COLUMBIA UNIVERSITY

Statistical Methods for NLP

Hidden Markov Models II, MapReduce

Sameer Maskey

•Some of the lecture slides are from
Bhuvana Ramabhadran, Stanley Chen, Michael Picheny, Ellen Eide

Announcements

- Homework 2 out
  - Start early!!
  - Submission Method :

    Upload it in your courseworks dropbox
- Project Intermediate Report
  - Due next Wednesday
  - 1 person team – 4 pages
  - 2 person team – 6 pages
  - 3 person team – 8 pages
- 10% of the project grade

Project Intermediate Report I

- Title
- Abstract/Summary
- Introduction/Motivation
- Related Work
- Data
- System Description
  - NLP/ML Algorithm
  - User Interaction Mockup
  - Experiment
- Results and Discussion
- Conclusion

Project Intermediate Report I Details

- You need to adhere to the format I have described
- You need to provide data description
- You need to have completed basic prototype implementation of the ML algorithm
- You need to have done 1 set of experiments
- You need to present at least 1 table of results
- You need to present a mockup of web interface
- You need to submit through courseworks dropbox

Topics for Today

- Hidden Markov Models
- MapReduce
- Machine Learning with large scale data
- Amazon Web Services and Project setup in amazon

Hidden Markov Models

- We can define HMM by
- State : $Q = q_1 q_2 q_N$
- Transition Probabilities $T = a_{11} a_{12} a_{nn}$
- Emission Probabilities $B = b_i(o_t)$
- Observation Sequence $O = o_1 o_2 o_T$
- Start and Final State q_0, q_F

Markov Model with 5 states with 10 possible observation in each state will have T and B of what sizes?

Three problems of general interest for an HMM

3 problems need to be solved for HMM's:

- Given an observed output sequence $X=x_1x_2..x_T$, compute $P_\theta(X)$ for a given model θ (scoring)

$$P(x_1, x_2, , x_T; \theta)$$

- Given X, find the most likely state sequence (Viterbi algorithm)

  find best $\hat{S}_1, \ldots, \hat{S}_T$ using $\hat{x}_1, \ldots, \hat{x}_T$

- Estimate the parameters of the model (training) using n observed sequences of varying length

$$q(S_t|S_{t-1}), \; b_i(o_t|S_t)$$

Problem 1: Forward Pass Algorithm

Let $\alpha_t(s)$ for t ε {1..T} be the probability of being in state s at time t and having produced output $x_1^t = x_1..x_t$

$$\alpha_t(s) = \sum_{s'} \alpha_{t-1}(s') P_\theta(s|s') P_\theta(x_t|s'->s) + \sum_{s'} \alpha_t(s') P_\theta(s|s')$$

1st term: sum over all output producing arcs 2nd term: all null arcs

This is called the Forward Pass algorithm.

This calculation allows us to solve Problem 1 efficiently:

$$P(x_1, x_2, ..., x_T; \theta) = \sum_s \alpha_T(s)$$

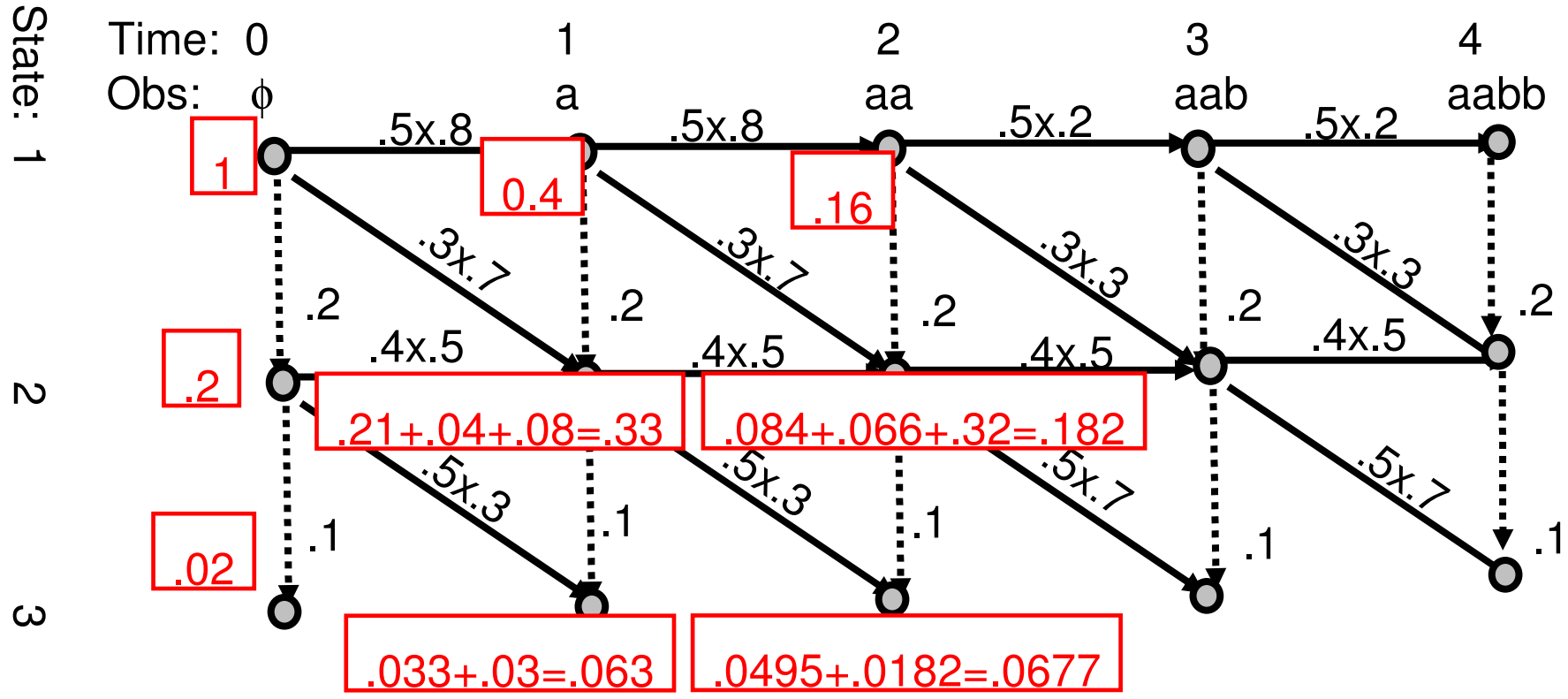
N^2 * T

$$P(x_1, x_2, ..., x_T; \theta) = \sum_{s_1, ...S_T} P(x_1, x_2, ..., x_T, s_1, s_2, ..., s_T)$$

N^T

Problem 1: Trellis Diagram

- Now let's accumulate the scores. Note that the inputs to a node are from the left and top, so if we work to the right and down all necessary input scores will be available.



Problem 2

Given the observations X, find the most likely state sequence

This is solved using the Viterbi algorithm

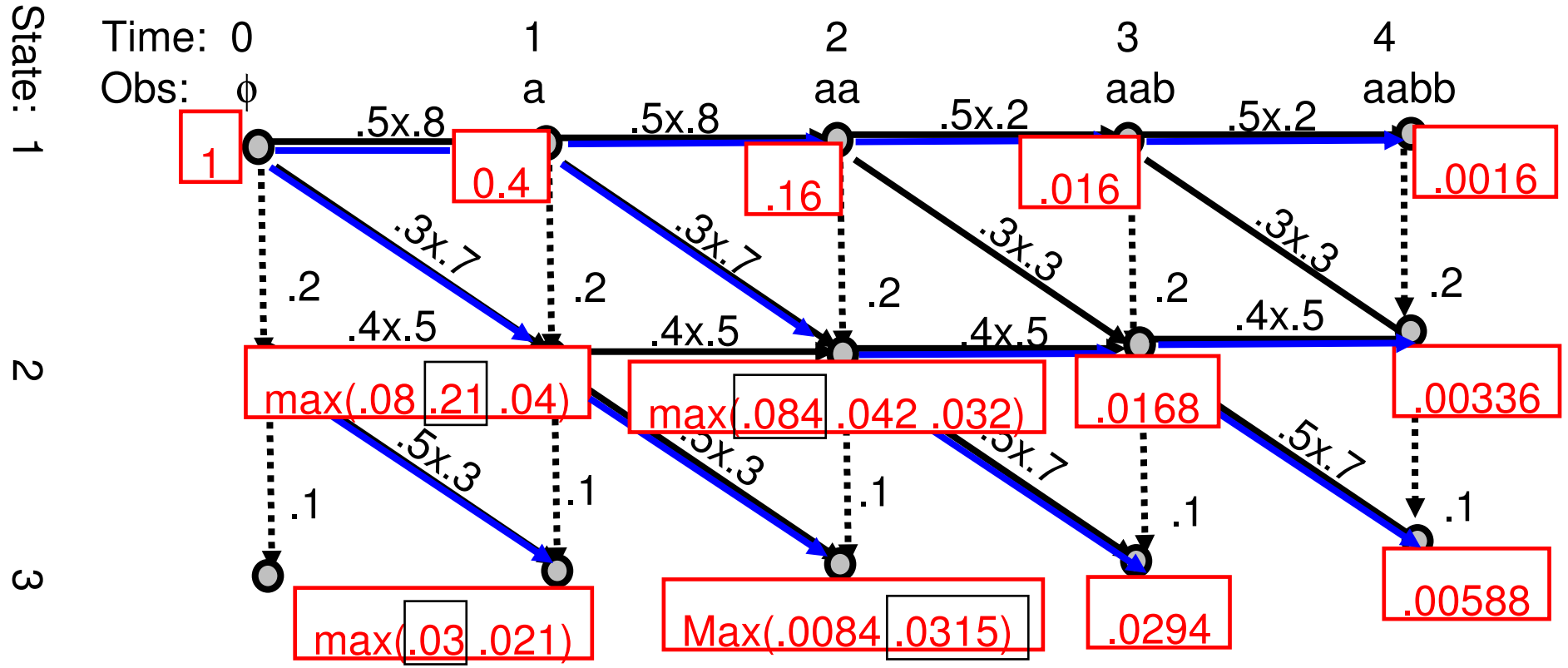
Preview:

The computation is similar to the forward algorithm, except we use max( ) instead of +

Also, we need to remember which partial path led to the max

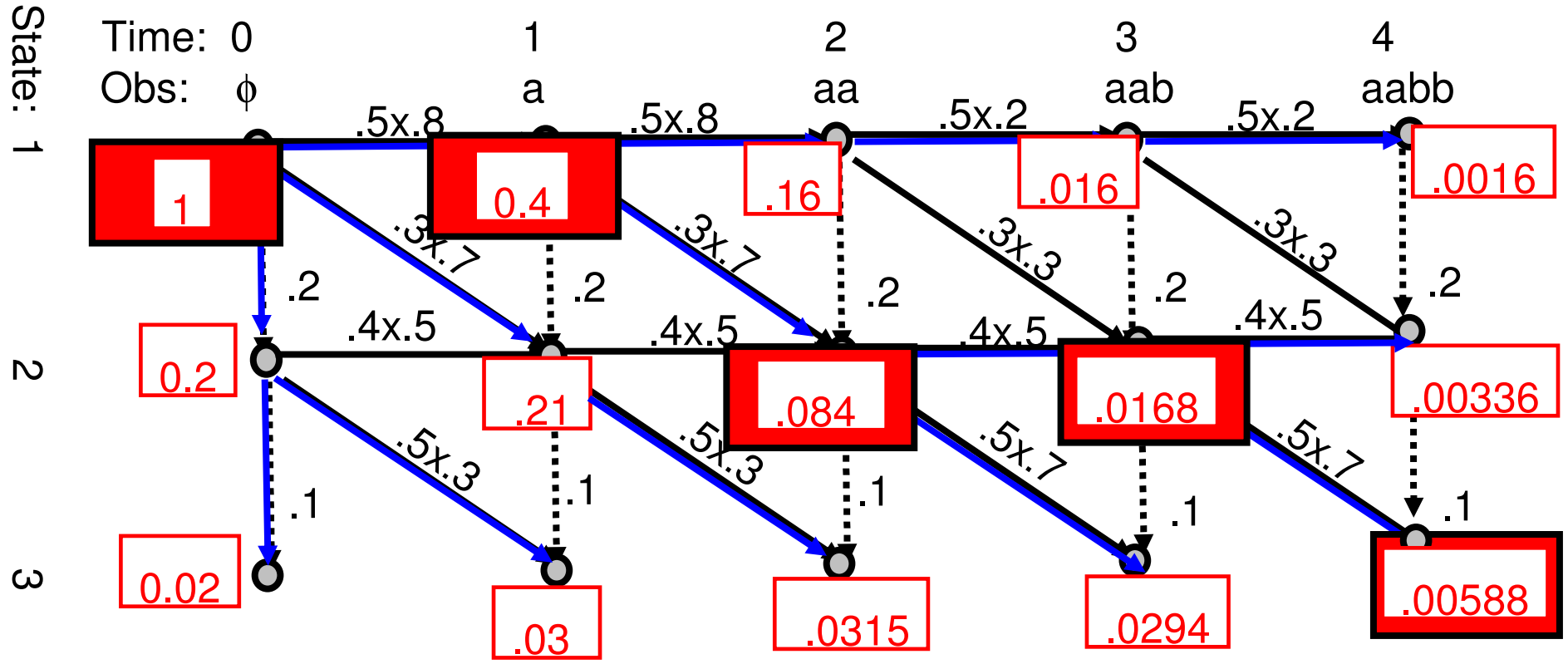
Problem 2: Viterbi algorithm

Returning to our example, let's find the most likely path for producing aabb. At each node, remember the max of predecessor score x transition probability. Also store the best predecessor for each node.



Problem 2: Viterbi algorithm, cont'd

Starting at the end, find the node with the highest score.
Trace back the path to the beginning, following best arc leading into each node along the best path.



Problem 3

Estimate the parameters of the model. (training)

- Given a model topology and an output sequence, find the transition and output probabilities such that the probability of the output sequence is maximized.

Generalization to Hidden MM case

State-observable

- Unique path
- Give a count of 1 to each transition along the path

Hidden states

- Many paths
- Assign a fractional count to each path
- For each transition on a given path, give the fractional count for that path
- Sum of the fractional counts =1
- How to assign the fractional counts??

How to assign the fractional counts to the paths

- Guess some values for the parameters
- Compute the probability for each path using these parameter values
- Assign path counts in proportion to these probabilities
- Re-estimate parameter values
- Iterate until parameters converge

Estimating Transition and Emission Probabilities

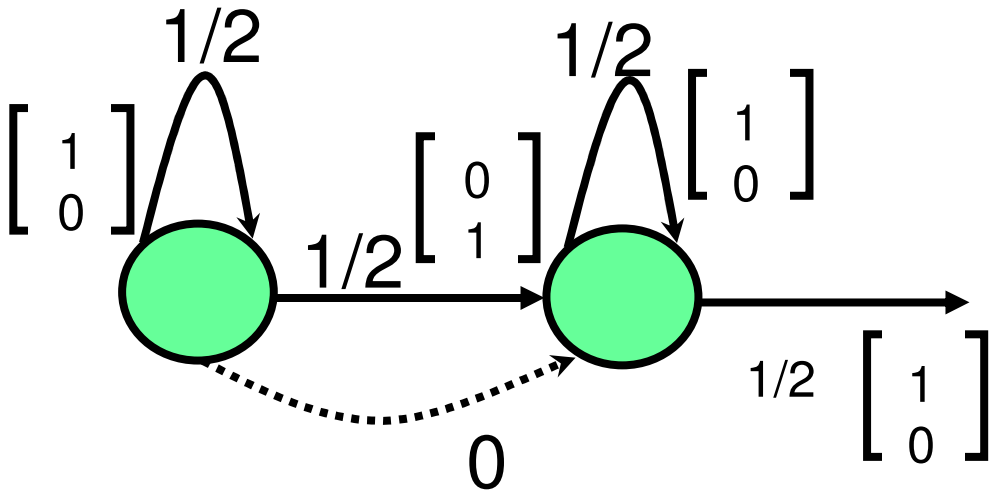
$$a_{ij} = \frac{count(i \rightarrow j)}{\sum_{q \in Q} count(i \rightarrow q)}$$

$$\hat{a}_{ij} = \frac{\text{Expected number of transitions from state i to j}}{\text{Expected number of transitions from state i}}$$

$$\hat{b}_j(x_t) = \frac{\text{Expected number of times in state j and observing symbol xt}}{\text{Expected number of time in state j}}$$

Problem 3: Enumerative Example

Step	Pr(X)
1	0.00914
2	0.02437
3	0.02507
10	0.04341
16	0.0625 converged



Problem 3: Parameter Estimation Performance

- The above re-estimation algorithm converges to a local maximum.
- The final solution depends on the starting point.
- The speed of convergence depends on the starting point.

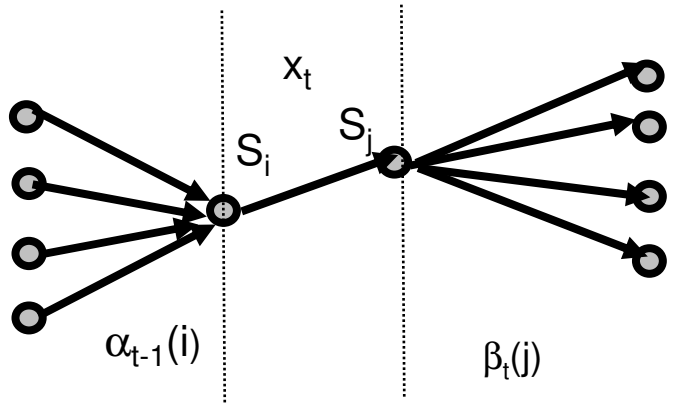
Problem 3: Forward-Backward Algorithm

- The forward-backward algorithm improves on the enumerative algorithm by using the trellis
- Instead of computing counts for each path, we compute counts for each transition at each time in the trellis.
- This results in the reduction from exponential computation to linear computation.

Problem 3: Forward-Backward Algorithm

Consider transition from state i to j, tr_{ij}

Let $p_t(tr_{ij},X)$ be the probability that tr_{ij} is taken at time t, and the complete output is X.



$$p_t(tr_{ij},X) = \alpha_{t-1}(i)\ a_{ij}\ b_{ij}(x_t)\ \ \beta_t(j)$$

Problem 3: F-B algorithm cont'd

$$p_t(tr_{ij},X) = \alpha_{t-1}(i)\ a_{ij}\ b_{ij}(x_t)\ \ \beta_t(j)$$

where:

$\alpha_{t-1}(i)$ = Pr(state=i, $x_1 \ldots x_{t-1}$) = probability of being in state i and having produced $x_1 \ldots x_{t-1}$

a_{ij} = transition probability from state i to j

$b_{ij}(x_t)$ = probability of output symbol x_t along transition ij

$\beta_t(j)$ = Pr($x_{t+1} \ldots x_T$|state= j) = probability of producing $x_{t+1} \ldots x_T$
given you are in state j

Problem 3: F-B algorithm cont'd

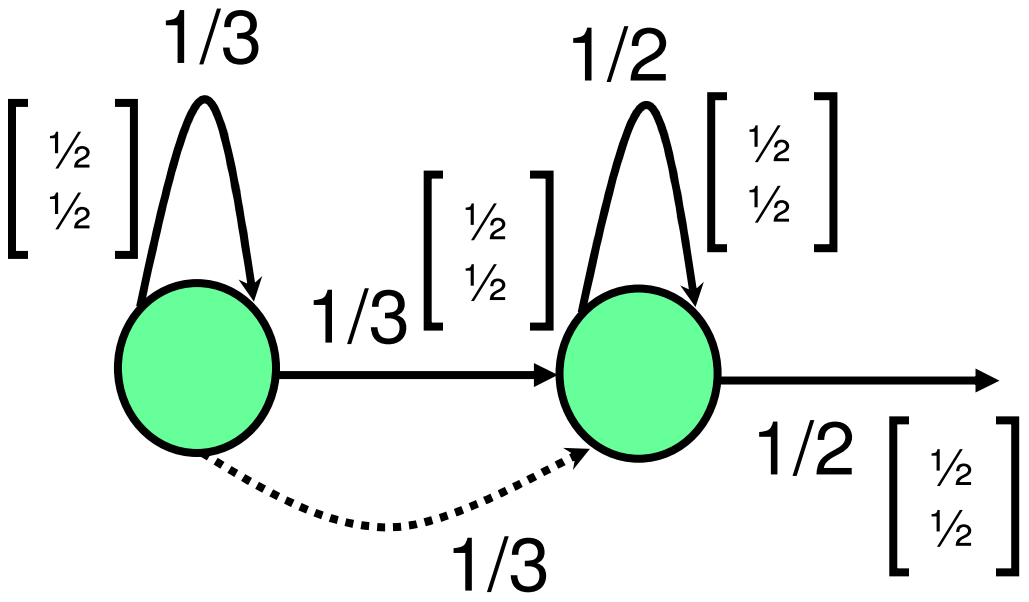
- Transition count $c_t(tr_{ij}|X) = p_t(tr_{ij},X) / Pr(X)$
- The β's are computed recursively in a backward pass (analogous to the forward pass for the α's)

$\beta_t(j) = \Sigma_k \beta_{t+1}(k) \; a_{jk} \; b_{jk}(x_{t+1})$ (for all output producing arcs)

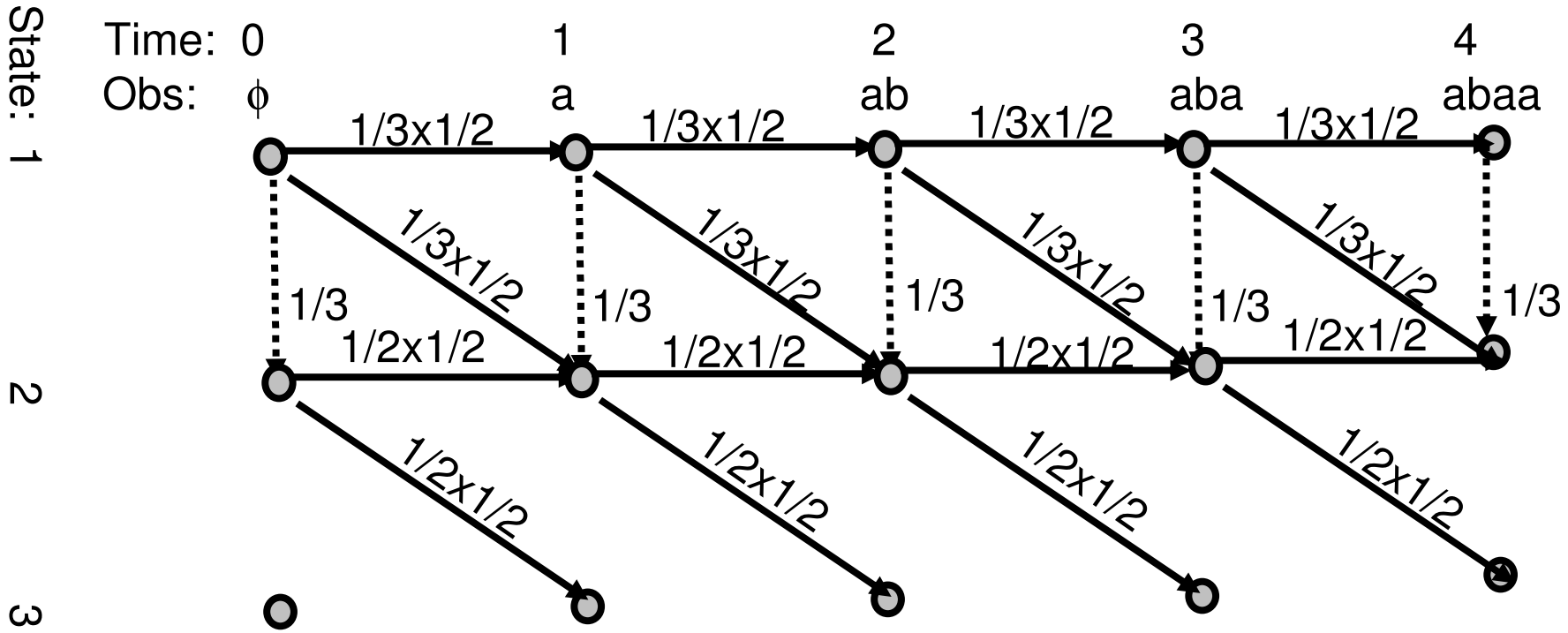
$+ \Sigma_k \beta_t(k) \; a_{jk}$ (for all null arcs)

Problem 3: F-B algorithm cont'd

- Let's return to our previous example, and work out the trellis calculations

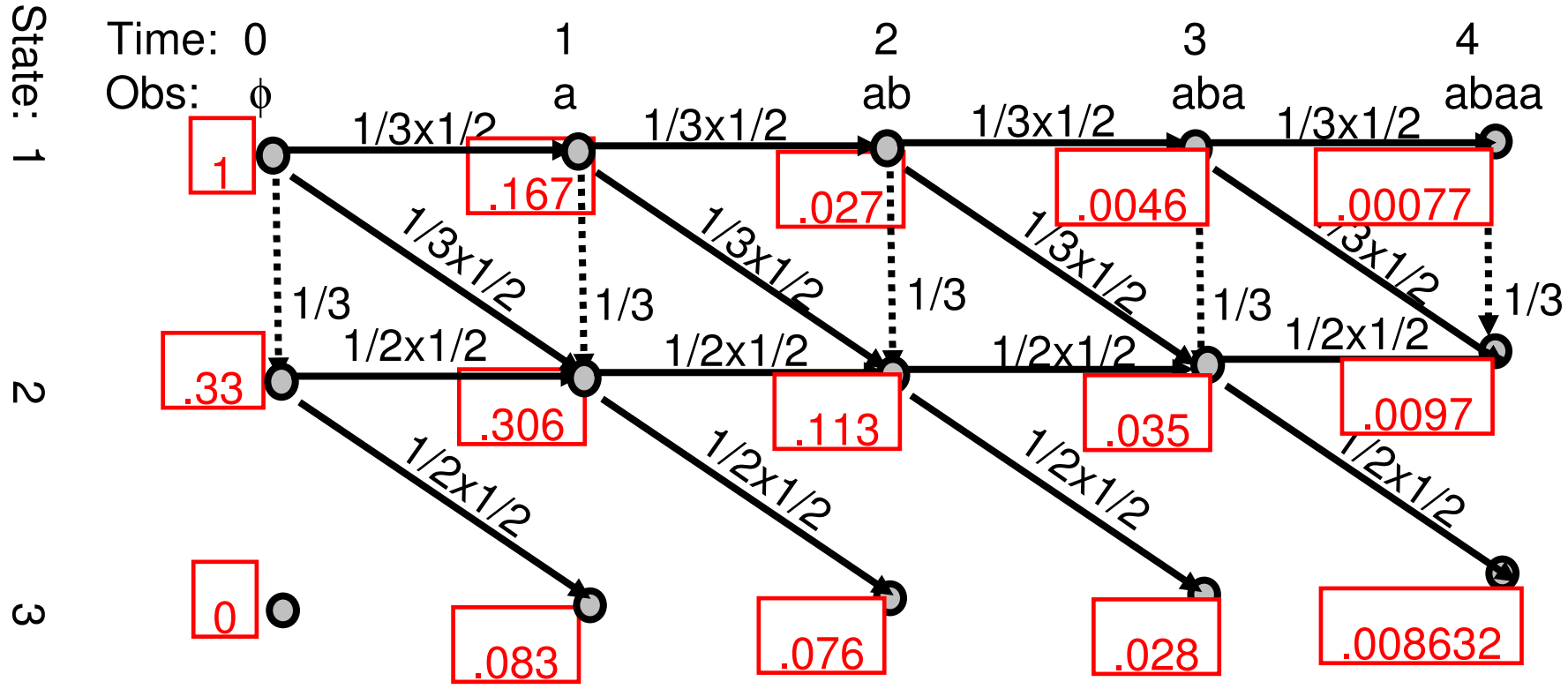


Problem 3: F-B algorithm, cont'd



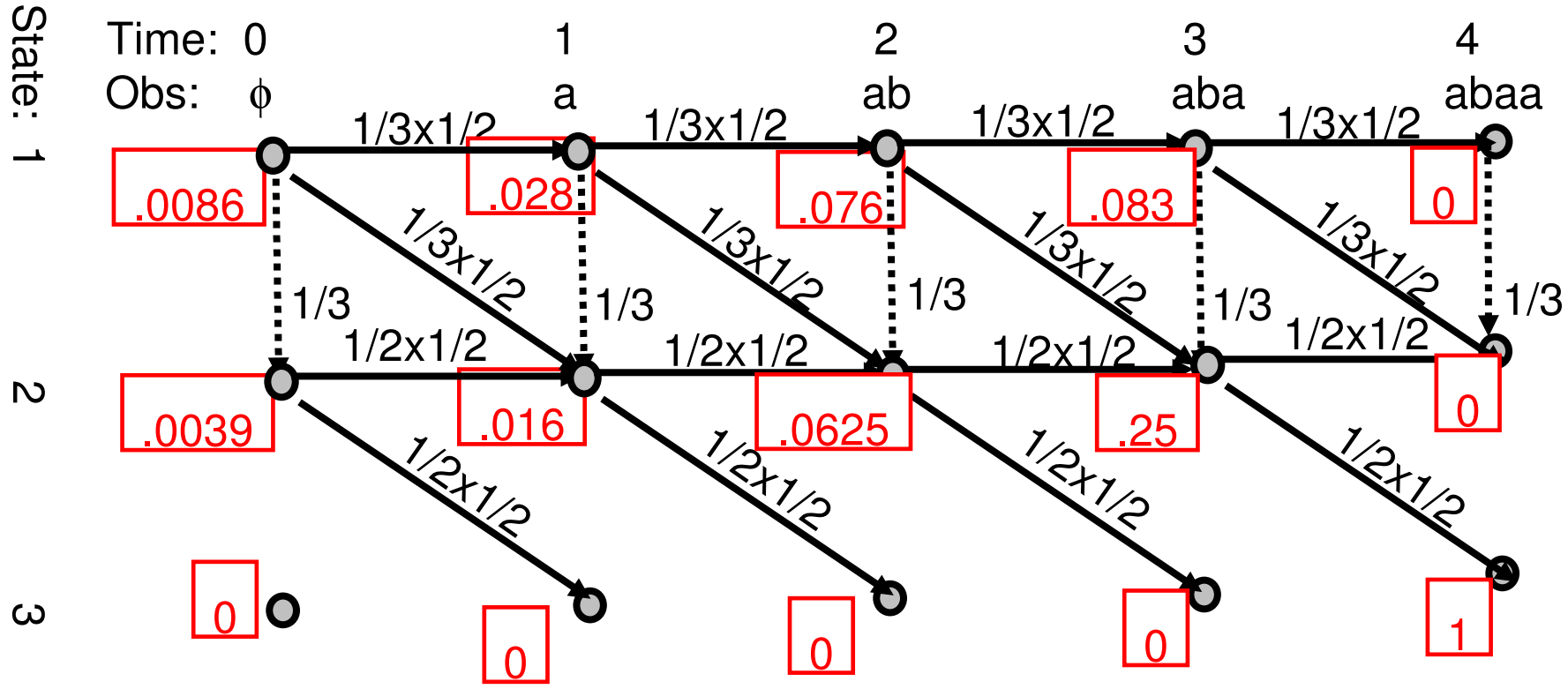
Problem 3: F-B algorithm, cont'd

Compute α's. since forced to end at state 3, α_T=.008632=Pr(X)



Problem 3: F-B algorithm, cont'd

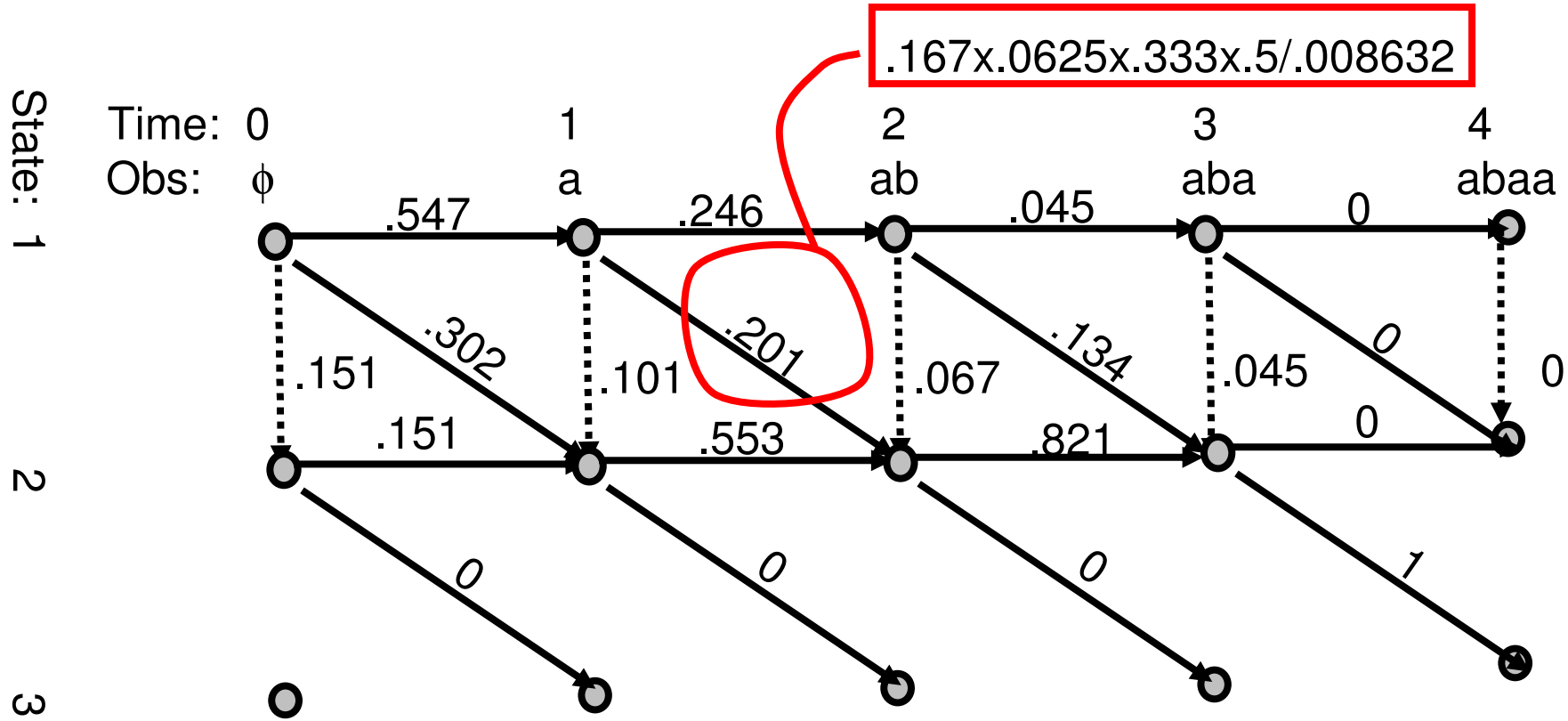
Compute β's.



Problem 3: F-B algorithm, cont'd

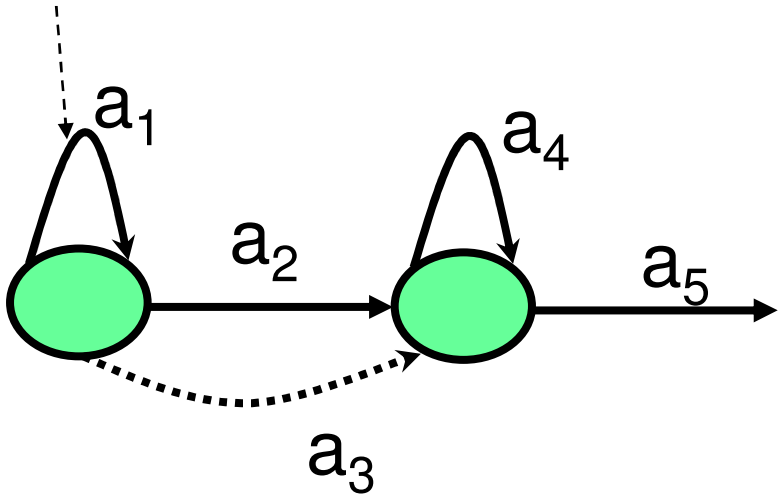
Compute counts. (a posteriori probability of each transition)

$$c_t(tr_{ij}|X) = \alpha_{t-1}(i)\, a_{ij}\, b_{ij}(x_t)\ \ \beta_t(j)/\ Pr(X)$$



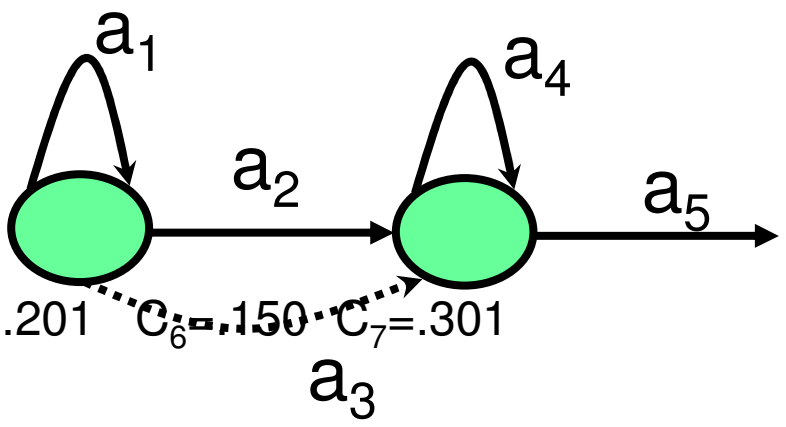
Problem 3: F-B algorithm cont'd

- $C(a_1)$=.547+.246+.045
- $C(a_2)$=.302+.201+.134
- $C(a_3)$=.151+.101+.067+.045
- $C(a_4)$=.151+.553+.821
- $C(a_5)$=1



- C(a1,'a')=.547+.045, C(a1,'b')=.246
- C(a2,'a')=.302+.134, C(a2,'b')=.201
- C(a4,'a')=.151+.821, C(a4,'b')=.553
- C(a5,'a')=1, C(a5,'b')=0

Remember : We Normalized Fractional Counts to get Transition and Observation Probabilities



- Let C_i be the a posteriori probability of path i
- $C_i = pr(X,path_i)/pr(X)$
-
- $C_1 = .045$ $C_2 = .067$ $C_3 = .134$ $C_4 = .100$ $C_5 = .201$ $C_6 = .150$ $C_7 = .301$

- $Count(a_1) = 3C_1+2C_2+2C_3+C_4+C_5 = .838$
- $Count(a_2) = C_3+C_5+C_7 = .637$
- $Count(a_3) = C_1+C_2+C_4+C_6 = .363$

a1= C(a1)/{C(a1) + C(a2) + C(a3)}

- New estimates:
- $a_1 = .46$ $a_2 = .34$ $a_3 = .20$

- $Count(a_1,'a') = 2C_1+C_2+C_3+C_4+C_5 = .592$ $Count(a_1,'b') = C_1+C_2+C_3 = .246$

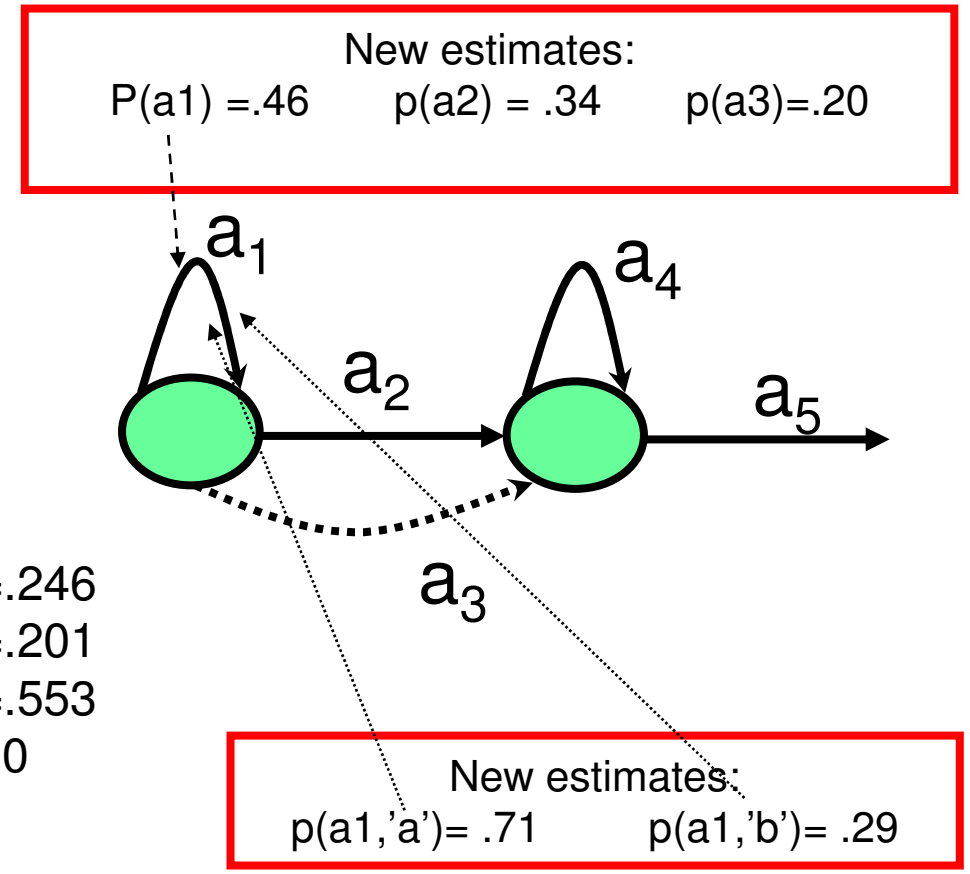
- New estimates:
- $p(a_1,'a') = .71$ $p(a_1,'b') = .29$

1st term 2C1 because in abaa, last 'a' by a5 so 2'a's in aba

Problem 3: F-B algorithm cont'd

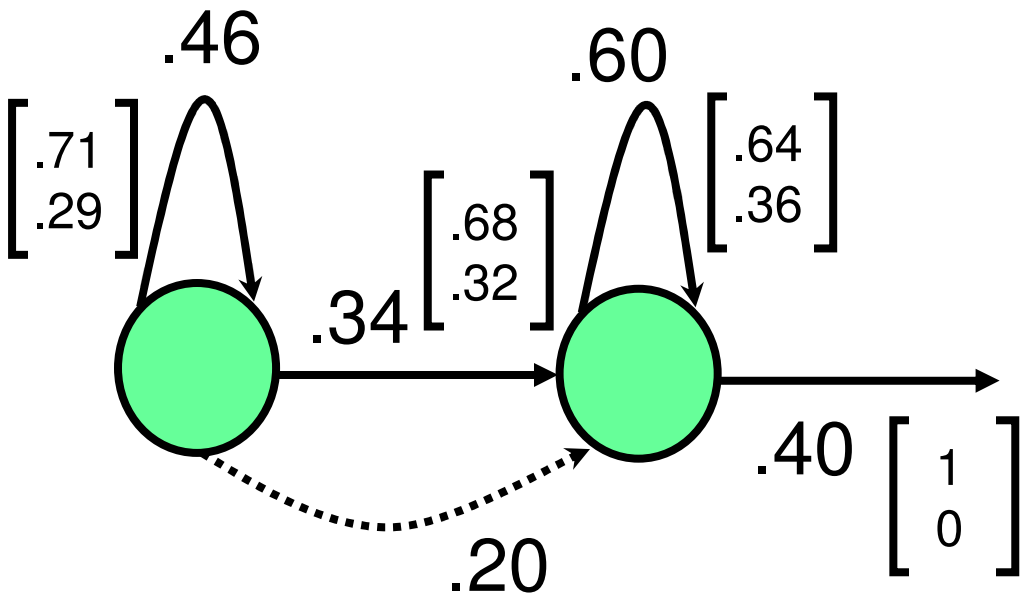
- $C(a_1)$=.547+.246+.045
- $C(a_2)$=.302+.201+.134
- $C(a_3)$=.151+.101+.067+.045
- $C(a_4)$=.151+.553+.821
- $C(a_5)$=1

- C(a1,'a')=.547+.045, C(a1,'b')=.246
- C(a2,'a')=.302+.134, C(a2,'b')=.201
- C(a4,'a')=.151+.821, C(a4,'b')=.553
- C(a5,'a')=1, C(a5,'b')=0



Problem 3: F-B algorithm cont'd

Normalize counts to get new parameter values.



Result is the same as from the enumerative algorithm!!

Summary of Markov Modeling Basics

- Key idea 1: States for modeling sequences

  Markov introduced the idea of state to capture the dependence on the past (time evolution). A state embodies all the relevant information about the past. Each state represents an equivalence class of pasts that influence the future in the same manner.

- Key idea 2: Marginal probabilities

  To compute Pr(X), sum up over all of the state sequences than can produce X

  $$\Pr(X) = \sum_s \Pr(X,S)$$

  For a given S, it is easy to compute Pr(X,S)

- Key idea 3: Trellis

  The trellis representation is a clever way to enumerate all sequences. It uses the Markov property to reduce exponential-time enumeration algorithms to linear-time trellis algorithms.

Topics for NLP on Large Data

- NLP problems with Big Data
- Machine Learning with Big Data
- MapReduce
- Hadoop
- Hadoop in Amazon Web Services
- Your account in AWS
- Running your Hadoop in AWS

Big Data : NLP

- NLP is a lot about processing text
  - Books
  - Webpages
  - Forums
  - Blogs
  - Twitter Messages
  - Comments
  - Reviews

Lot of Data Out There

- Google processes 24 PB a day
- Facebook has 2.5 PB of user data + 15 TB/day (4/2009)
- eBay has 6.5 PB of user data + 50 TB/day (5/2009)
- CERN has 200 petabytes of data from 800 trillion collisions
- AT&T transfers 30 petabytes a day

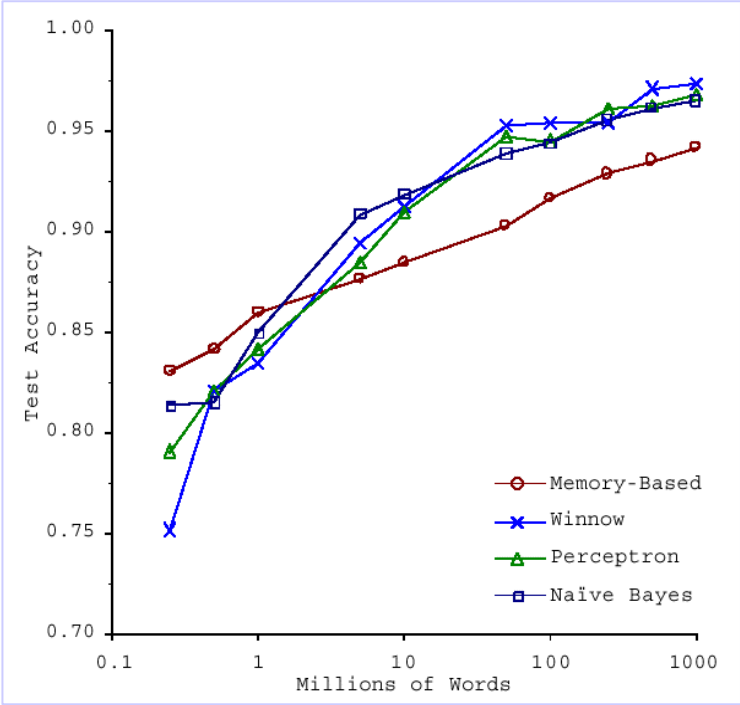
1 PB = 1 million GB = 1 thousand TB

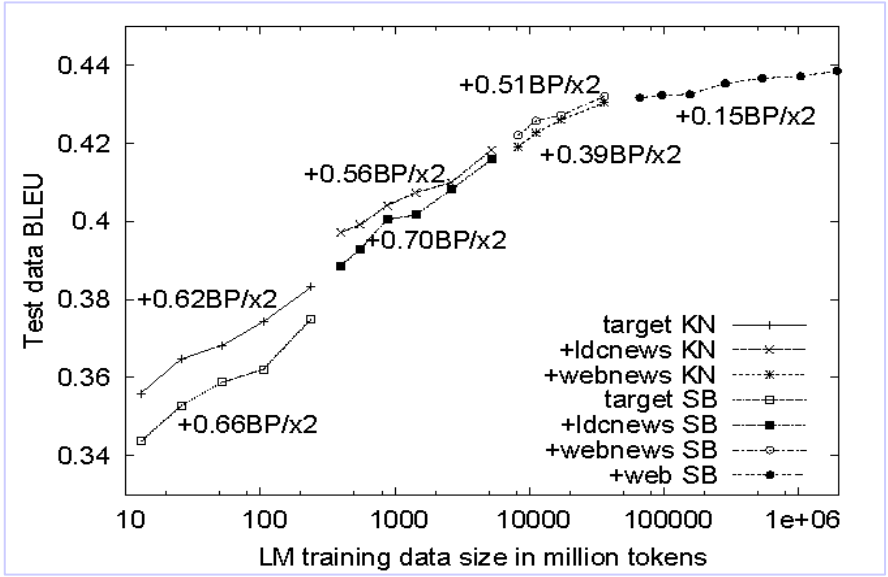
Use More Data for Statistical NLP

- We build a text classifier for Homework 1
- Small set of Hockey and Baseball documents

- Classifier accuracy may depend on
  - Modeling technique
  - Features
  - Data

Better Classifier with More Data





(Banko and Brill, ACL 2001)

(Brants et al., EMNLP 2007)

Naïve Bayes Classifier for Text

- Given the training data what are the parameters to be estimated?

$P(Y)$

Diabetes : 0.8
Hepatitis : 0.2

$P(X|Y_1)$

the: 0.001
diabetic : 0.02
blood : 0.0015
sugar : 0.02
weight : 0.018
…

$P(X|Y_2)$

the: 0.001
diabetic : 0.0001
water : 0.0118
fever : 0.01
weight : 0.008
…

Naïve Bayes Classifier for Text

$$P(Y = y_k | X_1, X_2, ..., X_N) = \frac{P(Y=y_k) P(X_1, X_2, .., X_N | Y=y_k)}{\sum_j P(Y=y_j) P(X_1, X_2, .., X_N | Y=y_j)}$$

$$= \frac{P(Y=y_k) \Pi_i P(X_i | Y=y_k)}{\sum_j P(Y=y_j) \Pi_i P(X_i | Y=y_j)}$$

$$Y \leftarrow argmax_{y_k} P(Y = y_k) \Pi_i P(X_i | Y = y_k)$$

Building Classifiers and Clustering Models from Data

- Building Naïve Bayes classifier required
  - Counting the documents word occur in
  - Dividing by the total number of document
- Building K-Means model
  - Count number of points that belong to each cluster
  - Re-estimate the means
- Building E-M model
  - Compute fractional counts for each point assigning fractional counts to each class
  - Re-estimate mean and variance
- Building HMM model
  - Count number of times State A transitions to State B
  - Count number of times each state produces Observation o_i
  - Estimate transition and emission matrices

Typical Large-Data Problem

- Iterate over a large number of items (lines, documents, records)
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

(Dean and Ghemawat, OSDI 2004)

Frame the Problem in Map Reduce

- Iterate over a large number of items (lines, documents, records)
- Extract something of interest from each

MAP

- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

REDUCE

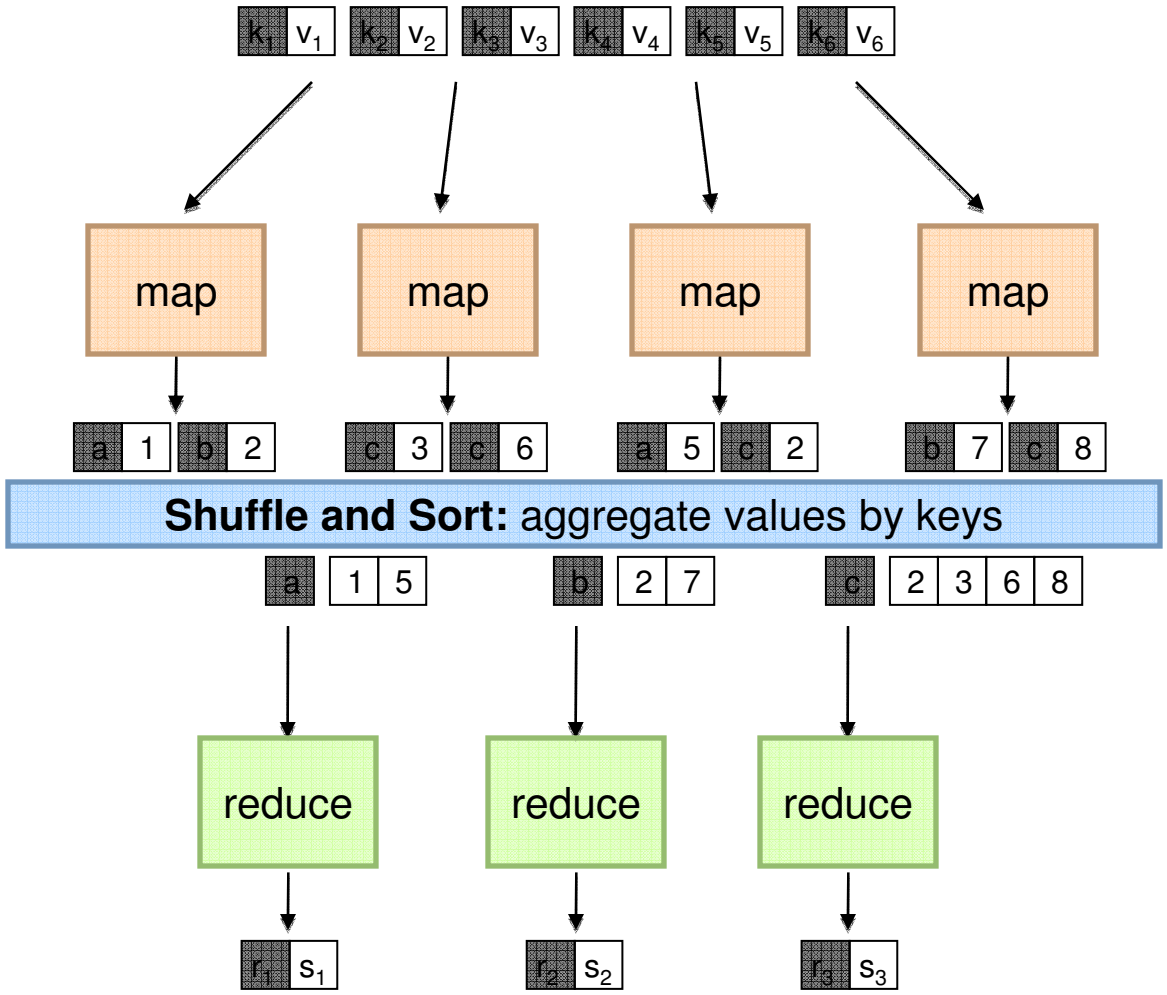
(Dean and Ghemawat, OSDI 2004)

MapReduce

- Programming model to handle “embarrassingly parallel” problems
- Programmers specify two functions:

Map (k1, v1) → list(k2, v2)
Applied to all pairs in parallel for input corpus
(e.g. all document for example)

Reduce (k2, list(v2)) → list (v3)
Applied to all groups with the same key in parallel
(e.g. all words for example)



Picture from [1]

MapReduce Framework

- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles “data distribution”
  - Moves processes to data
- Handles synchronization
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects worker failures and restarts
- Everything happens on top of a distributed FS

Naïve Bayes in MapReduce

- Assume you have 1 PB of data [1 million GB !]
- We want to build a Naïve Bayes Classifer
- We want to classify 500 TB of documents
- Is this embarrassingly parallel?

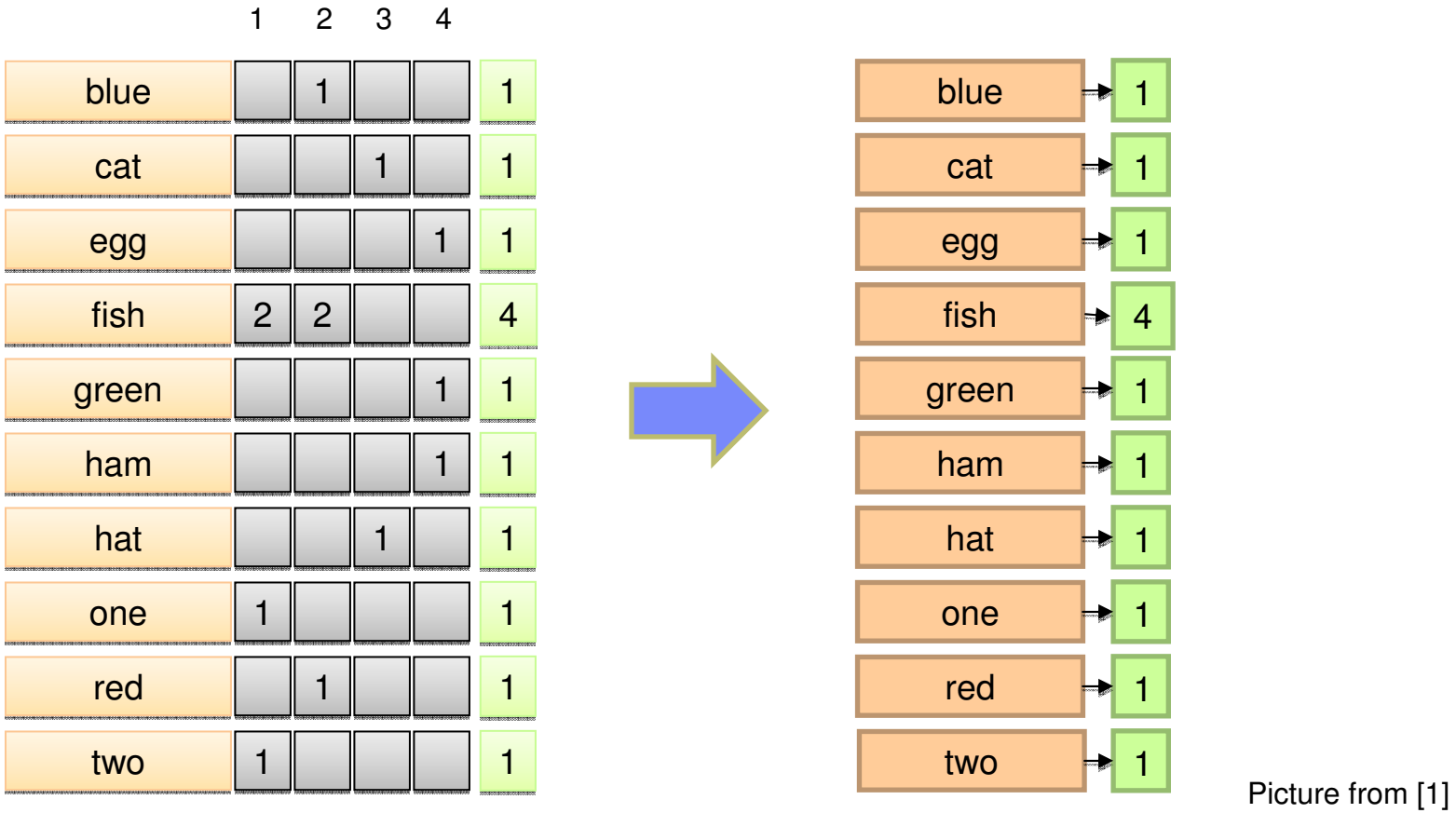
To Build Naïve Bayes We Need Counts

Doc 1 **one fish, two fish**

Doc 2 **red fish, blue fish**

Doc 3 **cat in the hat**

Doc 4 **green eggs and ham**



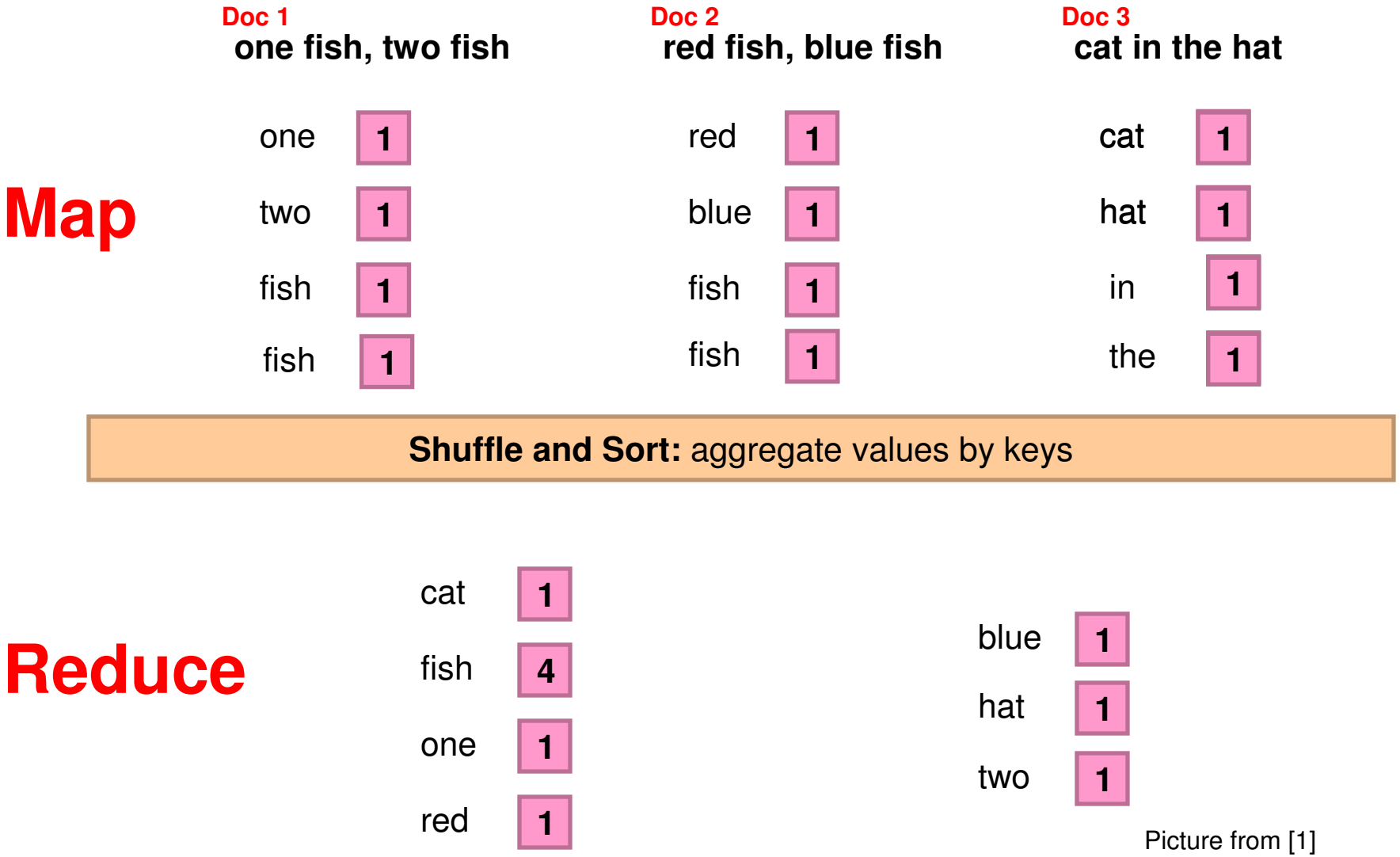
Picture from [1]

Word Count Example

```
Map(String docid, String text):
    for each word w in text:
        Emit(w, 1);
```

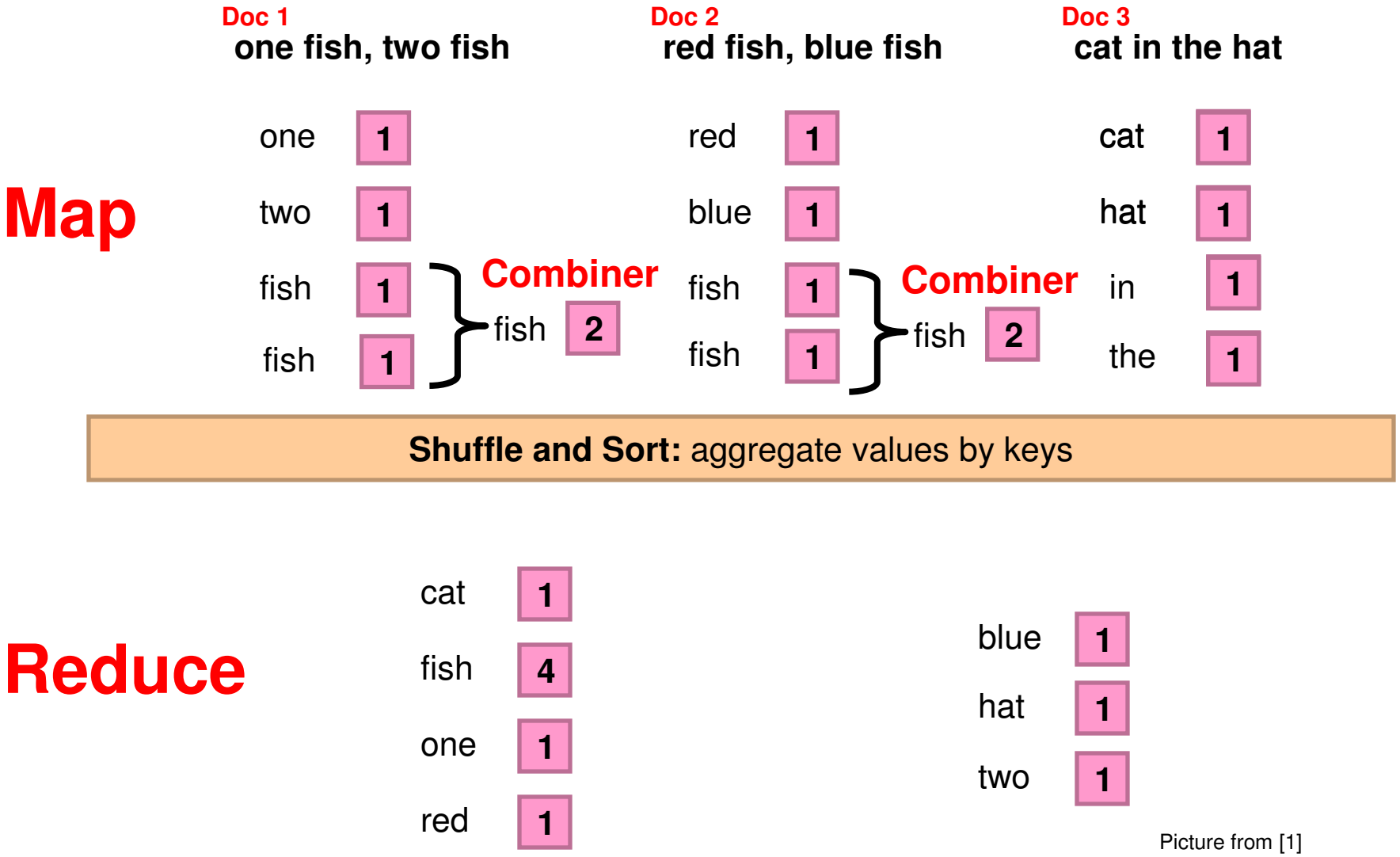
```
Reduce(String term, Iterator<Int> values):
    int sum = 0;
    for each v in values:
        sum += v;
        Emit(term, value);
```

MapReduce for Naïve Bayes



Picture from [1]

MapReduce for Naïve Bayes



Picture from [1]

Mapper : Word Count Example

- Map

```
lines = sys.stdin

for line in lines:
    words = line.split()
    for word in words:
        print word, "\t", 1;
```

Key = wordID
Value = 1

Reducer :Word Count Example

- Reduce

```
wordFreq={}

lines = sys.stdin

for line in lines:
    words = line.split("\t")
    key = words[0]
    val = words[1]
    if key in wordFreq:
        wordFreq[key] = wordFreq[key]+1
    else:
        wordFreq[key] = 1

for key in wordFreq.keys():
    print key, "\t", wordFreq[key]
```

Using AWS

- **https://stanlpcolumbia.signin.aws.amazon.com/console**

- UserID Password emailed to you
- Access Keys emailed you to as well

- LogIn
- Create Instance with LAMP stack AMI or whatever AMI you want
- Create certificates and store it in a safe place
- Access Virtual Server

- Create S3 bucket
- Try MapReduce Example

Reference

- http://www.cs.jhu.edu/~jason/papers/#tnlp02