OPTICAL CHARACTER RECOGNITION USING HIDDEN MARKOV MODELS

Jan Rupnik

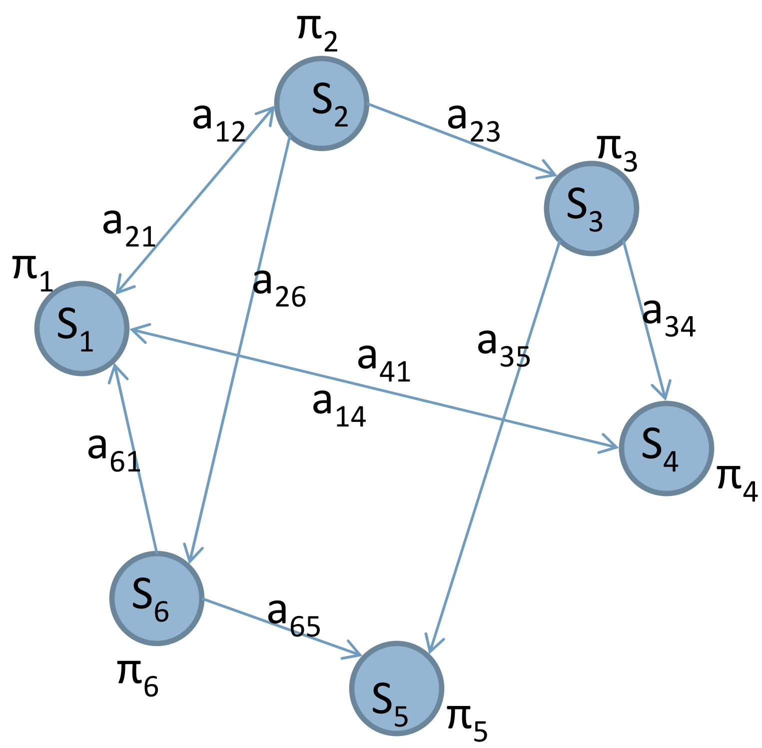
OUTLINE

- HMMs
  - Model parameters
  - Left-Right models
  - Problems
- OCR - Idea
- Symbolic example
- Training
- Prediction
- Experiments

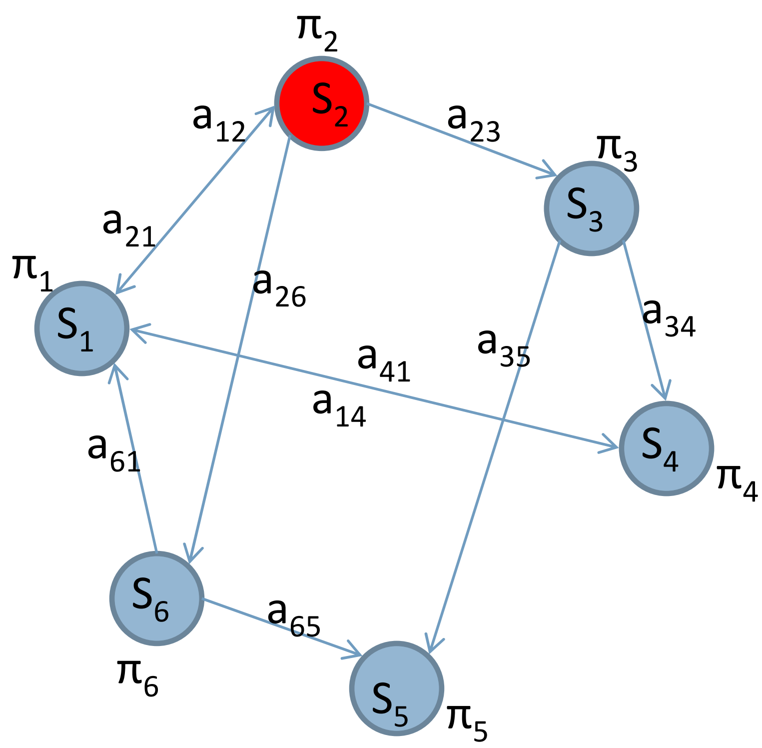
HMM

- Discrete Markov model : probabilistic finite state machine
- Random process: random memoryless walk on a graph of nodes called states.
- Parameters:
  - Set of states $S = \{S_1, ..., S_n\}$ that form the nodes
  - Let q_t denote the state that the system is in at time t
  - Transition probabilities between states that form the edges, $a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i),\ 1 \leq i,j \leq n$
  - Initial state probabilities, $\pi_i = P(q_1 = S_i),\ 1 \leq i \leq n$

MARKOV PROCESS



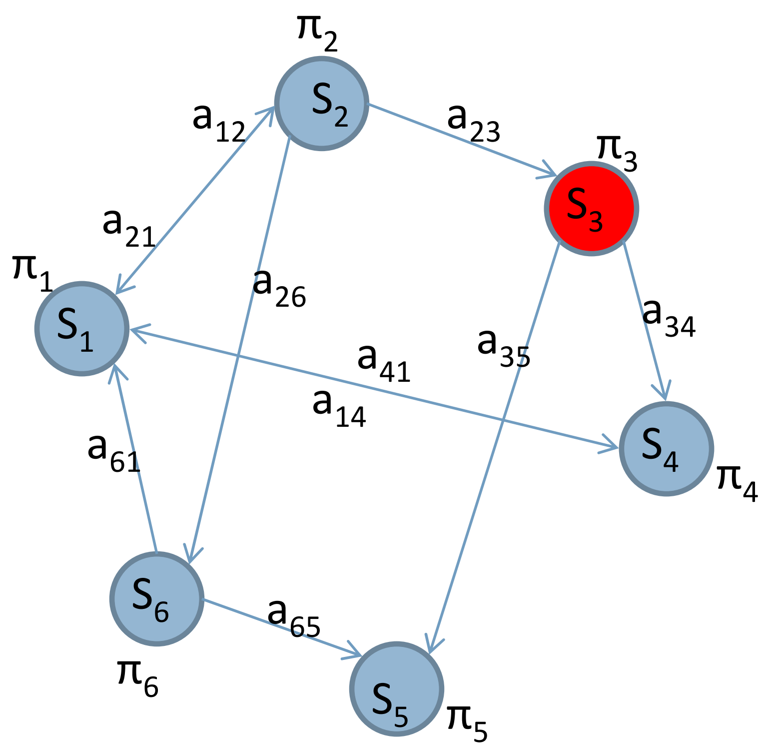
MARKOV PROCESS



Pick q_1 according to the distribution π

$q_1 = S_2$

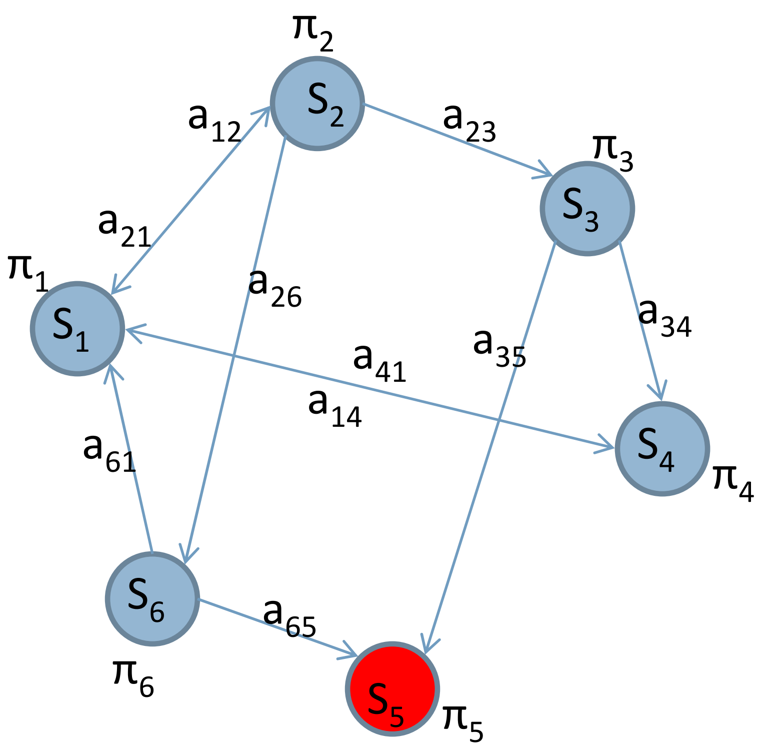
MARKOV PROCESS



Move to a new state according to the distribution a_{ij}

$q_2 = S_3$

MARKOV PROCESS



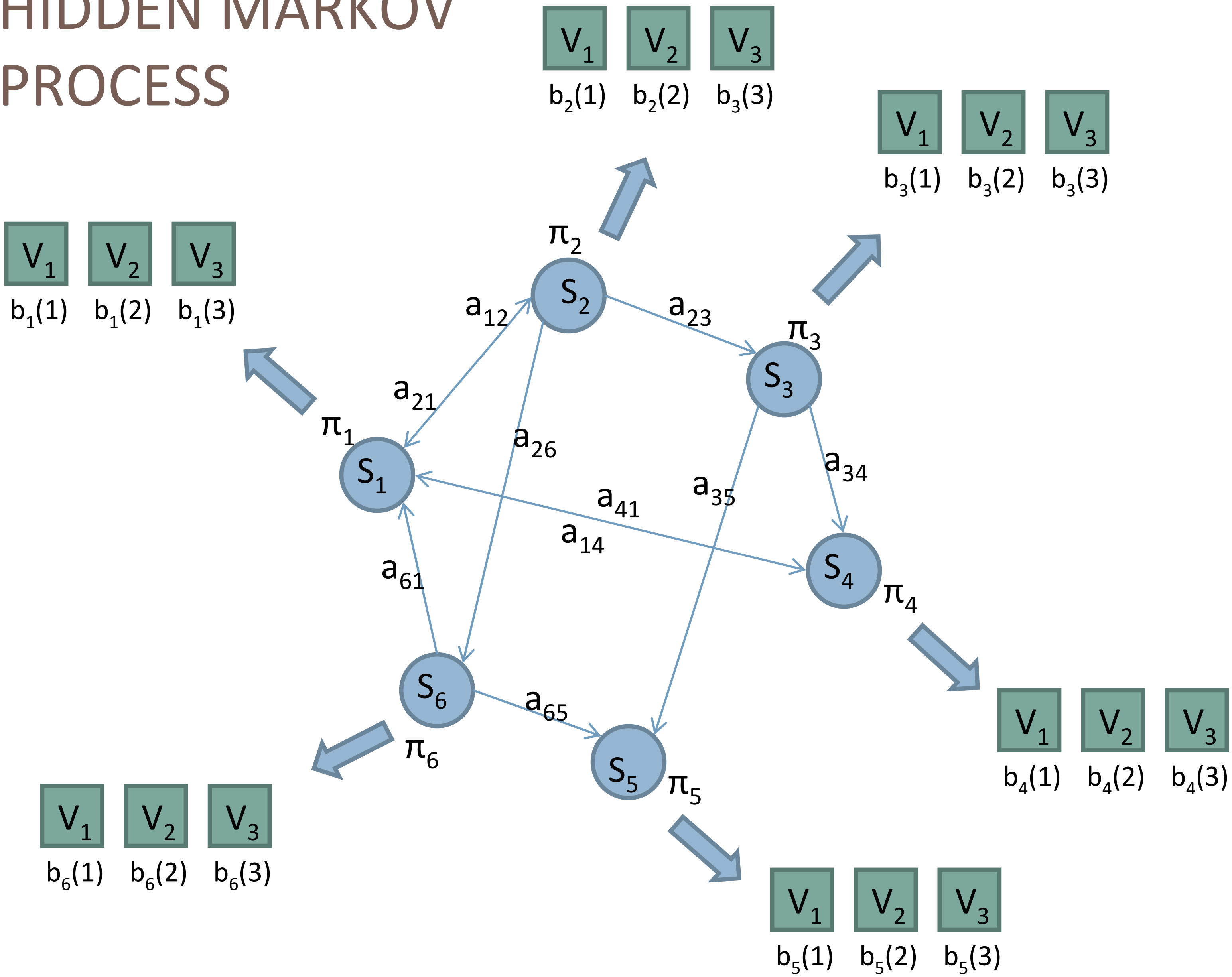
Move to a new state according to the distribution a_{ij}

$q_3 = S_5$

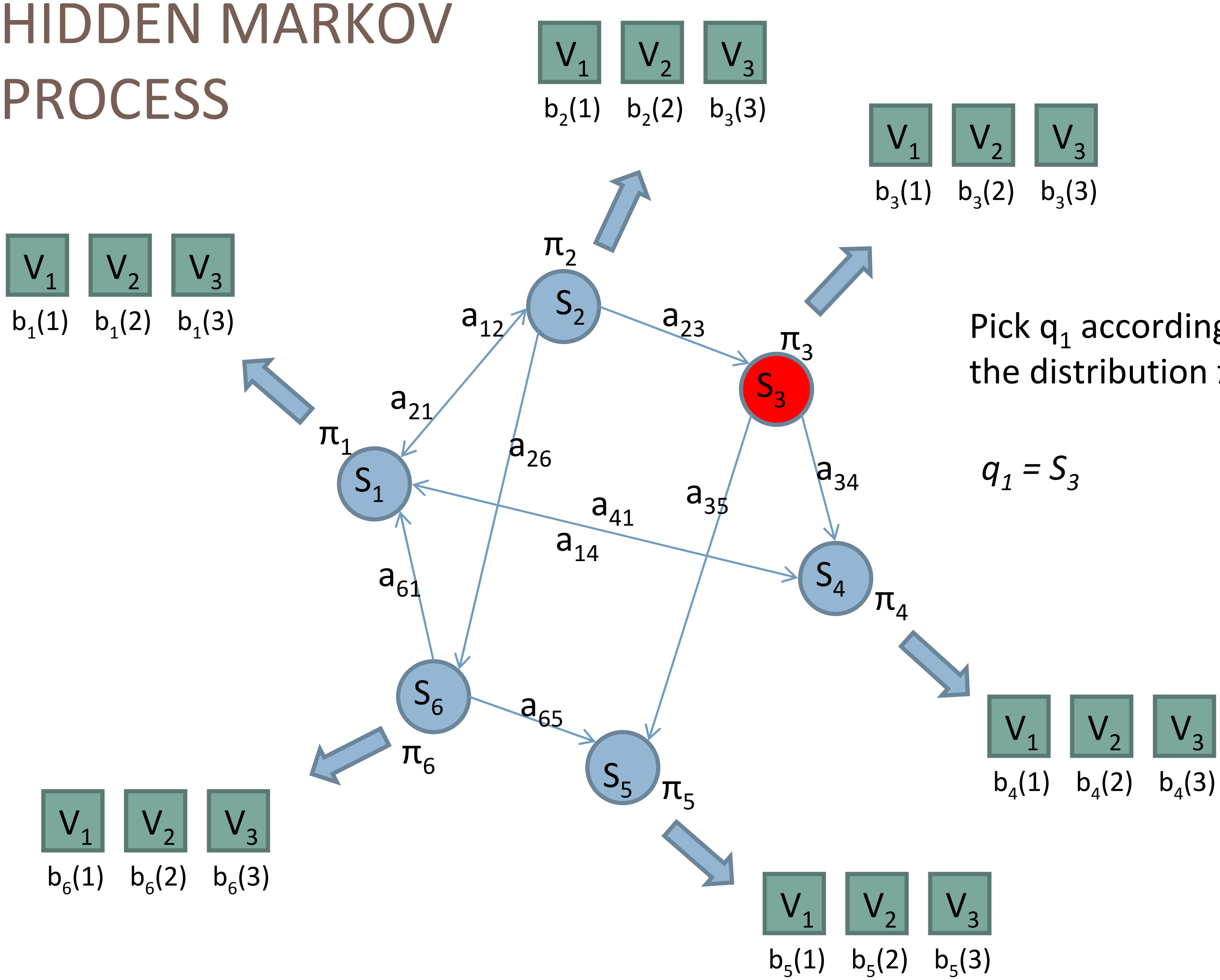
HIDDEN MARKOV PROCESS

- The Hidden Markov random process is a partially observable random process.
- Hidden part: Markov process q_t
- Observable part: sequence of random variables with the same domain v_t, where conditioned on the variable q_i the distribution of the variable v_i is independent of every other variable, for all i = 1,2,...
- Parameters:
  - Markov process parameters: π_i, a_{ij} for generation of q_t
  - Observation symbols $V = \{V_1, \dots , V_m\}$
  - Let v_t denote the observation symbol emitted at time t.
  - Observation emission probabilities $b_i(k) = P(v_t = V_k \mid q_t = S_i)$
- Each state defines its own distribution over observation symbols

HIDDEN MARKOV PROCESS



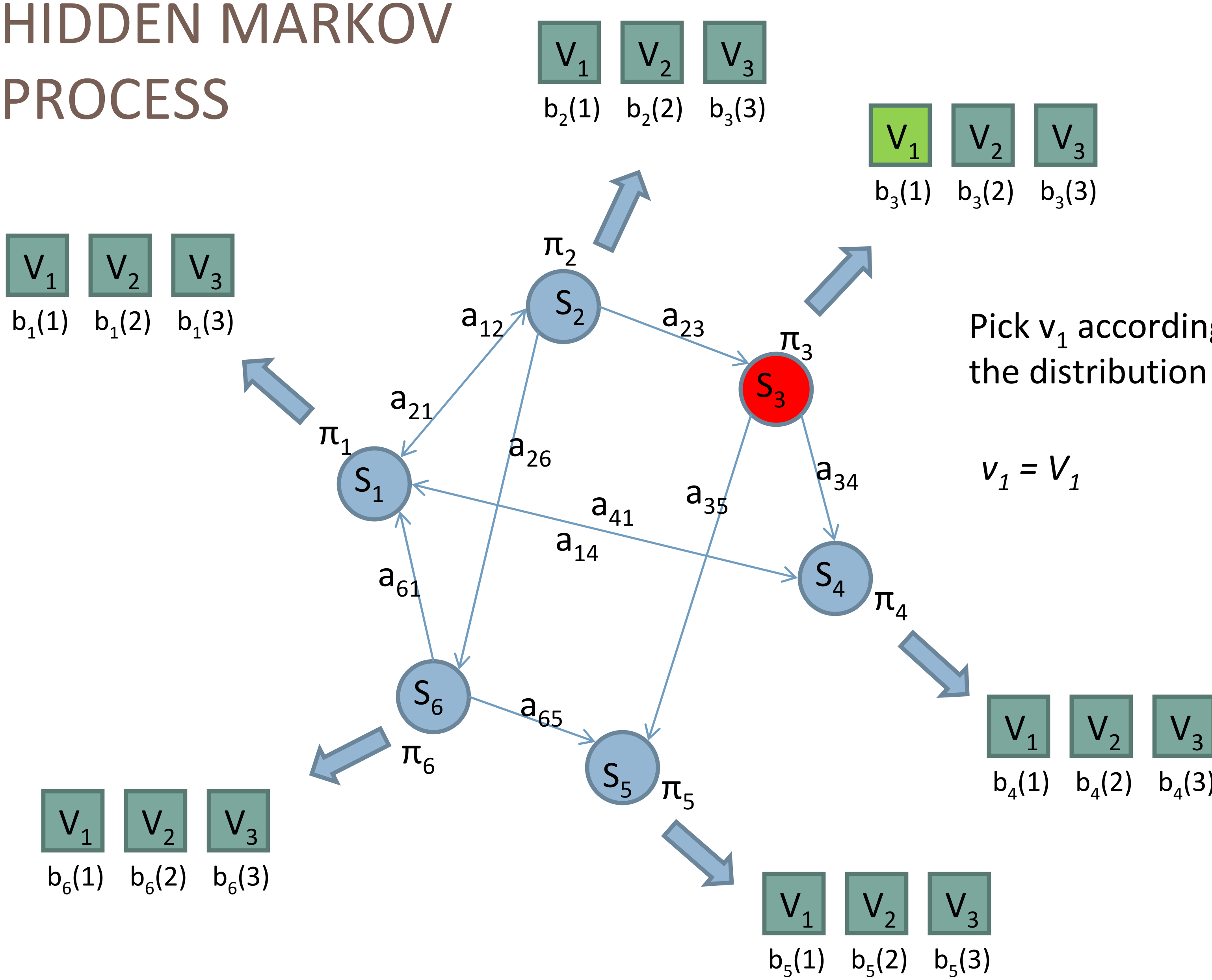
HIDDEN MARKOV PROCESS



Pick q_1 according to the distribution π

$q_1 = S_3$

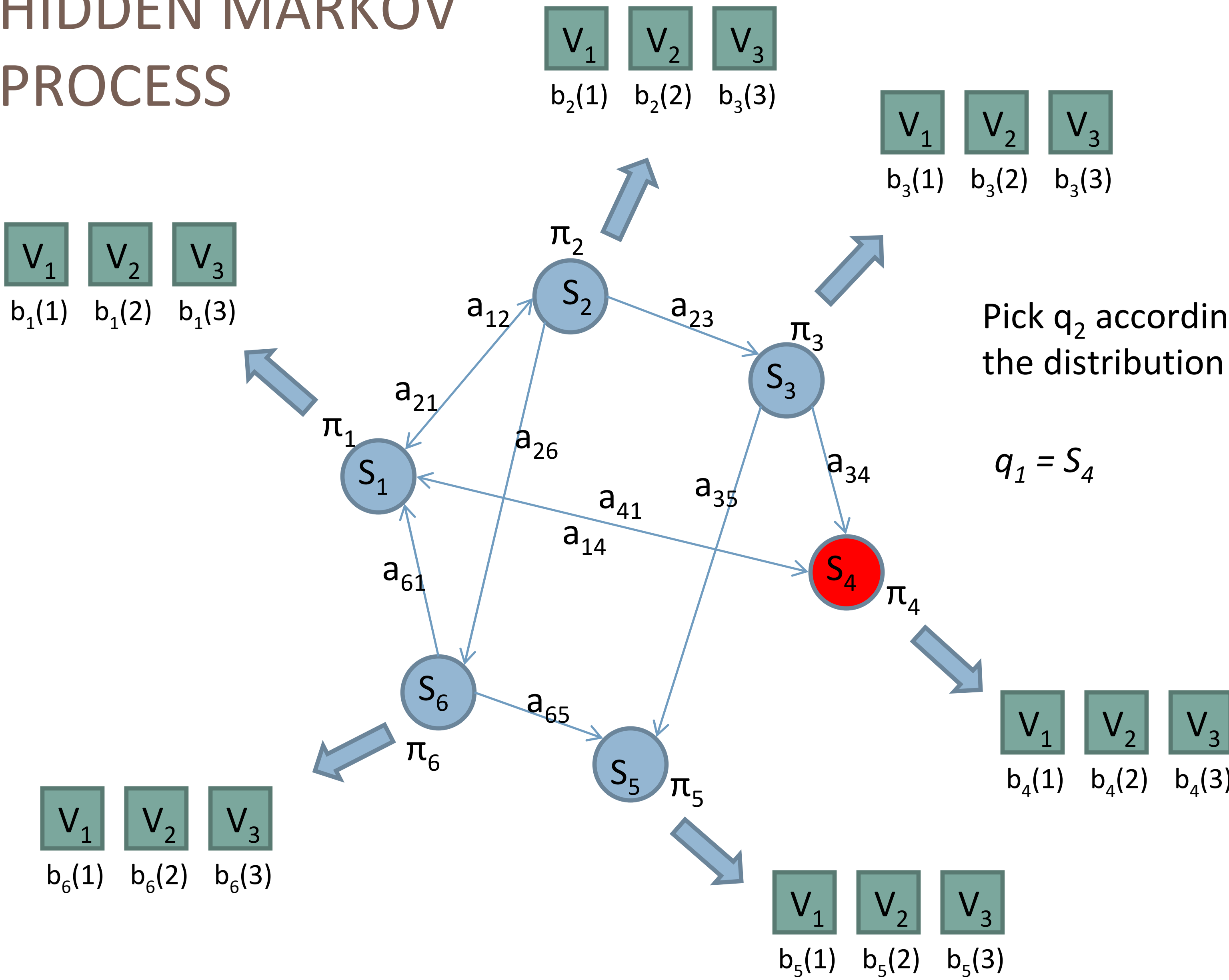
HIDDEN MARKOV PROCESS



Pick v_1 according to the distribution b_1

$v_1 = V_1$

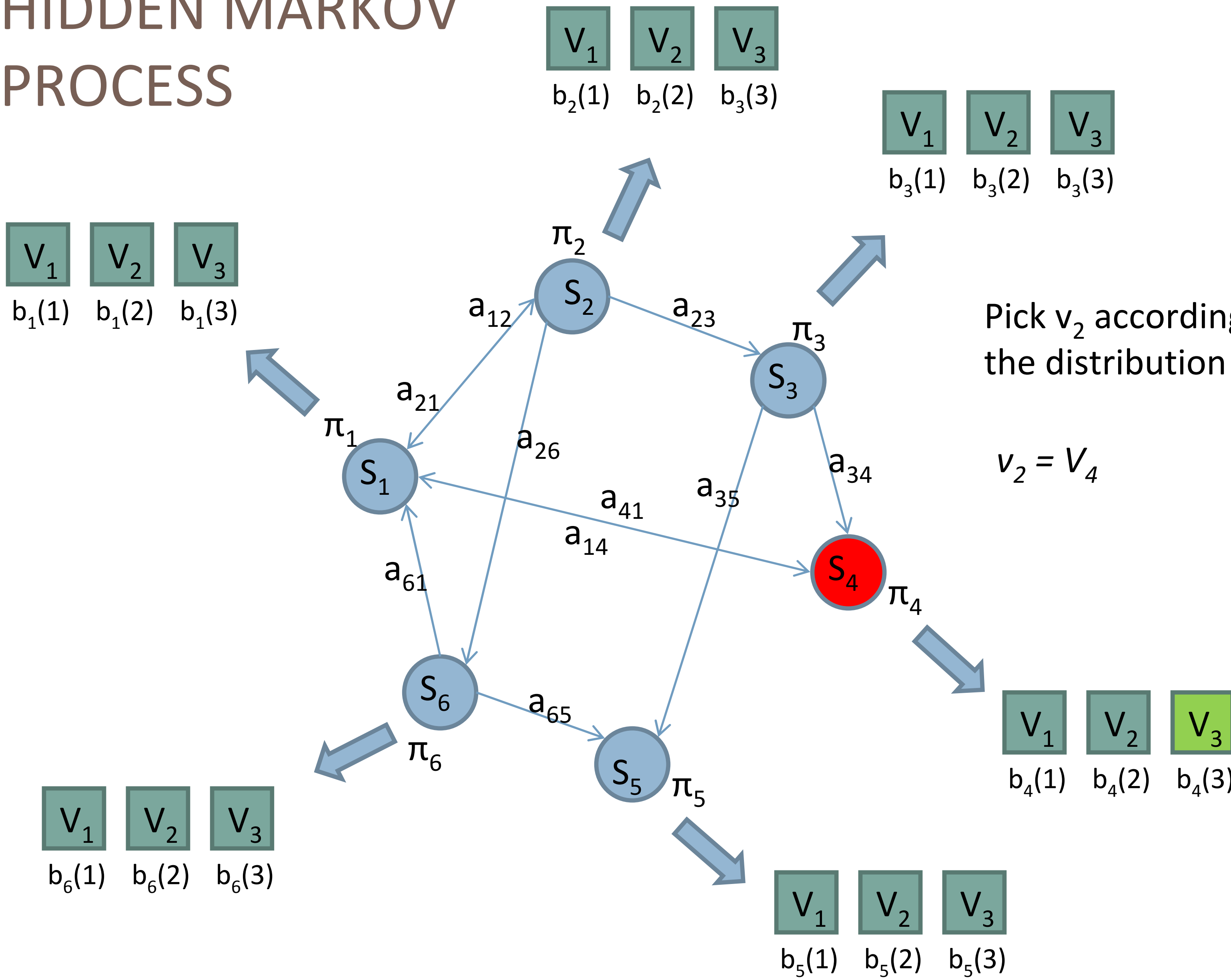
HIDDEN MARKOV PROCESS



Pick q_2 according to the distribution a_{ij}

$q_1 = S_4$

HIDDEN MARKOV PROCESS

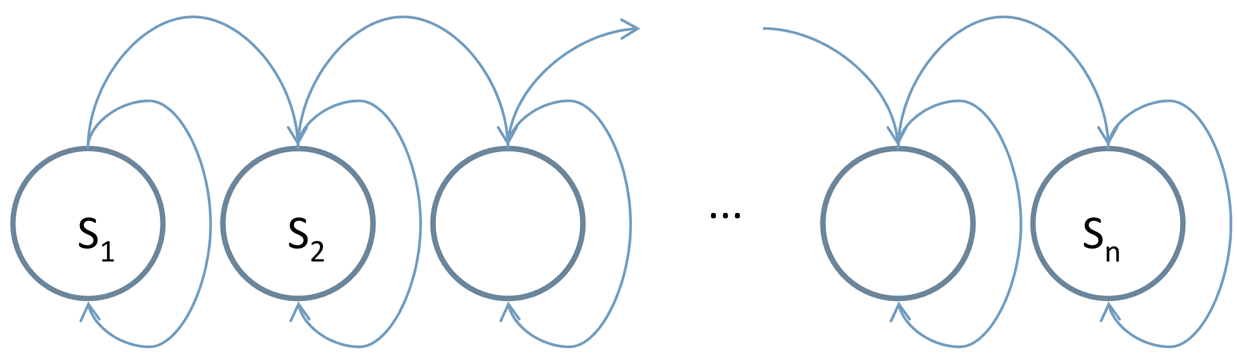


Pick v_2 according to the distribution b_4

$v_2 = V_4$

LEFT RIGHT HMM

- Sparse and easy to train
- Parameters
  - $\pi_1 = 1, \pi_i = 0, i > 1$
  - $a_{ij} = 0 \text{ if } i > j$
- Left-right HMM example

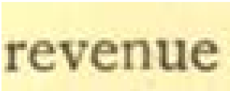


PROBLEMS

- Given a sequence of observations $v_1, \ldots, v_T$ find the sequence of hidden states $q_1, \ldots, q_T$ that most likely generated it.
- Solution: Viterbi algorithm (dynamic programming, complexity: $O(Tn^2)$)
- How to determine the model parameters π_i, a_{ij}, $b_i(k)$?
- Solution: Expectation Maximization (EM) algorithm that finds the local maximum of the parameters, given a set of initial parameters π_i^0, a_{ij}^0, $b_i(k)^0$.

OPTICAL CHARACTER RECOGNITION

Input for **training** of the OCR system: pairs of word images and their textual strings



"THE" "revenue" "of" …

Input for the **recognition** process: a word image

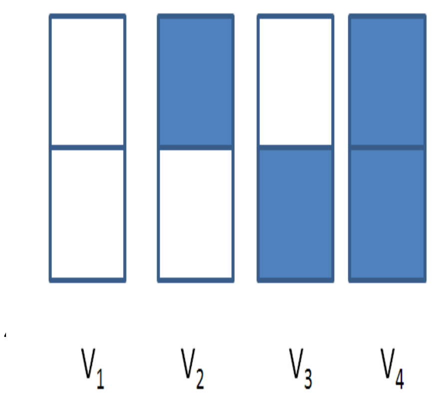
"?"

IDEA

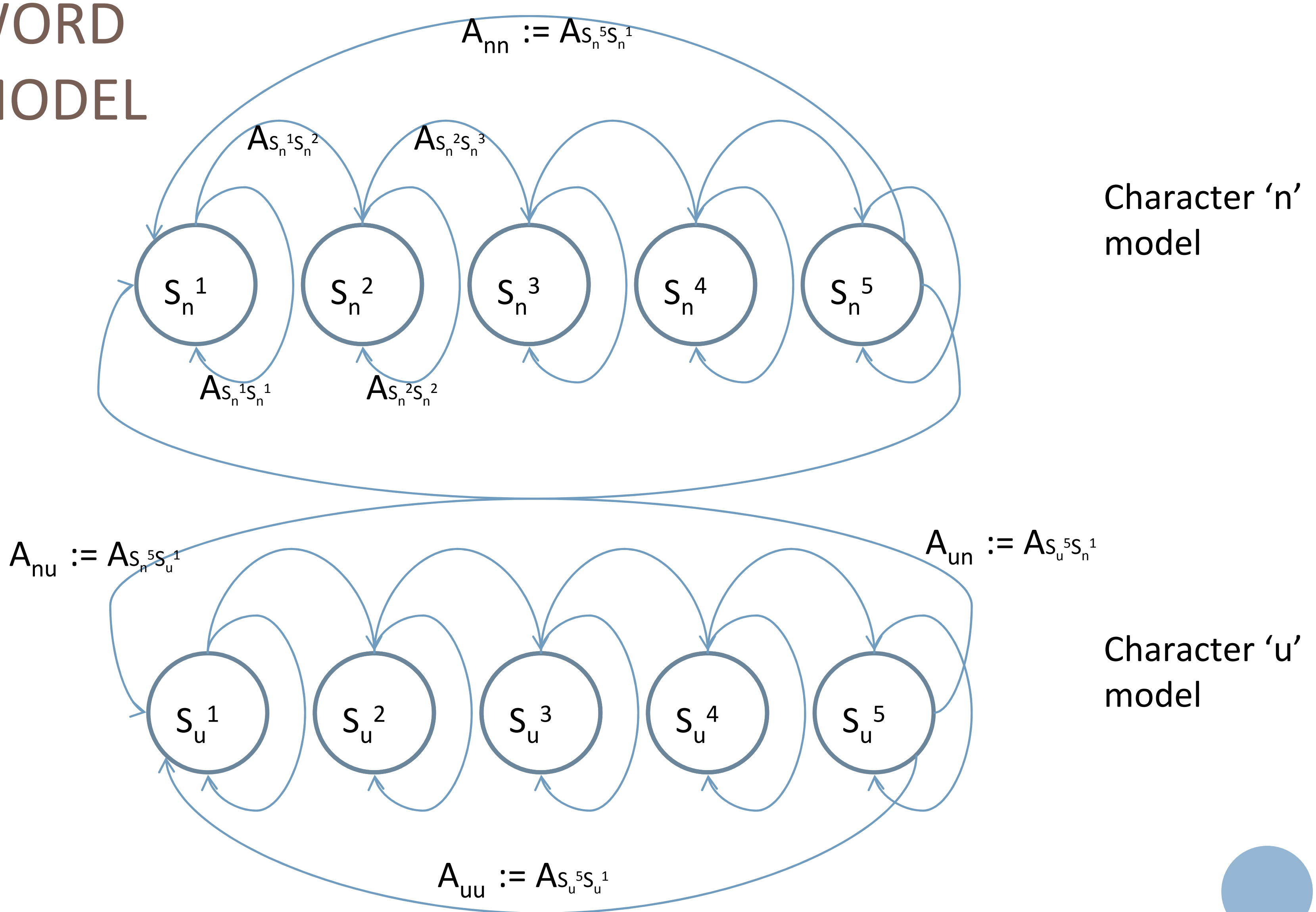
- The modelling of the generation of character images is accomplished by generating sequences of thin vertical images (segments)
- Build a HMM for each character separately (model the generation of images of character ‘a’, ‘b’,...)
- Merge the character HMMs into the word HMM (model the generation of sequences of characters and their images)
- Given a new image of a word, use the word HMM to predict the most likely sequence of characters that generated the image.

SYMBOLIC EXAMPLE

- Example of word images generation for a two character alphabet {‘n’, ‘u’}.
- Set S = $\{S_u^1, \ldots, S_u^5, S_n^1, \ldots, S_n^5\}$
- Set V = $\{V_1, V_2, V_3, V_4\}$
- Assign to each V_i a thin vertical image:
- The word model (for words like ‘unnunuuu
is constructed by joining two left-right character models.

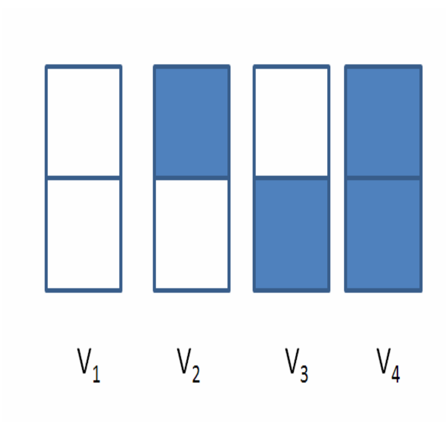


WORD MODEL



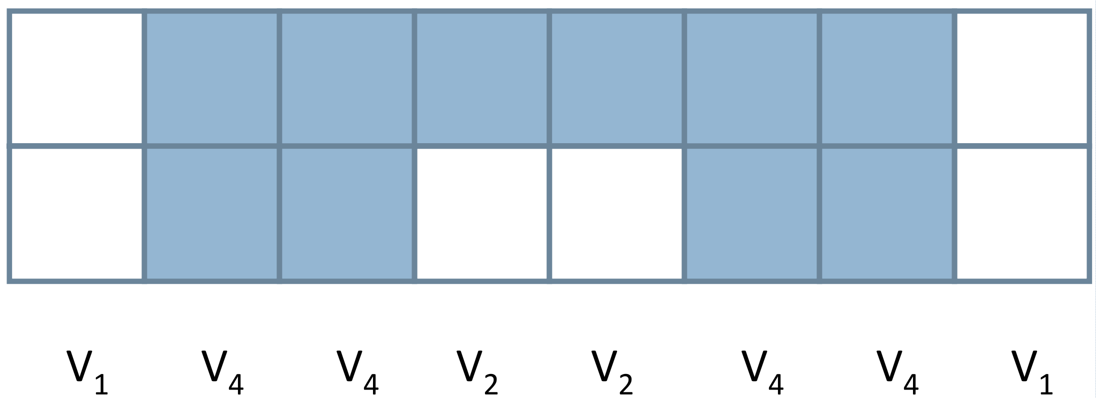
Word model state transition architecture

WORD MODEL



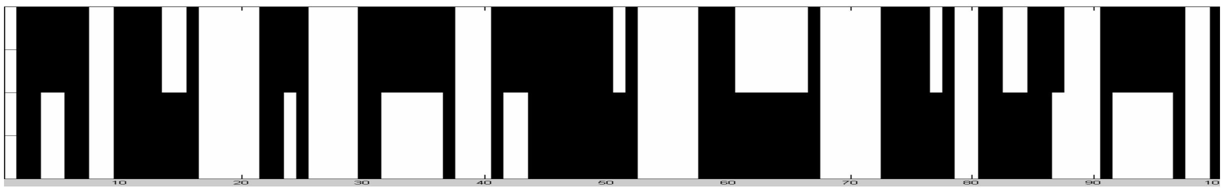
A	Sn1	Sn2	Sn3	Sn4	Sn5	Su1	Su2	Su3	Su4	Su5
Sn1	0.5	0.5								
Sn2		0.5	0.5							
Sn3			0.5	0.5						
Sn4				0.5	0.5					
Sn5	**0.33**				0.33	**0.33**				
Su1						0.5	0.5			
Su2							0.5	0.5		
Su3								0.5	0.5	
Su4									0.5	0.5
Su5	**0.33**					**0.33**				0.33

B	V1	V2	V3	V4
Sn1	1			
Sn2			0.05	0.95
Sn3		1		
Sn4			0.05	0.95
Sn5	1			
Su1	1			
Su2		0.05		0.95
Su3			1	
Su4		0.05		0.95
Su5	1			

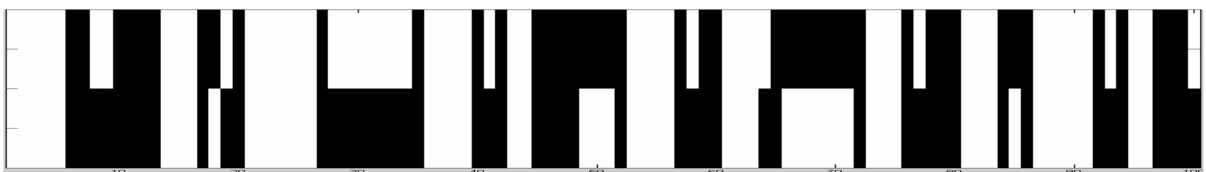


A sequence of observation symbols that correspond to an "image" of a word

EXAMPLE OF IMAGES OF GENERATED WORDS



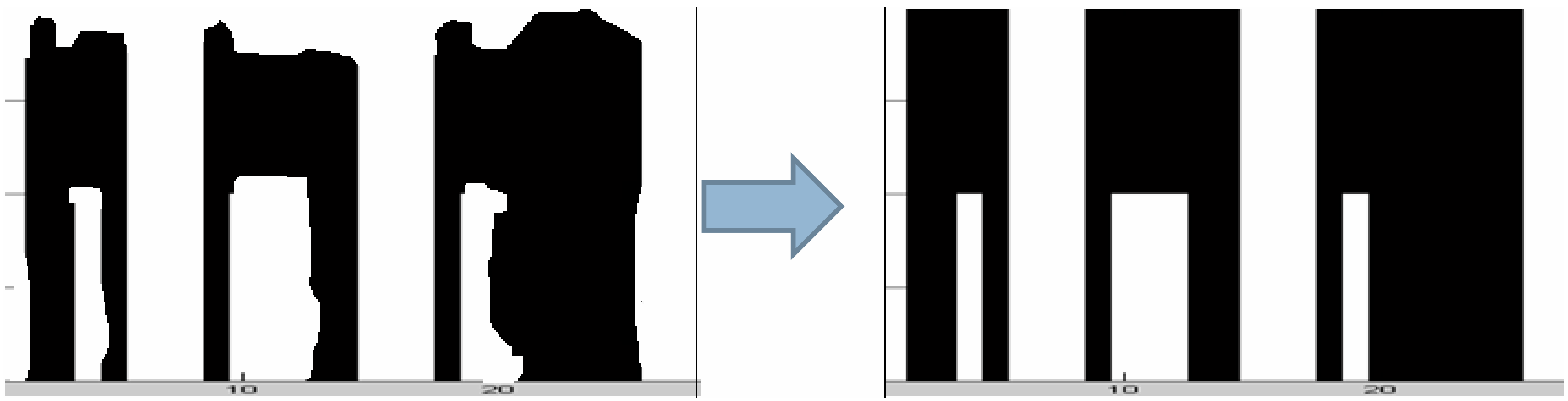
Example: word 'nunnnuuun'



Example: word ‘uuuunununu’

RECOGNITION

Find best matching patterns



1-1 correspondence

Viterbi

$S_n^1S_n^2S_n^2S_n^3S_n^4S_n^5S_n^1S_n^1S_n^2S_n^3S_n^3S_n^3S_n^4S_n^4S_n^5$ ...
...$S_n^1S_n^1S_n^2S_n^3S_n^4S_n^4S_n^4S_n^4S_n^4S_n^4S_n^5S_n^1S_n^1$

$V_1V_4V_4V_2V_4V_1V_1V_1V_4V_2V_2V_2V_4V_4V_1V_1V_1V_4V_2V_4V_4V_4V_4V_4V_4V_1V_1$

Look at transitions of type $S_*^5S_{**}^1$ to find transitions from character to character

Predict: ‘nnn’

FEATURE EXTRACTION, CLUSTERING, DISCRETIZATION

- **Discretization**: if we have a set of basic patterns (thin images of the observation symbols), we can transform any sequence of thin images into a sequence of symbols (previous slide) – the input for our HMM.
- We do not deal with images of thin slices directly but rather with some **feature vectors** computed from them (and then compare vectors instead of matching images).
- The basic patterns (feature vectors) can be found with **k-mean clustering** (from a large set of feature vectors).

FEATURE EXTRACTION

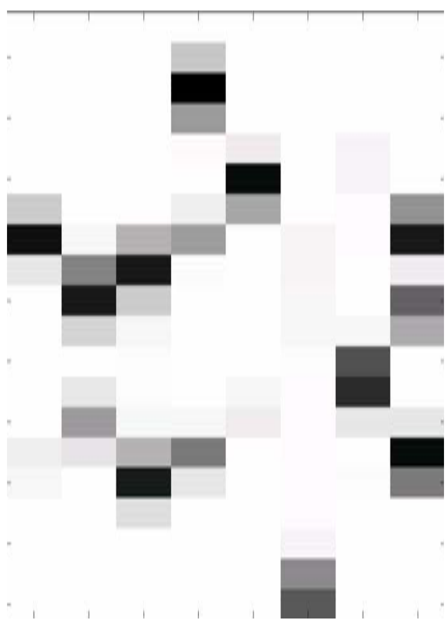
•Transformation of the image into a sequence of 20-dimensional feature vectors.

•Thin overlapping rectangles split into 20 vertical cells.

•The feature vector for each rectangle is computed by computing the average luminosity of each cell.

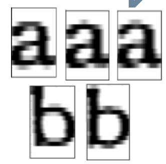
CLUSTERING

- Given a large set of feature vectors (100k) extracted from the training set of images and compute the k-means clustering with 512 clusters.
- Eight of the typical feature vectors:



TRAINING

GATHERING OF TRAINING EXAMPLES



Input: images of words
Output: instances of character images

FEATURE EXTRACTION

Input: instances of character images
Output: a sequence of 20-dimensional vectors per character intance

CLUSTERING

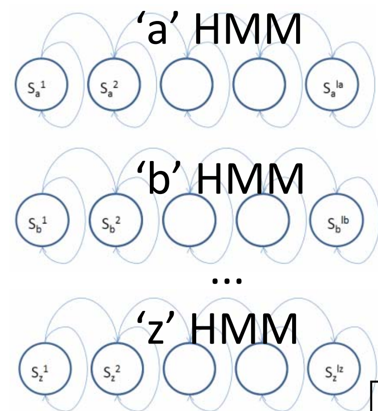
Input: all feature vectors computed in the previous step, number of clusters
Output: a set of centroid vectors C

GET CHARACTER TRANSITION PROBABILITIES

Input: large corpus of text
Output: character transition probabilities

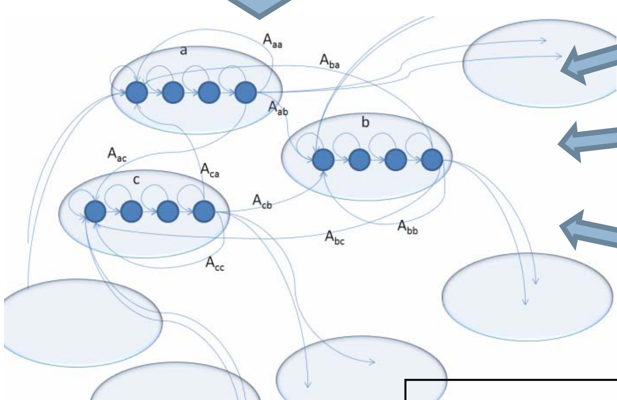
DISCRETIZATION

Input: a sequence of feature vectors for every character instance, C
Output: a sequence of discrete symbols for every character instance



CHARACTER MODELS

Input: a sequence of observation symbols for each character instance
Output: separately trained character HMMs for each character



WORD MODEL

Input: character transition probabilities, character HMMs
Output: word HMM

PREDICTION

An image of a word that is to be recognized

FEATURE EXTRACTION

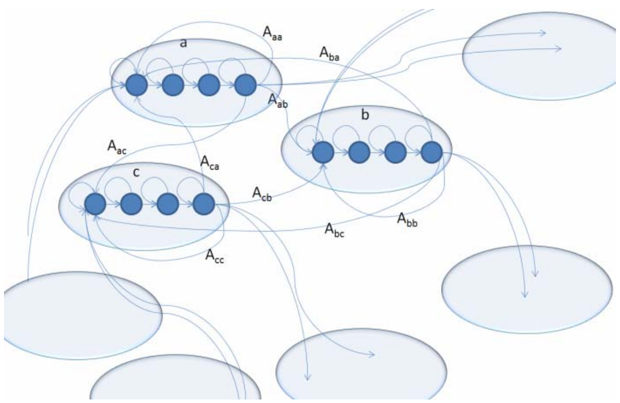
Input: image of a word
Output: sequence of 20-dimensional vectors

Set of 20-dimensional centroid vectors C, computed in the **training phase**.

DISCRETIZATION

Input: sequence of feature vectors for the image of a word, C computed in training phase
Output: a sequence of symbols from a finite alphabet for the image of a word

WORD MODEL computed in the **training phase**.



VITERBI DECODING

Input: word HMM and a sequence of observation symbols
Output: sequence of states that most likely emitted the observations.

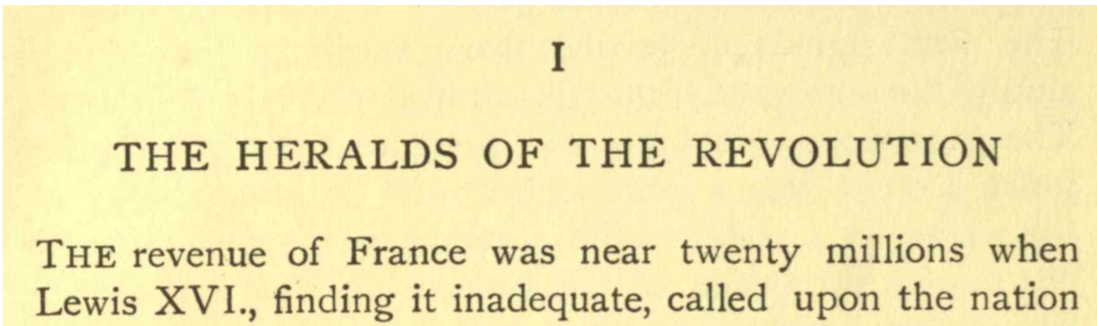
PREDICTION

Input: sequence of states
Output: sequence of characters

‘c’ ‘o’ ‘n’ ‘t’ ‘i’ ‘n’ ‘u’ ‘i’ ‘t’ ‘y’

EXPERIMENTS

- Book on French Revolution, John Emerich Edward Dalberg-Acton (1910) (source: archve.org)



I

THE HERALDS OF THE REVOLUTION

THE revenue of France was near twenty millions when Lewis XVI., finding it inadequate, called upon the nation

- Test word error rate on the words containing lower-case letters only.
- Approximately 22k words on the first 100 pages

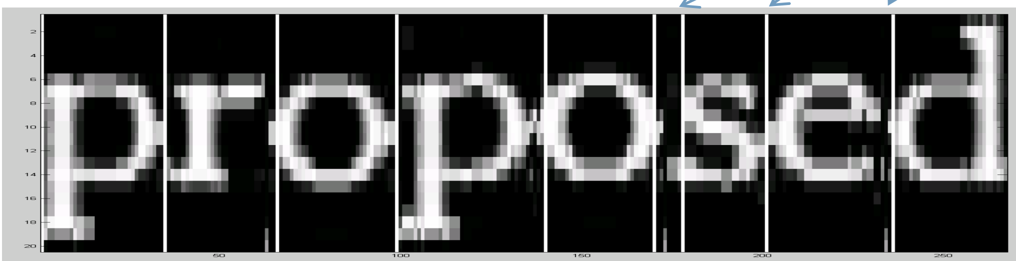
EXPERIMENTS – DESIGN CHOICES

- Extract 100 images of each character
- Use 512 clusters for discretization
- Build 14-state models for all characters, except for ‘i’, ’j’ and ‘l’ where we used 7-state models, and ‘m’ and ‘w’ where we used 28-state models.
- Use character transition probabilities from [1]
- Word error rate: 2%

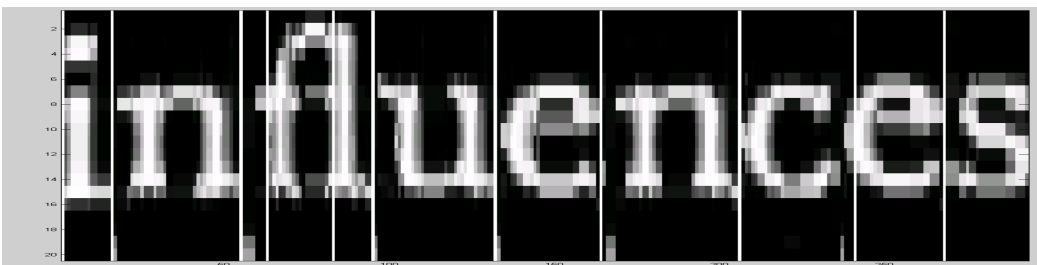
[1] Michael N. Jones; D.J.K. Mewhort , Case-sensitive letter and bigram frequency counts from large-scale English corpora, *Behavior Research Methods, Instruments, & Computers, Volume 36, Number 3*, August 2004 , pp. 388-396(9)

TYPICAL ERRORS

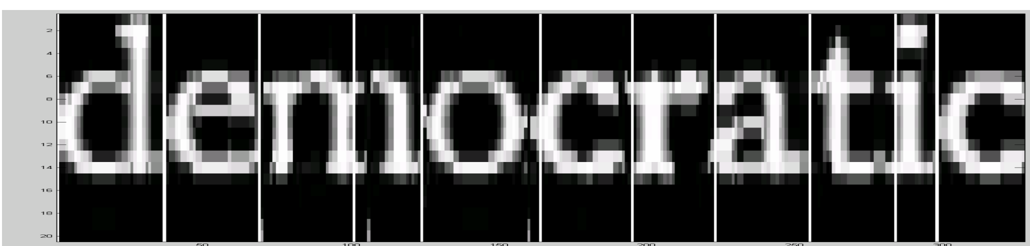
Places where the system predicted transitions between character models



Predicted: **proposled**. We see that the there is a short strip between 'o' and 's' where the system predicted an 'l'.



Predicted: **inifluenoes**. The system interpreted the character 'c' and the beginning of character 'e' as an 'o', and it predicted an extra ‘i'.



Predicted: **dennocratic**. The character 'm' was predicted as a sequence of two 'n' characters.

QUESTIONS?