \geq Pr_{\mathcal{M}}(y)$.

This problem can be modified as a decision problem. It is called as the *most probable string* problem, defined as below.
Name: Most probable string (MPS)
Instance: A probabilistic machine $\mathcal{M}$ and $p \geq 0$
Question: Find in Σ^* a string x such that $Pr_{\mathcal{M}}(x) > p$.

If an extra constraint for the length of strings is added, it is called as the *bounded most probable string* problem, which is defined as below.
Name: Bounded most probable string (BMPS)
Instance: A probabilistic machine $\mathcal{M}$, $p \geq 0$, and an integer b
Question: Find in $\Sigma^{\leq b}$ a string x such that $Pr_{\mathcal{M}}(x) > p$.

2.2 Our Problem

In this project, we will solve another problem. In our setting, we are given a string so that the solution space is reduced to the set of string within $k-$distance with the given string. We call it as the *most probable string within $k-$distance* problem and the formal definition is described as below.
Name: Most probable string within $k-$distance
Instance: A probabilistic machine $\mathcal{M}$, $p \geq 0$, a string w, and $k \in \mathbb{N}$
Question: Find in Σ^* a string x such that $Pr_{\mathcal{M}}(x) > p$ and dist(x, w)$\leq k$.
Note that the solution space is a subset of $\Sigma^{\leq |w|+k}$. So, it is not necessary to add the constraint for the length of strings.

3 Previous Research

The problem of finding the most probable string (MPS) for a distribution generated by a probabilistic finite-state automaton is NP-hard. [3, 4] Therefore the decision problem of it, which determines whether there exist a string whose probability on this machine is bigger than a given probability p, is also NP-hard. The one main reason is that there is no bound on the string. The most probable string can be of exponential length. [3, 4]

We focus on the bounded decision problem of MPS(BMPS). In that case, there is an additional input integer b that bounds the length of the string lower than or equal to b. There is a pseudo-polynomial exact algorithm to solve BMPS. [3, 4]

The problem we're interested in is MPS where distance within k (k-MPS). Note that k-MPS is inherently a kind of BMPS because the length of string is bounded by edit(or hamming) distance k. There is an exact algorithm suggested by Marco Cognetta that solve this problem in pseudo-polynomial time when k is lower than or equal to 1. However this algorithm cannot solve the problem in pseudo-polynomial time when k is bigger than 1. We'll brainstorm how to efficiently solve k-MPS when k is bigger than 2.

4 Approaches

As we mentioned above, there is no efficient algorithm to solve k-MPS. Therefore we expect to design such an efficient algorithm for k-MPS. What we have to do first is obtaining a PFA $\mathcal{M}^{'}$ that generates strings w which can be generated by the original PFA $\mathcal{M}$ and satisfy $\text{dist}(w,\, w_0) \leq k$. If we successfuly obtain $\mathcal{M}^{'}$, then we can reduce the problem to an instance of MPS. However, the existing algorithm for MPS is psudo-polynomial, not strictly efficient. In fact, the time compleity of it is proportional to $\frac{1}{p}$. Therefore, there are some challenges to do this,

- **Obtaining $\mathcal{M}^{'}$.** How to obtain $\mathcal{M}^{'}$ in polynomial time.
- **MPS on $\mathcal{M}^{'}$.** We have two options, one is to use the existing pseudo-polynomial approximation algorithm [4], and the other is to invent a new polynomial approximation algorithm.

5 Contribution

If we solve k-MPS, we expect below applications [1],

- **Spell Correction.** Since PFA is a generative model which generates distribution, we can obtain a PFA over a person's chat, for instance, from Kakao Talk. We expect to correct wrong sentences given the PFA and the algorithm that solve k-MPS. For example, we can easily correct a sentence "I am a toy", which is obviously wrong sentence, to "I am a boy".
- **Image/Audio Reconstruction.** If we have a PFA on some image dataset like ImageNet, we can reconstruct(or restore) a broken image to normal one.

6 Implementation

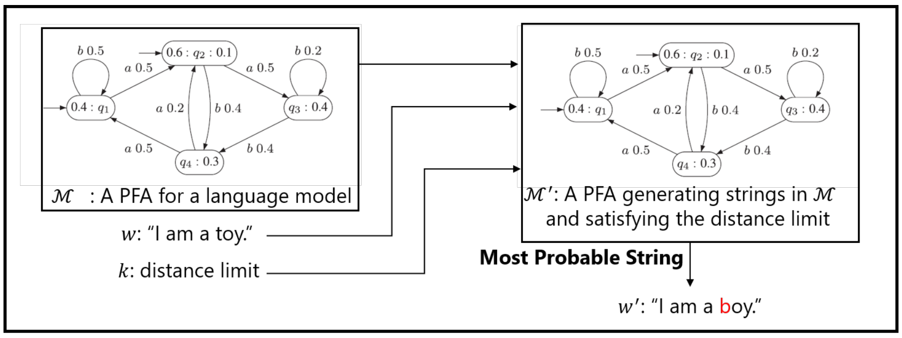

Figure 2: Graphical representation of Spell Correction using PFA and k-MPS

As an application of k-MPS, we will implement a spell correction algorithm. In order to do this, we first design a PFA framework. On the framework we learn a PFA for language modeling using spectral learning [5]. The overall architecture for this implementation is illustrated in Figure 2.

7 Role Division

- **Yu-Min Kim.**
  - Design and implement the algorithm
  - Implement forward steps in the PFA framework
- **Myeong-Jang Pyeon.**
  - Design and implement the algorithm
  - Implement the learning algorithm for PFA
- **Yu-Min Kim.**
  - Design and implement the algorithm
  - Implement the GUI in the PFA framework

8 Schedule

Below is our schedule of this project.

Week	Schedule
Week 1	Survey on previous research
Week 2	Proposal
Week 3	Design the algorithm for MPS (1)
Week 4	Design the algorithm for MPS (2)
Week 5	Design the algorithm for MPS (3)
Week 6	Interim 1
Week 7	Implement the PFA framework (1)
Week 8	Implement the PFA framework (2)
Week 9	interim 2
Week 10	Extract the PFA for the language model (1)
Week 11	Extract the PFA for the language model (2)
Week 12	Integrate all modules
Week 13	Final

References

[1] Marco Cognetta and Yo-Sub Han. Online stochastic pattern matching. In *International Conference on Implementation and Application of Automata*, pages 121–132. Springer, 2018.

[2] Colin De La Higuera. *Grammatical inference: learning automata and grammars.* Cambridge University Press, 2010.

[3] Colin De La Higuera and José Oncina. Computing the most probable string with a probabilistic finite state machine. In *Proceedings of the 11th International Conference on Finite State Methods and Natural Language Processing*, pages 1–8, 2013.

[4] Colin De La Higuera and José Oncina. The most probable string: an algorithmic study. volume 24, pages 311–330. Oxford University Press, 2013.

[5] Colin De La Higuera and José Oncina. Spectral learning techniques for weighted automata, transducers, and grammars. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[6] Enrique Vidal, Franck Thollard, Colin De La Higuera, Francisco Casacuberta, and Rafael C Carrasco. Probabilistic finite-state machines-part i. volume 27, pages 1013–1025. IEEE, 2005.

[7] Enrique Vidal, Frank Thollard, Colin De La Higuera, Francisco Casacuberta, and Rafael C Carrasco. Probabilistic finite-state machines-part ii. volume 27, pages 1026–1039. IEEE, 2005.