AI6122 Text Data Management & Analysis

Topic: Exercise 1 Discussions

Q1.1 Regular Expression

- Write regular expressions for the following languages. By “word”, we mean an alphabetic string separated from other words by whitespace, any relevant punctuation, line breaks, and so forth.

1. The set of all lower case alphabetic strings ending with a letter *b*;

2. The set of all strings with two consecutive repeated words (e.g., “Humbert Humbert” and “the the” but not “the bug” or “the big bug”);

3. All strings that have both the word *grotto* and the word *raven* in them (but not, e.g., words like *grottos* that merely *contain* the word *grotto*);

Q1.1 Regular Expression

1. The set of all lower case alphabetic strings ending with a letter b;

   [a-z]*b\b

2. The set of all strings with two consecutive repeated words (e.g., “Humbert Humbert” and “the the” but not “the bug” or “the big bug”);

   ([a-zA-Z]+)\s+\1

Explanation

- [a-zA-Z]+ → all alphabetic strings
- \s → whitespace (space, tab..)
- \1 → used to refer to back to the first pattern in the expression which is put **inside a parentheses** ( )
- We may have \2 or \3 to refer to the second and third patterns put inside parentheses.

Q1.1 Regular Expression

3. All strings that have both the word `grotto` and the word `raven` in them (but not, e.g., words like `grottos` that merely contain the word `grotto`)

   (.*\bgrotto\b.*\braven\b.*)|(.*\braven\b.*\bgrotto\b.*)

   Explanation

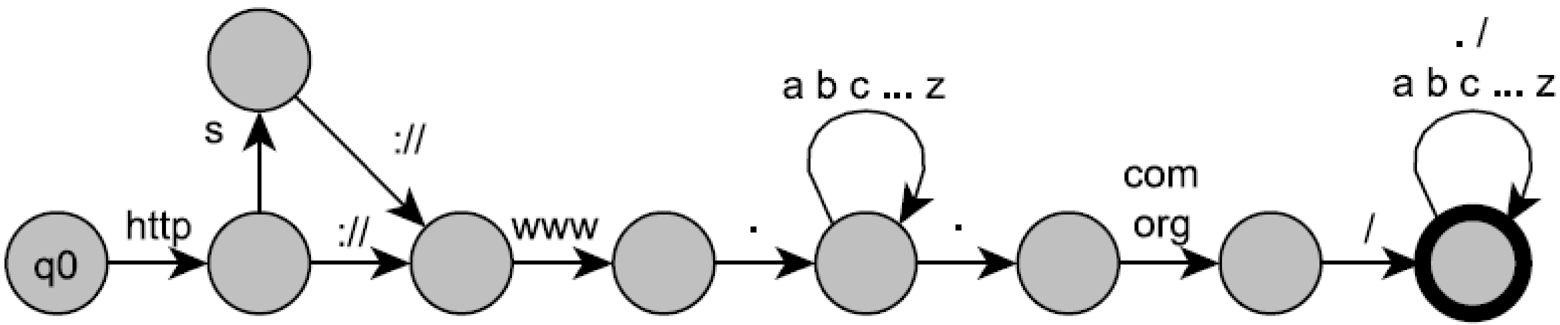
   – The two words grotto and raven may appear in any order.

   – There could be other strings around the two words

- http://regexr.com/

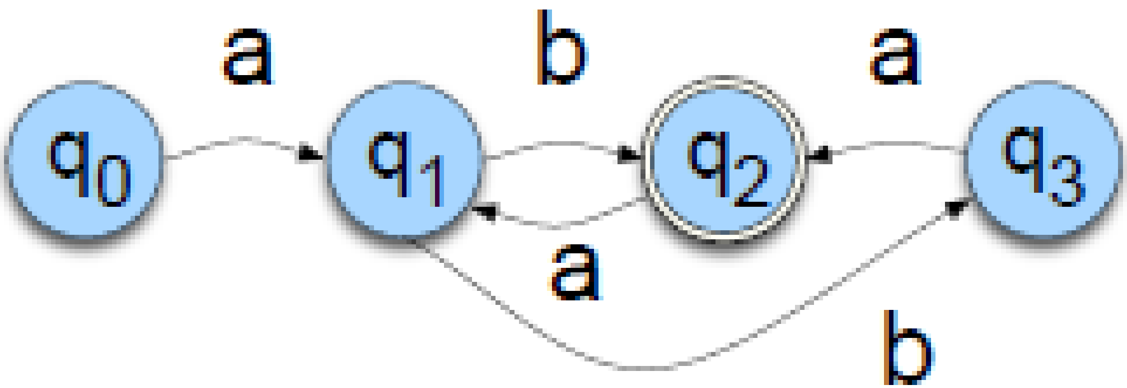
Q1.2 FSA

- Design an FSA that accepts a subset of valid web addresses. A commonly seen web address typically starts with “http” or “https”, followed by a “://www.” and name of the organization or company, then “.com/” or “.org/”. The address then has directory nesting like “abc/def/ghi”

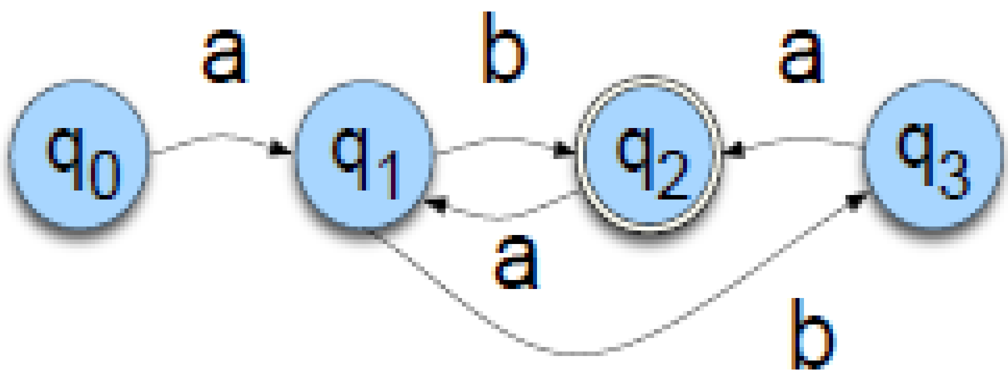


Q1.3 Regular Expression and FSA

- Write a regular expression for the language accepted by the following NFSA.



Q1.3 Regular Expression and FSA

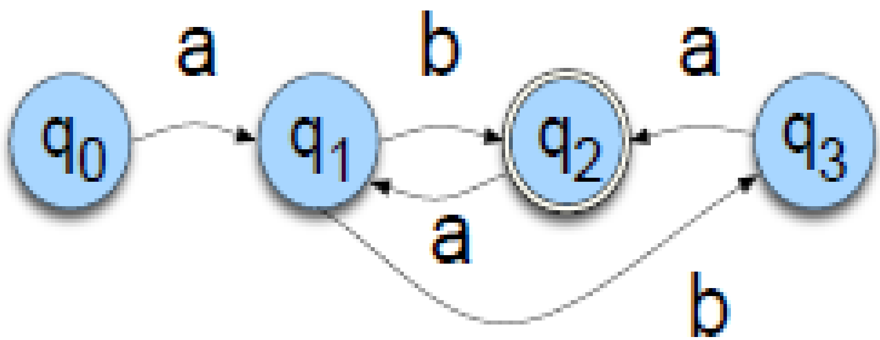


- **ab**
- **aba**
- **abab**
- **ababa**
- **ababab**
- **abaab**
- **…**

Hint: List the strings can be generated from the NFSA

Q1.3 Regular Expression and FSA

- **(aba?)+**



q_0 ::= ε

q_1 ::= q_0a | q_2a

q_1 ::= a | q_2a

q_2 ::= q_1b | q_3a

q_3 ::= q_1b

q_2 ::= q_1b | $q_1$ba

q_2 ::= ab | $q_2$ab | aba | $q_2$aba

q_2 ::= ab | aba | $q_2$ab | $q_2$aba

q_2 ::= (aba?)|($q_2$aba?)

q_2 ::= (aba?)+

Q1.4 Inverted Index

- You are given a collection of 4 documents. Draw the inverted index that would be built this document collection. State your assumptions about any preprocessing.

  – D1: School offers a new course
  – D2: A new course is offered
  – D3: The new course is good
  – D4: School offers many courses

Related topics?

Tokenization, Inverted Index

- Token and Tokenization
- Stop words
- English Morphology
- Stemming and Lemmatization
- Edit Distance

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Q1.4 Inverted Index

- **State your assumptions** about any preprocessing.
  - Case folding?
  - Stemming, lemmatization?
  - Stopword removal?

D1: School offers a new course
D2: A new course is offered
D3: The new course is good
D4: School offers many courses

- Preprocessing:
  - Case folding: School -> school
  - Lemmatization: offers, offered -> offer; courses -> course
  - Stopword removal: is, a, the

D1: school offer new course
D2: new course offer
D3: new course good
D4: school offer many course

Q1.4 Inverted Index

D1: school offer new course
D2: new course offer
D3: new course good
D4: school offer many course

- Inverted index:
  - Dictionary
  - Postings

Term	Document Freq	Pointer to postings	Postings
course	4	→	1, 2, 3, 4
good	1	→	3
many	1	→	4
new	3	→	1, 2, 3
offer	3	→	1, 2, 4
school	2	→	1, 4

Terms sorted

Doc Ids sorted

Q1.5 Index size estimation

- In ordinary English text, there are about 4.5 characters per word on average. After indexing, the average length of a dictionary word is 8 characters. Suppose an index has a dictionary with 100,000 words. Assume that a byte is the smallest storage unit and one character occupies one byte. Estimate the space usage in number of bytes of this dictionary by using each of the following two storage strategies.
  - Dictionary-as-a-string without blocking.
  - Dictionary-as-a-string, using blocked dictionary storage with block size of 8.

Q1.5 Index size estimation

- In ordinary English text, there are about 4.5 characters per word on average. After indexing, the average length of a dictionary word is 8 characters. Suppose an index has a dictionary with 100,000 words. Assume that a byte is the smallest storage unit and one character occupies one byte. Estimate the space usage in number of bytes of this dictionary by using each of the following two storage strategies.

- **Useful information**
  - In ordinary English text, there are about 4.5 characters per word on average.
  - After indexing, the **average length of a dictionary word is 8 characters**.
  - Suppose an index has a **dictionary with 100,000 words**.
  - Assume that a byte is the smallest storage unit and **one character occupies one byte**.

  - Two storage strategies

Related topic: Dictionary-as-a-String

- Store dictionary as a (long) string of characters:
  - Pointer to next word shows end of current word
  - Hope to save up to 60% of dictionary space.

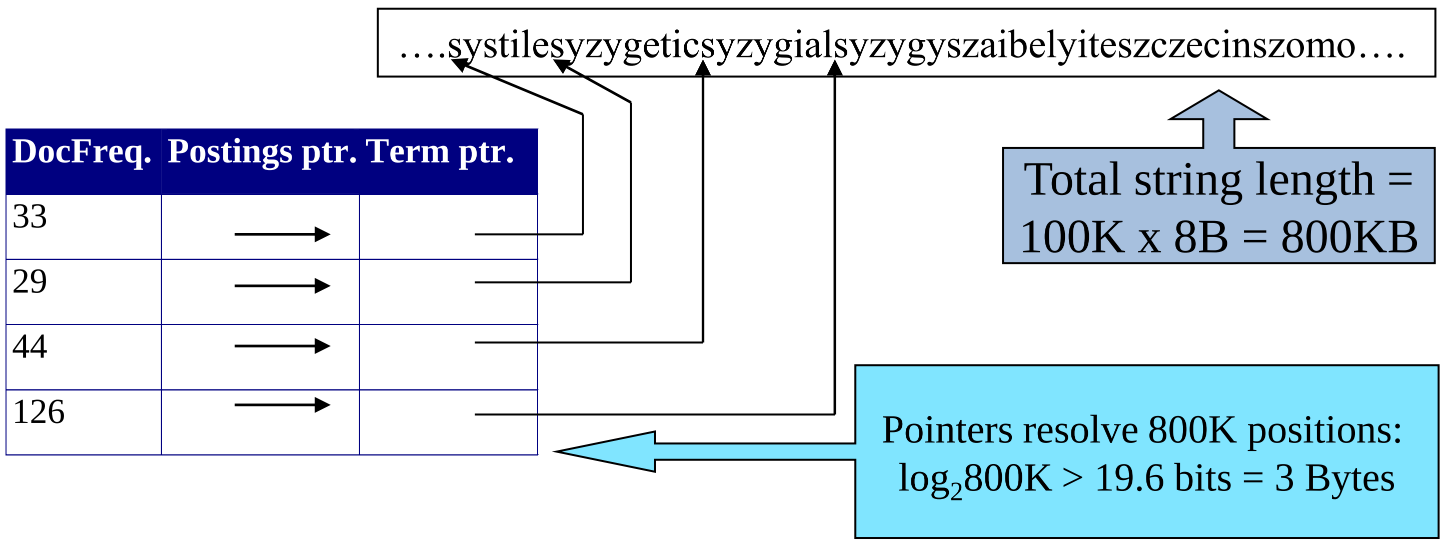
….systilesyzygeticsyzygialsyzygyszaibelyiteszczecinszomo….

DocFreq.	Postings ptr.	Term ptr.
33	→	→
29	→	→
44	→	→
126	→	→

Q1.5 Index size estimation

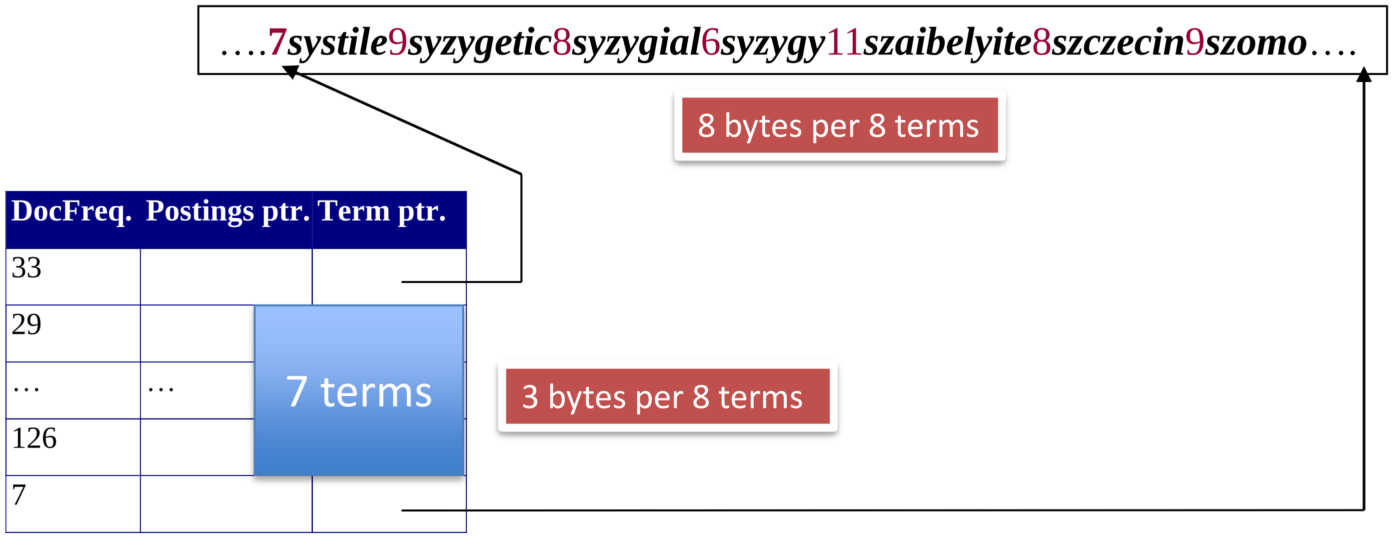
- Suppose an index has a dictionary with 100,000 words.
- Average length of a dictionary word is 8 characters.
  - Each words: 8B, total length of dictionary string: 800KB
  - Length of each address pointer: $\log_2 800K = 3\ Byte$
  - Integer for document frequency (4 Byte), Pointer to postings (4 Byte)

- Storage: (8 + 4 + 4 + 3) x 100,000 = 1900 KB = 1.9MB



Q1.5 Index size estimation

- Average length of a dictionary word is 8 characters.
- Using blocked dictionary storage with block size of 8.
  - One address pointer an 8-word block, 3 bytes
  - For each of 8 words, one byte for word length
- Storage: (8 + 4 + 4) x 100,000 + 11 x 100,000/8 = 1600 KB + 137.5KB= 1.7375 MB



Q1.6 Sentence level word co-occurrence

- Describe an approach to compute sentence-level word co-occurrences in a given document collection using the MapReduce framework.
- Your approach shall produce the output in the format of (w_a, w_b, n), where
  - w_a and w_b are two words,
  - n is the number of sentences that both words appear in.

- Question:
  - Does the order of (w_a, w_b) matter?

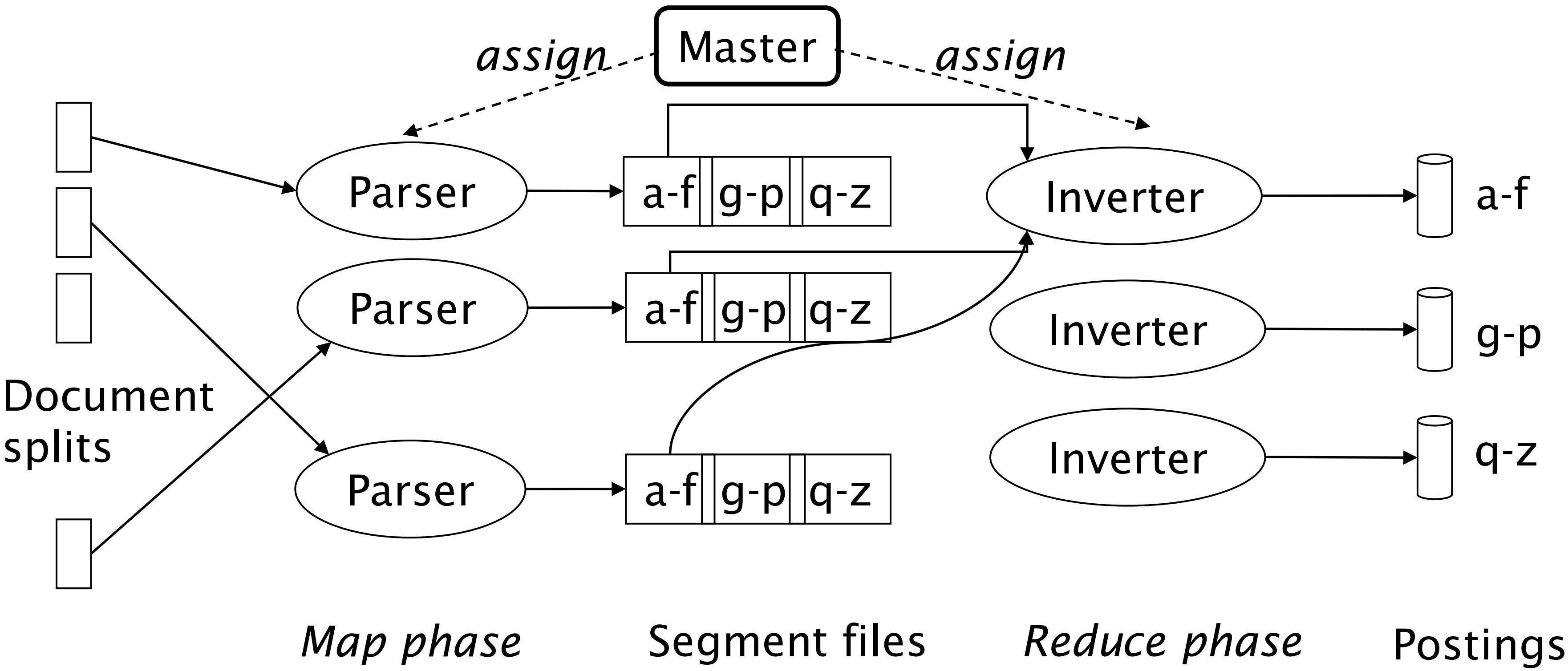
Review

Term-partitioned distributed indexing in parallel

- Maintain a **master** machine **directing** the indexing job
  - Break up indexing into sets of (parallel) tasks.
  - Master machine assigns each task to an idle machine from a pool.
- For indexing, we use two sets of parallel tasks
  - **Parsers**
  - **Inverters**
- Break the input document collection into splits
  - Each split is a subset of documents

Review

Term-partitioned distributed indexing: MapReduce



Review

Parsers and Inverters

- Master assigns a split of documents to an idle parser machine

- Parser
  - reads a document at a time, and emits <term, docID> pairs
  - Parser writes pairs into ***j* partitions,** each partition is for a range of terms' first letters (e.g., ***a-f***, ***g-p***, ***q-z***) – here *j* = 3.

- An inverter
  - collects all <term, docID> pairs for one term-partition (e.g., ***a-f***)
  - Sorts and writes to postings lists

Review

Schema for index construction in MapReduce

- MapReduce breaks a large problem into smaller parts using
  - key-value pairs (k, v)

- Schema of map and reduce functions
  - Map phase: input → list(k, v)
  - Reduce phase: (k,list(v)) → output

- Instantiation of the schema for **index construction**
  - map: collection → list(term, docID) Parser
  - reduce: (<term1, list(docID)>, <term2, list(docID)>, …) → (postings list1, postings list2, …) Inverter

Q1.6 Sentence level word co-occurrence

- Describe an approach to compute sentence-level word co-occurrences in a given document collection using the MapReduce framework.
- Your approach shall produce the output in the format of (w_a,w_b,n), where
  - w_a and w_b are two words,
  - n is the number of sentences that both words appear in.

- Question:
  - Does the order of (w_a,w_b) matter?
  - What is "key" here?
  - What is "value" here?

Q1.6 Sentence level word co-occurrence

- Master assigns a split of documents to an idle parser machine

- Parser
  - reads a document at a time, perform sentence segmentation, and emits <word1-word2, 1> pairs from each sentence
  - Word1 and word2 are sorted; and now word1-word2 serves as one term.
  - Parser writes pairs into ***j* partitions,** each partition is for a range of terms' first letters (e.g., ***a-f***, ***g-p***, ***q-z***) – here *j* = 3.

- An inverter
  - collects all <term, 1> pairs for one term-partition (e.g., ***a-f***)
  - merge occurrence numbers based by terms and write out (w_a,w_b,n),

Q1.7 Bigram

- Given the following three word sequences (i.e., the corpus).

  – *very good tennis player in US open*
  – *tennis player US Open*
  – *tennis player qualify play US Open*

  (i) Build a table of bigram counts from the word sequences.
  (ii) Compute the bigram probabilities using Laplace smoothing.

Review

Bigram Counts

- Out of 9222 sentences
  - Eg. “I want” occurred 827 times

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Review

Laplace-Smoothed Bigram Probabilities

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Q1.7 Bigram

- Given the corpus, build a table of bigram counts from the word sequences
  - very good tennis player in US open
  - tennis player US Open
  - tennis player qualify play US Open

	very	good	tennis	player	in	us	open	qualify	play
very	0	1	0	0	0	0	0	0	0
good	0	0	1	0	0	0	0	0	0
tennis	0	0	0	3	0	0	0	0	0
player	0	0	0	0	1	1	0	1	0
in	0	0	0	0	0	1	0	0	0
us	0	0	0	0	0	0	3	0	0
open	0	0	0	0	0	0	0	0	0
qualify	0	0	0	0	0	0	0	0	1
play	0	0	0	0	0	1	0	0	0

Compute the bigram probabilities using Laplace smoothing

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

- Unigram counting
  - *very 1 good 1 tennis 3 player 3 in 1 US 3 open 3 qualify 1 play 1*

	very	good	tennis	player	in	US	open	qualify	play
very	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
good	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1
tennis	1/12	1/12	1/12	4/12	1/12	1/12	1/12	1/12	1/12
player	1/12	1/12	1/12	1/12	2/12	2/12	1/12	2/12	1/12
in	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1
US	1/12	1/12	1/12	1/12	1/12	1/12	4/12	1/12	1/12
open	1/12	1/12	1/12	1/12	1/12	1/12	1/12	1/12	1/12
qualify	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2
play	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1

Q1.8 Viterbi Algorithm

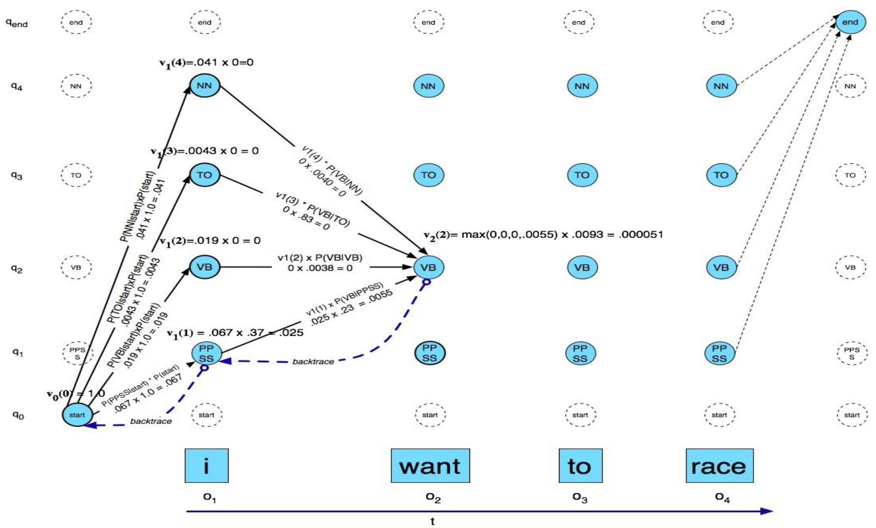
- Finish the computation of the Viterbi algorithm in the Viterbi example used in class for HMM. The transition probability and word likelihood probabilities are in the following tables.

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

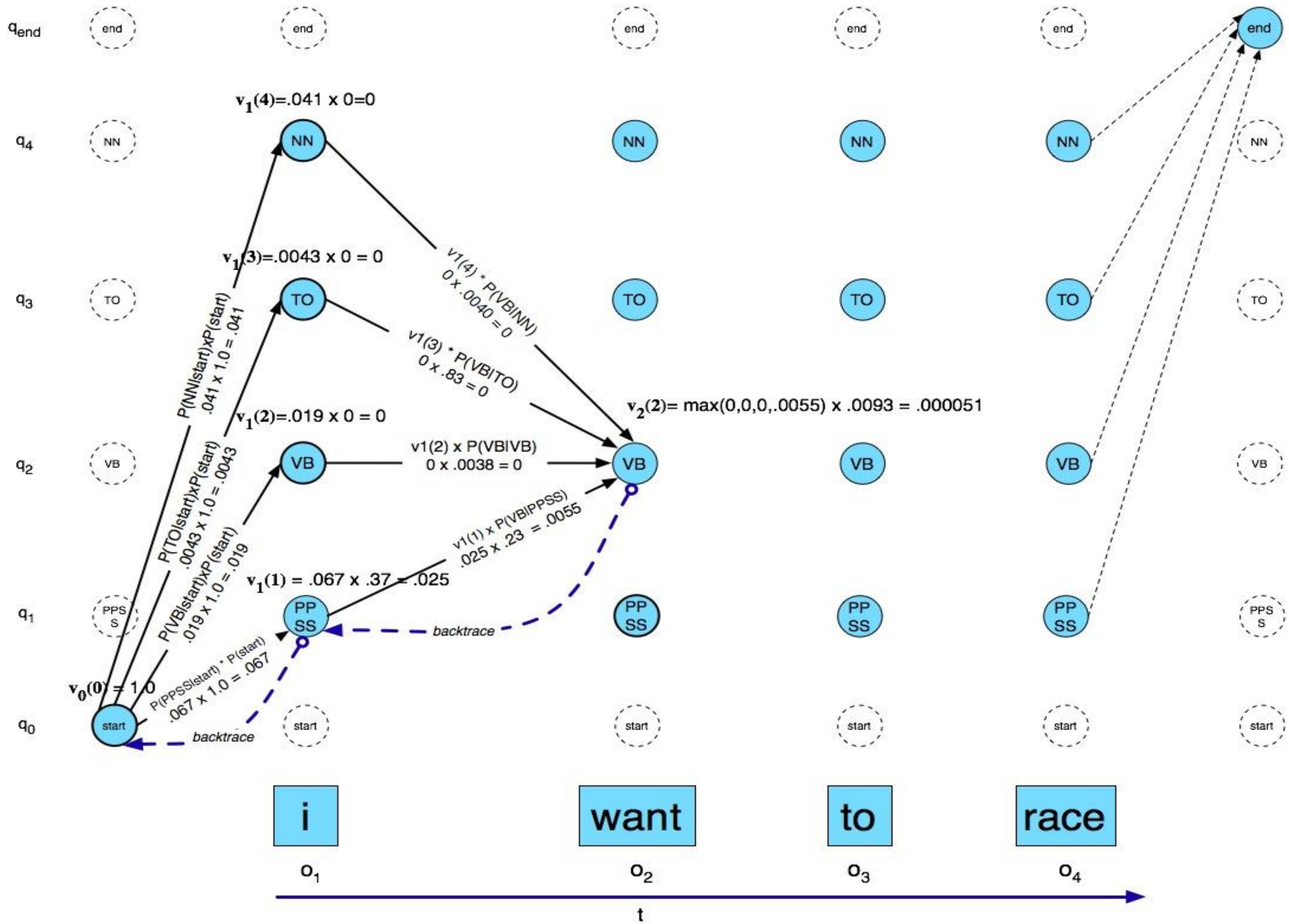
	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

Main Idea

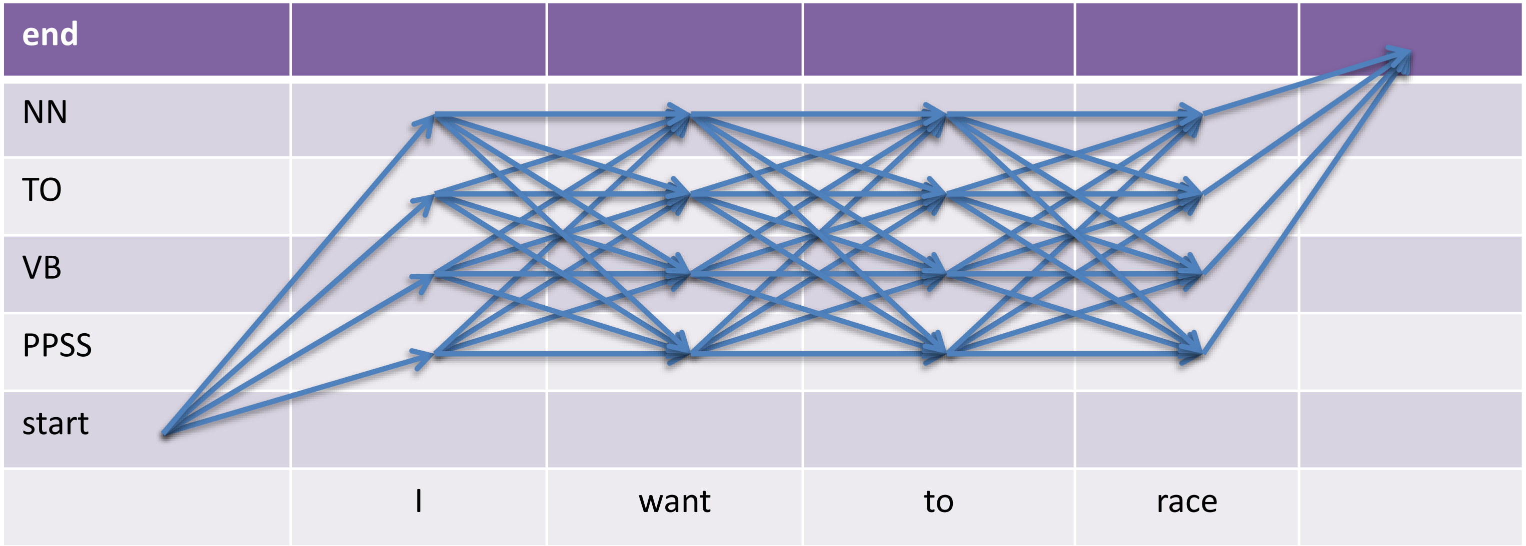
- We also have a matrix.
  - Each column– a time ‘t’ (observation)
  - Each row – a state ‘i’
  - For each cell $v_t[i]$, we compute the probability of the **best path** to the cell
- the **Viterbi path probability** at time t for state i
  - there are $|Q|$ number of paths from $t-1$ to $v_t[i]$
  - if we know **the best path** to each cell in $t-1$, or $v_{t-1}[j]$
  - $\arg\max_j v_{t-1}[j] \times P(i|j) \times P(s_t|i)$



Viterbi Example



Required computations



(This figure does not show the backtrace pointers)

Computation

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

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

end					
NN	$p(NN\| <s>) * p(I\|NN) = 0$				
TO	$p(TO\| <s>) * p(I\|TO) = 0$				
VB	$p(VB\| <s>) * p(I\|VB) = 0$				
PPSS	$p(PPSS\| <s>) * p(I\|PPSS) = 0.067 * 0.37 = 0.02479$				
start					
	I	want	to	race	

Computation

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

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

end					
NN	0	$.02479 * p(NN\|PPSS) * p(want\|NN) =$ $.02479 * .0012 * .000054 =$ 0.00000000160639			
TO	0	$.02479 * p(TO\|PPSS) * p(want\|TO) = 0$			
VB	0	$.02479 * p(VB\|PPSS) * p(want\|VB) =$ $.02479 * .23 * .0093 =$ 0.00005302581			
PPSS	0.02479	$.02479 * p(PPSS\|PPSS) * p(want\|PPSS) = 0$			
start					
	I	want	to	race	

Computation

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

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

end					
NN	0	1.6*10e-9	$\max(1.6*10e-9 * p(NN\|NN), 5.3*10e-5 * p(NN\|VB)) * p(to\|NN) = 0$		
TO	0	0	$\max\big(1.6*10e-9 * p(TO\|NN), \quad 5.3*10e-5 * p(TO\|VB)\big) * p(to\|TO) =$ $\max(1.6*10e-9 * .016, 5.3*10e-5 * .035) * .99$ $= 1.84*10e-6$		
VB	0	5.3*10e-5	$\max(1.6*10e-9 * p(VB\|NN), 5.3*10e-5 * p(VB\|VB)) * p(to\|VB) = 0$		
PPSS	0.02479	0	$\max(1.6*10e-9 * p(PPSS\|NN), 5.3*10e-5 * p(PPSS\|VB)) * p(to\|PPSS) = 0$		
start					
	I	want	to	race	

Computation

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

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

end					
NN	0	1.6*10e-9	0	$1.84*10e-6*\ p(NN\|TO)\ *\ p(race\|NN) =$ $1.84*10e-6*\ .00047\ *\ .00057 =\ 4.92*10e-14$	
TO	0	0	1.84*10e-6	$1.84*10e-6*\ p(TO\|TO)\ *\ p(race\|TO) = 0$	
VB	0	5.3*10e-5	0	$1.84*10e-6\ *\ p(VB\|TO)\ *\ p(race\|VB) =$ $1.84*10e-6*\ .83\ *\ .00012 =\ 1.83*10e-10$	
PPSS	0.02479	0	0	$1.84*10e-6*\ p(PPSS\|TO)\ *\ p(race\|PPSS) = 0$	
start					
	I	want	to	race	

Computation

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

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

end					
NN	0	1.6*10e-9	0	$1.84 * 10e-6 * p(NN\|TO) * p(race\|NN) =$ $1.84 * 10e-6 * .00047 * .00057 = 4.92 * 10e-14$	
TO	0	0	1.84*10e-6		
VB	0	5.3*10e-5	0	$1.84 * 10e-6 * p(VB\|TO) * p(race\|VB) =$ $1.84 * 10e-6 * .83 * .00012 = 1.83 * 10e-10$	
PPSS	0.02479	0	0		
start					
	I	want	to	race	

Computation

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

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

end					
NN	0	1.6*10e-9	0	$1.84*10e-6*\ p(NN\|TO)\ *\ p(race\|NN) =$ $1.84*10e-6*\ .00047\ *\ .00057 =\ 4.92*10e-14$	
TO	0	0	1.84*10e-6		
VB	0	5.3*10e-5	0	$1.84*10e-6\ *\ p(VB\|TO)\ *\ p(race\|VB) =$ $1.84*10e-6*\ .83\ *\ .00012 =\ \mathbf{1.83*10e-10}$	
PPSS	0.02479	0	0		
start					
	I	want	to	race	