Fun with weighted FSTs

Informatics 2A: Lecture 18

Shay Cohen

School of Informatics
University of Edinburgh

29 October 2018

POS Tag Ablation It is also not well explored what word features are being used by the encoders. To understand which classes of words were most important we ran an ablation study, selectively removing nouns, verbs (including participles and auxiliaries), adjectives & adverbs, and function words (adpositions, determiners, conjunctions). All datasets were automatically tagged using the spaCy part-of-speech (POS) tagger[9]. The em-

Kedzie et al. (2018) - “Content Selection in Deep Learning Models of Summarization”

Testing which word classes are important for summarizing a document

Requires a part-of-speech tagger

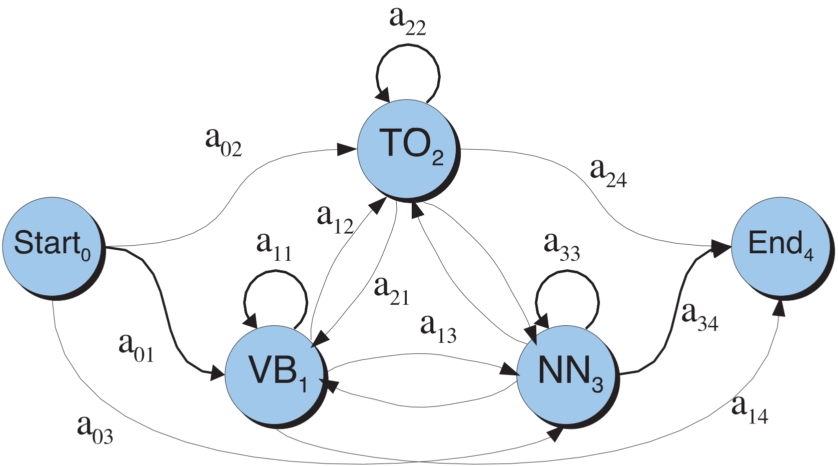
Definition of Hidden Markov Models

For our purposes, a Hidden Markov Model (HMM) consists of:

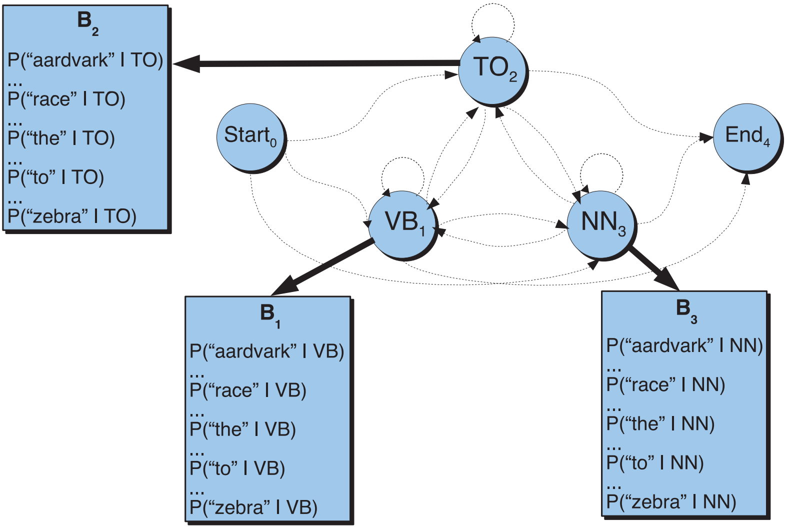
- A set $Q = \{q_0, q_1, \dots, q_T\}$ of states, with q_0 the start state. (Our non-start states will correspond to *parts-of-speech*).
- A transition probability matrix $A = (a_{ij} \mid 0 \leq i \leq T, 1 \leq j \leq T)$, where a_{ij} is the probability of jumping from q_i to q_j. For each i, we require $\sum_{j=1}^{T} a_{ij} = 1$.
- For each non-start state q_i and word type w, an emission probability $b_i(w)$ of outputting w upon entry into q_i. (Ideally, for each i, we'd have $\sum_w b_i(w) = 1$.)

We also suppose we're given an observed sequence $w_1, w_2 \dots, w_n$ of word tokens generated by the HMM.

Transition Probabilities



Emission Probabilities

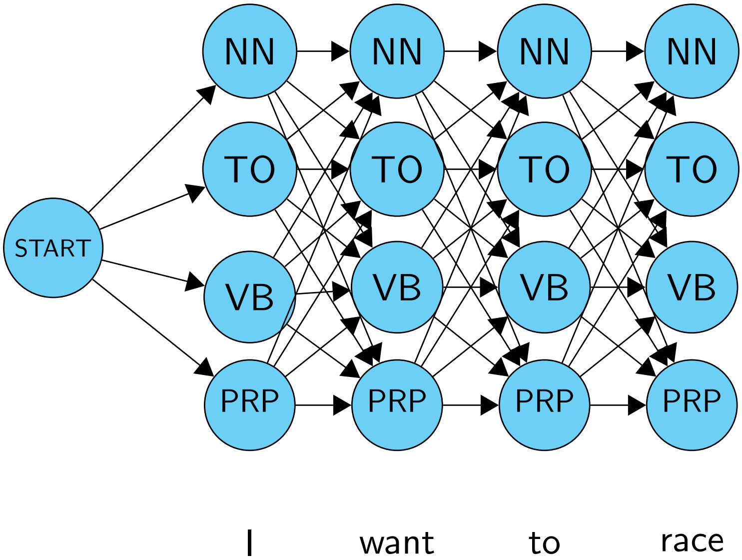


Transition and Emission Probabilities

	VB	**TO**	**NN**	**PRP**
<s>	.019	.0043	.041	.67
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PRP	.23	.00079	.001	.00014

	I	**want**	**to**	**race**
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PRP	.37	0	0	0

The HMM trellis



The Viterbi Algorithm

Keep a chart of the form $\mathrm{Table}(\mathrm{POS}, i)$ where POS ranges over the POS tags and i ranges over the indices in the sentence.

For all T and i:

$$\mathrm{Table}(T, i+1) \leftarrow \max_{T'} \mathrm{Table}(T', i) \times p(T|T') \times p(w_{i+1}|T)$$

and

$$\mathrm{Table}(T, 1) \leftarrow p(T|\langle s \rangle)p(w_1|T)$$

$\mathrm{Table}(., n)$ will contain the **probability** of the most likely sequence. To get the actual sequence, we need backpointers.

The Viterbi Algorithm: second example

q_4	NN	0				
q_3	TO	0				
q_2	VB	0				
q_1	PRP	0				
q_o	start	**1.0**				
		<s>	I	want	to	race
			w_1	w_2	w_3	w_4

- For each state q_j at time i, compute
  $v_i(j) = \max_{k=1}^{n} v_{i-1}(k) a_{kj} b_j(w_i)$

The Viterbi Algorithm

q_4	NN	0				
q_3	TO	0				
q_2	VB	0				
q_1	PRP	0				
q_o	start	**1.0**				
		<s>	I	want	to	race
			w_1	w_2	w_3	w_4

1. Create probability matrix, with one column for each observation (i.e., word token), and one row for each non-start state (i.e., POS tag).
2. We proceed by filling cells, column by column.
3. The entry in column i, row j will be the **probability of the most probable route to state q_j that emits $w_1 \ldots w_i$.**

The Viterbi Algorithm

q_4	NN	0	$1.0 \times .041 \times 0$			
q_3	TO	0	$1.0 \times .0043 \times 0$			
q_2	VB	0	$1.0 \times .19 \times 0$			
q_1	PRP	0	$1.0 \times .67 \times .37$			
q_o	start	**1.0**				
		<s>	I	want	to	race
			w_1	w_2	w_3	w_4

- For each state q_j at time i, compute $v_i(j) = \max_{k=1}^{n} v_{i-1}(k) a_{kj} b_j(w_i)$
- $v_{i-1}(k)$ is **previous Viterbi path probability**, a_{kj} is **transition probability**, and $b_j(w_i)$ is **emission probability**.
- There's also an (implicit) **backpointer** from cell (i, j) to the relevant $(i-1, k)$, where k maximizes $v_{i-1}(k) a_{kj}$.

The Viterbi Algorithm

q_4	NN	0	0	$.025 \times .0012 \times 0.000054$		
q_3	TO	0	0	$.025 \times .00079 \times 0$		
q_2	VB	0	0	$.025 \times .23 \times .0093$		
q_1	PRP	0	**.025**	$.025 \times .00014 \times 0$		
q_0	start	**1.0**				
		<s>	I	want	to	race
			w_1	w_2	w_3	w_4

- For each state q_j at time i, compute $v_i(j) = \max_{k=1}^{n} v_{i-1}(k) a_{kj} b_j(w_i)$
- $v_{i-1}(k)$ is **previous Viterbi path probability**, a_{kj} is **transition probability**, and $b_j(w_i)$ is **emission probability**.
- There's also an (implicit) **backpointer** from cell (i, j) to the relevant $(i-1, k)$, where k maximizes $v_{i-1}(k) a_{kj}$.

The Viterbi Algorithm

q_4	NN	0	0	.000000002	$.000053 \times .047 \times 0$	
q_3	TO	0	0	0	$.000053 \times .035 \times .99$	
q_2	VB	0	0	**.00053**	$.000053 \times .0038 \times 0$	
q_1	PRP	0	**.025**	0	$.000053 \times .0070 \times 0$	
q_0	start	**1.0**				
		<s>	I	want	to	race
			w_1	w_2	w_3	w_4

- For each state q_j at time i, compute $v_i(j) = \max_{k=1}^{n} v_{i-1}(k) a_{kj} b_j(w_i)$
- $v_{i-1}(k)$ is **previous Viterbi path probability**, a_{kj} is **transition probability**, and $b_j(w_i)$ is **emission probability**.
- There's also an (implicit) **backpointer** from cell (i, j) to the relevant $(i-1, k)$, where k maximizes $v_{i-1}(k)a_{kj}$.

The Viterbi Algorithm

q_4	NN	0	0	.000000002	0	$.0000018 \times .00047 \times .00057$
q_3	TO	0	0	0	**.0000018**	$.0000018 \times 0 \times 0$
q_2	VB	0	0	**.00053**	0	$.0000018 \times .83 \times .00012$
q_1	PRP	0	**.025**	0	0	$.0000018 \times 0 \times 0$
q_0	start	**1.0**				
		<s>	I	want	to	race
			w_1	w_2	w_3	w_4

- For each state q_j at time i, compute $v_i(j) = \max_{k=1}^{n} v_{i-1}(k) a_{kj} b_j(w_i)$
- $v_{i-1}(k)$ is **previous Viterbi path probability**, a_{kj} is **transition probability**, and $b_j(w_i)$ is **emission probability**.
- There's also an (implicit) **backpointer** from cell (i, j) to the relevant $(i-1, k)$, where k maximizes $v_{i-1}(k) a_{kj}$.

The Viterbi Algorithm

q_4	NN	0	0	.000000002	0	4.8222e-13
q_3	TO	0	0	0	**.0000018**	0
q_2	VB	0	0	**.00053**	0	**1.7928e-10**
q_1	PRP	0	**.025**	0	0	0
q_0	start	**1.0**				
		<s>	I	want	to	race
			w_1	w_2	w_3	w_4

- For each state q_j at time i, compute
  $v_i(j) = \max_{k=1}^{n} v_{i-1}(k) a_{kj} b_j(w_i)$
- $v_{i-1}(k)$ is **previous Viterbi path probability**, a_{kj} is **transition probability**, and $b_j(w_i)$ is **emission probability**.
- There's also an (implicit) **backpointer** from cell (i, j) to the relevant $(i-1, k)$, where k maximizes $v_{i-1}(k) a_{kj}$.

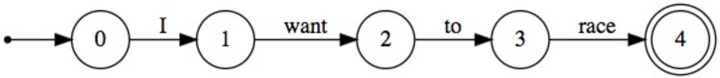
Example Demo

`http://nlp.stanford.edu:8080/parser/`

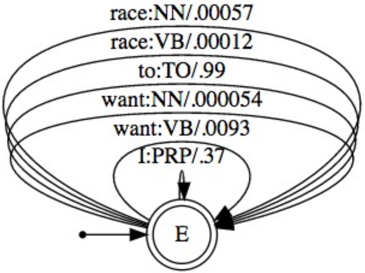
- Relies both on "distributional" and "morphological" criteria
- Uses a model similar to hidden Markov models

What is the connection between HMMs and FSTs? Why are FSTs useful? What does composition of FSTs mean?

Input as an FST

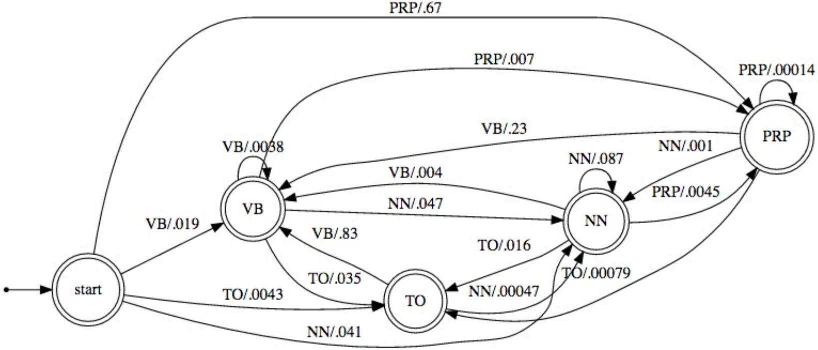


Emission table as an FST



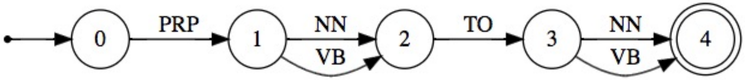
Notice the weights on the FST

Transition table as an FST



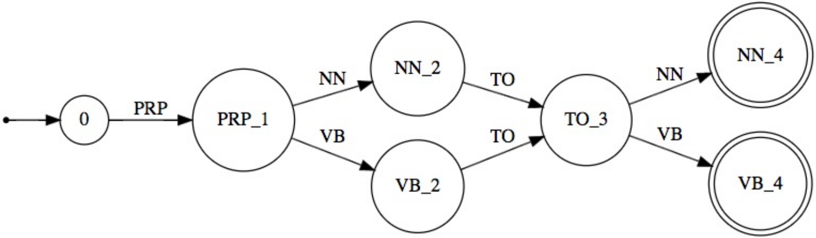
Input fst composed with emission fst

Input fst composed with emission fst



... Composed with transition fst

... Composed with transition fst



Language models

Rather than generate tag conditioned on previous tag, generate word conditioned on previous word.

Language models

Rather than generate tag conditioned on previous tag, generate word conditioned on previous word.

Bigrams:

Months the my and issue of year foreign new exchanges september were recession exchange new endorsed a acquire to six executives

Language models

Rather than generate tag conditioned on previous tag, generate word conditioned on previous word.

Bigrams:

Months the my and issue of year foreign new exchanges september were recession exchange new endorsed a acquire to six executives

Trigrams:

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the maj or central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her.

Language models

4-grams:

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions.

Language models

4-grams:

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions.

This basic idea is fundamental in any system that generates language: machine translation, speech recognition, optical character recognition, image captioning.

As we've just seen, can be (and is) implemented as a very large weighted FST!

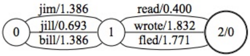
Speech recognition

Task: convert soundwaves to corresponding text.

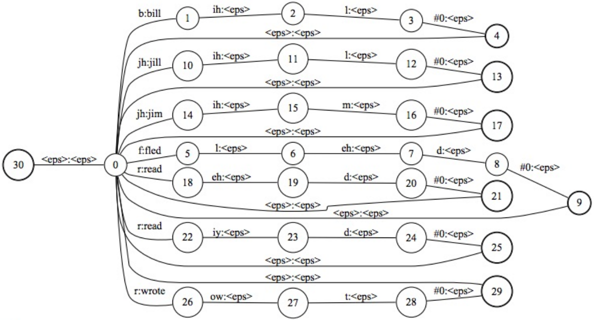
Intuition: both sound and text are (noisy) representations of the same underlying set of **phonemes**.

Mapping from words to phonemes is just transduction! (From phonemes to sound, signal processing). Coupled with a very large language model...

Speech recognition transducers (1)



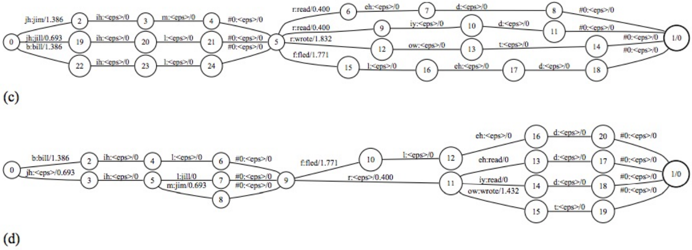
(a)



(b)

(a) Language model; (b) Phonemes to words

Speech recognition transducers (1)



(c) Composition of the LM and phoneme to word transducer; (d) Determinisation of c.

Machine tanslation

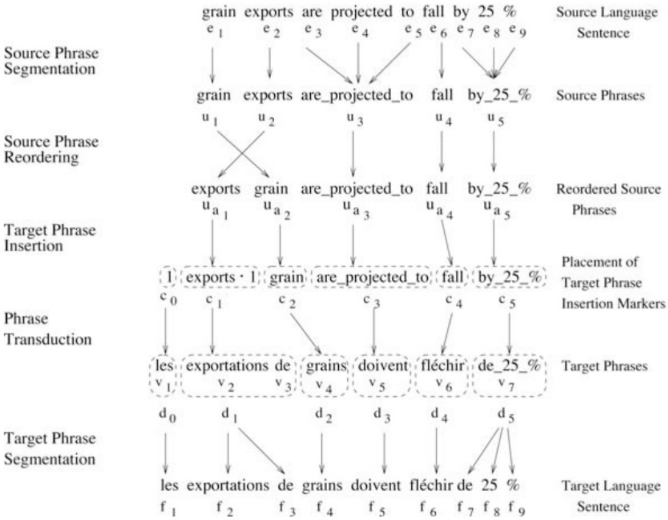
watashi wa hako wo akemasu $\rightarrow$ I open the box

(Japanese gloss: "I the box open", with two case markers)

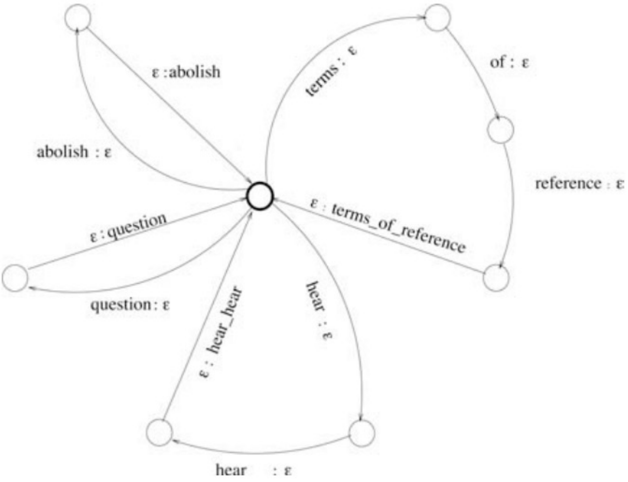
Two basic operation of a machine translation system:

1. substitute words or sequences of words.
2. permute word sequences.

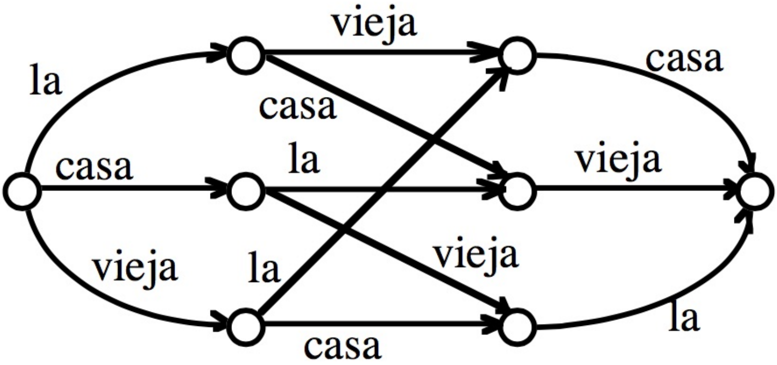
Machine translation models



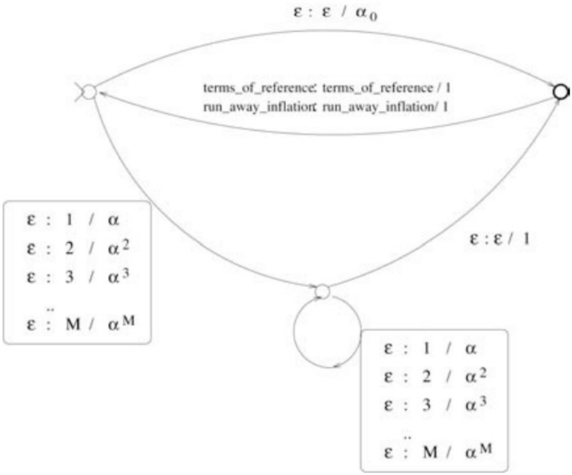
The segmentation transducer



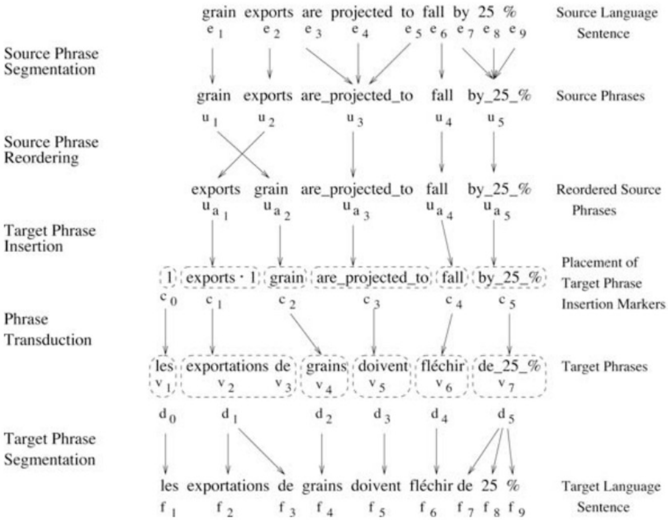
The permutation transducer



The insertion transducer



Machine translation models (again)



Every step can be encoded as an FST. Compose and run Viterbi!

Other applications

A search on Google scholar for “finite state transducers” leads to over 200,000 results.

Other applications for natural language:

- Named entity recognition
- Text normalisation
- Information extraction

FSTs are modular (can break the problem into different sub-problems and cascade the resulting FSTs together), highly efficient and are simple to understand and work with.

Takeaways:

1. Viterbi algorithm
2. Weighted finite state transducers are incredibly useful!

Next class: parsing natural language.