1 Overview

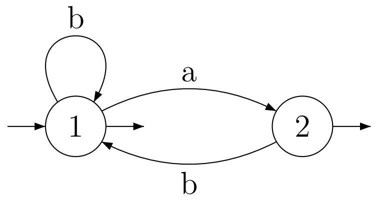
1. What are finite-state machines?
2. How do they generalize n-gram models?
3. N-gram models are one example of probabilistic deterministic finite-state machine (PDFMs).
4. PDFM-based language models can be trained with basically the same techniques, and they can express various types of long-distance dependencies.
5. Multiple PDFMs can be combined as independent factors to provide a joint distribution over string space, and they too can be trained with basically the same techniques [Shibata and Heinz, 2019].
6. Probabilistic Non-determinisistic FMs (PNFMs) are even more expressive than PDFMs, and there are learning strategies, but no guarantees.
7. Probabilistic Context-Free Grammars (PCFGs) are even more expressive than PNFMs.

$$\text{n-gram model} \subsetneq \text{PDFM} \subsetneq \text{PNFM} \subsetneq \text{PCFG}$$

2 Finite-State Machines

Finite-State Machines process *strings* (or more generally any recursive data structure).

2.1 Acceptors



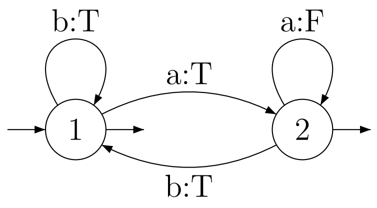
Is there a path through the machine for these strings?

1. bababa
2. babaab
3. bbabba
4. baabaa

Transducers

Multiplying along paths

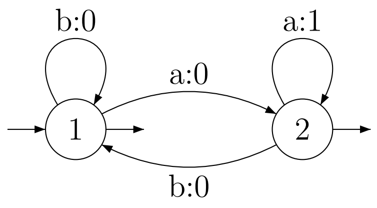
Boolean



Draw paths for the following strings which also show the outputs. Multiply the outputs with *conjunction*.

1. bababa
2. babaab
3. bbabba
4. baabaa

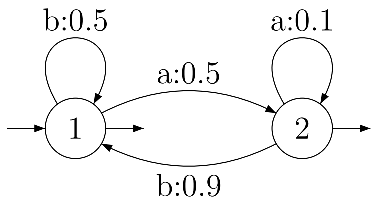
Counting



Draw paths for the following strings which also show the outputs. Multiply the outputs with *addition*.

1. bababa
2. babaab
3. bbabba
4. baabaa

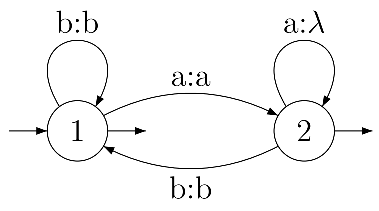
Probability



Draw paths for the following strings which also show the outputs. Multiply the outputs with *multiplication*.

1. bababa
2. babaab
3. bbabba
4. baabaa

Strings



Draw paths for the following strings which also show the outputs. Multiply the outputs with *concatenation*. Note λ denotes the empty string.

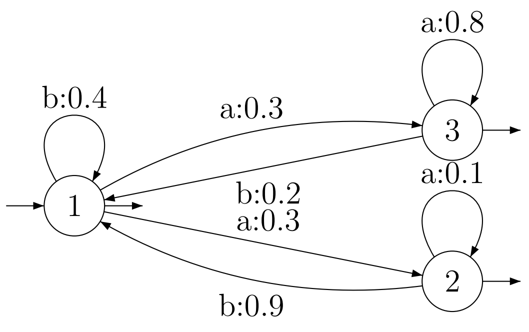
1. bababa
2. babaab
3. bbabba
4. baabaa

Summing across paths

The above acceptors were *deterministic*. That is, at each state upon reading a letter, there was at most one tranistion arc to take. Consequently, there is at most one path for each string.

The ones below are *non-deterministic*, which means there can be more than one path for each string. In this case, you have to take *all* the paths and *sum* them up.

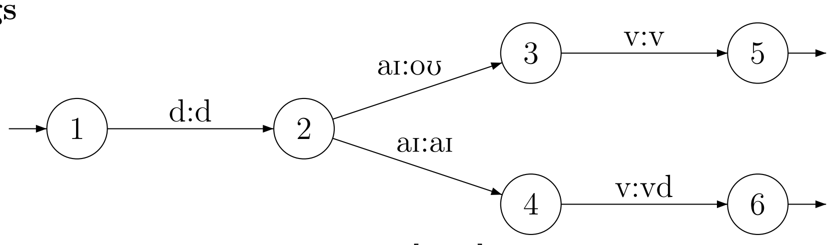
Probability



Draw all the paths for the string *baa* which also show the outputs. As before, along a path, multiplication is concatenation. Across paths, summing is *addition*.

So what is the probability of *baa* according to this transducer?

Strings



Draw all the paths for the string [daɪv] which also show the outputs. As before, along a path, multiplication is concatenation. Across paths, summing is *union*.

So what is the output of this transducer given the input [daɪv] ‘dive’?

Semirings

A semiring is a set K over which two binary operations are defined $\oplus, \otimes$, called 'addition/plus' and 'multiplication/times' which contains elements 1 and 0 with the following properties satisfied: if $x, y, z \in K$ then

- $x \oplus y, x \otimes y \in K$ (closure under $\oplus$ and $\otimes$)
- $x \oplus y = y \oplus x$ ($\oplus$ is commutative)
- $0 \oplus x = x \oplus 0 = x$ (0 is the identity for $\oplus$)
- $1 \otimes x = x \otimes 1 = x$ (1 is the identity for $\otimes$)
- $0 \otimes x = x \otimes 0 = 0$ (0 is an annihilator for $\otimes$)
- $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$. ($\otimes$ right distributes over $\oplus$)

Name	K	$\oplus$	$\otimes$	0	1
Boolean	$\{\texttt{true}, \texttt{false}\}$	$\vee$	$\wedge$	`false`	`true`
Natural	$\mathbb{N}$	$+$	$\times$	0	1
Viterbi	$[0, 1]$	`max`	$\times$	0	1
Tropical	$\mathbb{R} \cup \{\infty\}$	`min`	$+$	∞	0
Language	$\mathcal{P}(\Sigma^*)$	$\cup$	$\cdot$	$\emptyset$	$\{\lambda\}$

If the outputs of a finite-state machine M belong to K then M computes a function $f_M : \Sigma^* \to K$ as follows.

$$f_M(w) = \bigoplus_{w\text{-paths } p \in M} \bigotimes_{\text{transitions } t \in p} (\text{output of } t)$$

3 N-gram models as Probabilistic Finite-state Machines

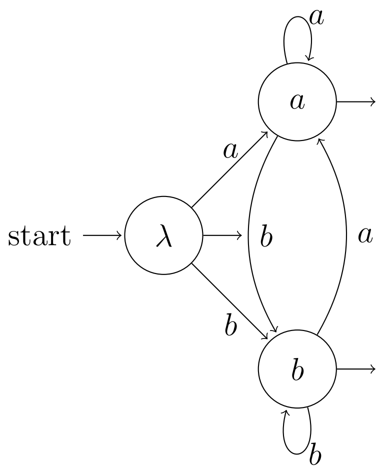


Figure 1: Example: Bigram Model with $\Sigma = \{a, b\}$.

The parameters of the n-gram model are the transitions in a particular finite-state machine where the states represent the *recent* history.

The MLE is obtained by passing the data through the deterministic finite-state machine and normalizing [Vidal et al., 2005a,b]. This is true for any deterministic finite-state model, not just ones representing n-gram models.

For example, suppose $D = \{ab, aabb\}$. Then for the machine above, we would determine the following.

Parameters	counts	normalized
$\theta_{\rtimes a}$	2	1
$\theta_{\rtimes b}$	0	0
$\theta_{\rtimes \ltimes}$	0	0
θ_{aa}	1	1/3
θ_{ab}	2	2/3
$\theta_{a\ltimes}$	0	0
θ_{ba}	0	0
θ_{bb}	1	1/3
$\theta_{b\ltimes}$	2	2/3

If the finite-state machines have different structures then different kinds of patterns, including certain non-local effects, can also be modeled [Heinz and Rogers, 2010].

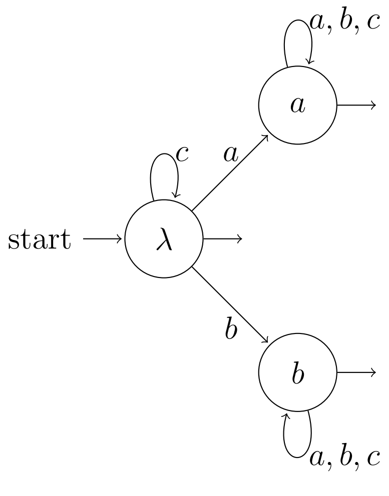


Figure 2: Example: Strictly Piecewise Model

4 Unknown Structure

In the above examples, the structures of the machines are known. What if they are unknown?

1. There are different kinds of approaches based on whether the models are assumed to be deterministic or not.

2. If they are deterministic then the state-merging approaches can be used (see the algorithm ALLEGRIA) Carrasco and Oncina [1994,?], de la Higuera [2010].

3. If they are non-deterministic, the idea is to assume a "super-structure" and let the strategy to non-determinism take care of it (see below).

5 Non-determinism

Non-determinism makes estimation of the probabilities complicated. This is because we do not know which paths in the underlying machine have been traversed by a given observation. This is the problem of "hidden states" or "latent factors".

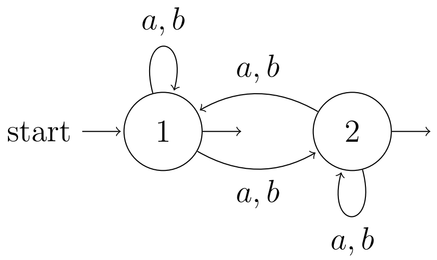


Figure 3: Example: Fullly connected, non-deterministic 2-state finite-state machine.

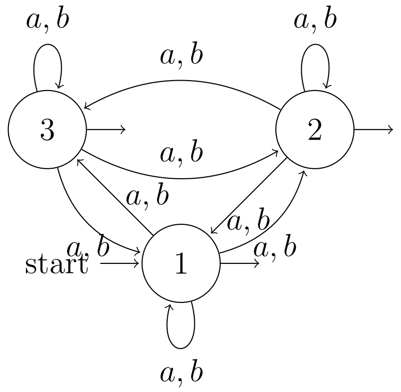


Figure 4: Example: Fullly connected, non-deterministic 3-state finite-state machine.

If there are n states and $|\Sigma|$ letters then there are $n^2|\Sigma|$ parameters.

References

Rafael C. Carrasco and José Oncina. Learning stochastic regular grammars by means of a state merging method. In *Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante, Spain, September 21-23, 1994, Proceedings*, pages 139–152, 1994.

Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars.* Cambridge University Press, 2010.

Jeffrey Heinz and James Rogers. Estimating strictly piecewise distributions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 886–896, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Chihiro Shibata and Jeffrey Heinz. Maximum likelihood estimation of factored regular deterministic stochastic languages. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 102–113, Toronto, Canada, 18–19 July 2019. Association for Computational Linguistics.

Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. Probabilistic finite-state machines-part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025, 2005a. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2005.147.

Enrique Vidal, Frank Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. Probabilistic finite-state machines-part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1026–1039, 2005b. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2005.148.