Speech recognition and synthesis

1. More about ASR
   - Introduction
   - Dynamic programming
   - Viterbi algorithm
   - Other approaches to decoding
   - Training acoustic models
   - FLOSS resources
   - Assignment
   - Bibliography

Introduction

Two technologies are needed to make the HMM framework practical

- Decoder technology to find the $\underset{Words}{argmax}\ P(Observation|Words) \cdot P(Words)$
- Determining the stochastic parameters of the HMM state automaton, i.e. training

Many pictures (and their copyrights) are from [Jurafsky and Martin(2000)]

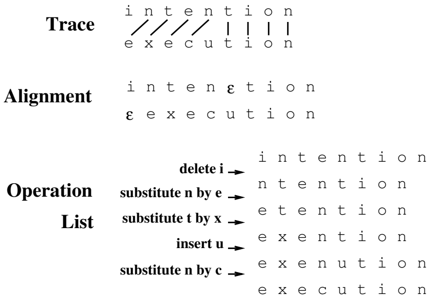
Introduction

Two technologies are needed to make the HMM framework practical

- Decoder technology to find the $\underset{Words}{argmax}\ P(Observation|Words) \cdot P(Words)$
- Determining the stochastic parameters of the HMM state automaton, i.e. training

Many pictures (and their copyrights) are from [Jurafsky and Martin(2000)]

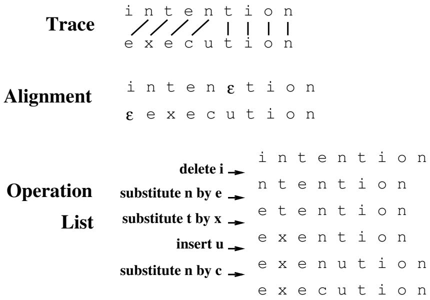
Dynamic programming



Look for best alignment: Minimum edit distance

- Delete
- Insert
- Substitute

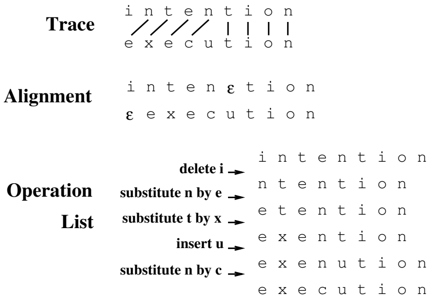
Dynamic programming



Look for best alignment: Minimum edit distance

- Delete
- Insert
- Substitute

Dynamic programming



Look for best alignment: Minimum edit distance

- Delete
- Insert
- Substitute

Dynamic programming

```
function MIN-EDIT-DISTANCE(target, source) returns min-distance

  n ← LENGTH(target)
  m ← LENGTH(source)
  Create a distance matrix distance[n+1,m+1]
  distance[0,0] ← 0
  for each column i from 0 to n do
     for each row j from 0 to m do
        distance[i, j] ← MIN( distance[i−1, j] + ins-cost(target_i),
                              distance[i−1, j−1] + subst-cost(source_j, target_i),
                              distance[i, j−1] + del-cost(source_j))
```

Fill a matrix with cumulative edit distances, $distance[i, j]$ = min of

- $distance[i-1, j]$ + insert-cost($target_i$)
- $distance[i-1, j-1]$ + substitution-cost($source_j$, $target_i$)
- $distance[i, j-1]$ + deletion-cost($source_j$)

Dynamic programming

```
function MIN-EDIT-DISTANCE(target, source) returns min-distance

  n ← LENGTH(target)
  m ← LENGTH(source)
  Create a distance matrix distance[n+1,m+1]
  distance[0,0] ← 0
  for each column i from 0 to n do
     for each row j from 0 to m do
        distance[i, j] ← MIN( distance[i−1,j] + ins-cost(target_i),
                              distance[i−1,j−1] + subst-cost(source_j, target_i),
                              distance[i,j−1] + del-cost(source_j))
```

Fill a matrix with cumulative edit distances, $distance[i,j] =$ min of

- $distance[i-1,j] + \text{insert-cost}(target_i)$
- $distance[i-1,j-1] + \text{substitution-cost}(source_j, target_i)$
- $distance[i,j-1] + \text{deletion-cost}(source_j)$

Dynamic programming

function MIN-EDIT-DISTANCE(*target*, *source*) **returns** *min-distance*

```
n ← LENGTH(target)
m ← LENGTH(source)
Create a distance matrix distance[n+1,m+1]
distance[0,0] ← 0
for each column i from 0 to n do
   for each row j from 0 to m do
      distance[i,j] ← MIN( distance[i−1,j] + ins-cost(target_i),
                           distance[i−1,j−1] + subst-cost(source_j, target_i),
                           distance[i,j−1] + del-cost(source_j))
```

Fill a matrix with cumulative edit distances, $distance[i,j]$ = min of

- $distance[i-1,j]$ + insert-cost($target_i$)
- $distance[i-1,j-1]$ + substitution-cost($source_j$, $target_i$)
- $distance[i,j-1]$ + deletion-cost($source_j$)

Dynamic programming

n	9	10	11	10	11	12	11	10	9	**8**
o	8	9	10	9	10	11	10	9	**8**	9
i	7	8	9	8	9	10	9	**8**	9	10
t	6	7	8	7	8	9	**8**	9	10	11
n	5	6	7	6	7	**8**	9	10	11	12
e	4	5	6	**5**	**6**	7	8	9	10	11
t	3	4	**5**	6	7	8	9	10	11	12
n	2	3	**4**	5	6	7	8	8	10	11
i	1	**2**	3	4	5	6	7	8	9	10
#	**0**	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

Trace back the choices of the minimal distance (bold numbers)

- This finds the globally minimal cost path
- Full search unwieldy for large and complex matrices
- In general, searches are pruned to exclude paths that deviate far from the diagonal: Beam search

Dynamic programming

n	9	10	11	10	11	12	11	10	9	**8**
o	8	9	10	9	10	11	10	9	**8**	9
i	7	8	9	8	9	10	9	**8**	9	10
t	6	7	8	7	8	9	**8**	9	10	11
n	5	6	7	6	7	**8**	9	10	11	12
e	4	5	6	**5**	**6**	7	8	9	10	11
t	3	4	**5**	6	7	8	9	10	11	12
n	2	3	**4**	5	6	7	8	8	10	11
i	1	**2**	3	4	5	6	7	8	9	10
#	**0**	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

Trace back the choices of the minimal distance (bold numbers)

- This finds the globally minimal cost path
- Full search unwieldy for large and complex matrices
- In general, searches are pruned to exclude paths that deviate far from the diagonal: Beam search

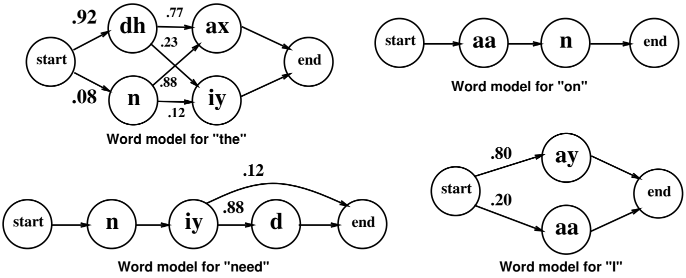
Dynamic programming

n	9	10	11	10	11	12	11	10	9	**8**
o	8	9	10	9	10	11	10	9	**8**	9
i	7	8	9	8	9	10	9	**8**	9	10
t	6	7	8	7	8	9	**8**	9	10	11
n	5	6	7	6	7	**8**	9	10	11	12
e	4	5	6	**5**	**6**	7	8	9	10	11
t	3	4	**5**	6	7	8	9	10	11	12
n	2	3	**4**	5	6	7	8	8	10	11
i	1	**2**	3	4	5	6	7	8	9	10
#	**0**	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

Trace back the choices of the minimal distance (bold numbers)

- This finds the globally minimal cost path
- Full search unwieldy for large and complex matrices
- In general, searches are pruned to exclude paths that deviate far from the diagonal: Beam search

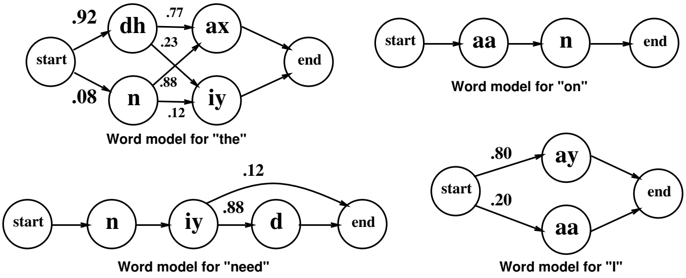
Viterbi algorithm



Simplified pronunciation networks [Jurafsky and Martin(2000)]

- Each word is modeled as a Finite State Machine
- Individual phoneme HMMs are trained from a corpus that does not contain all the words
- A pronunciation dictionary contains all word models
- Transition probabilities are "trained" from a transcribed speech corpus

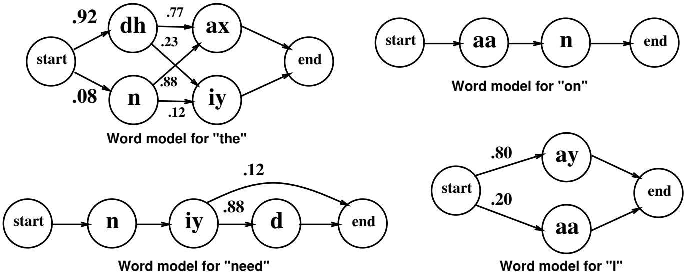
Viterbi algorithm



Simplified pronunciation networks [Jurafsky and Martin(2000)]

- Each word is modeled as a Finite State Machine
- Individual phoneme HMMs are trained from a corpus that does not contain all the words
- A pronunciation dictionary contains all word models
- Transition probabilities are "trained" from a transcribed speech corpus

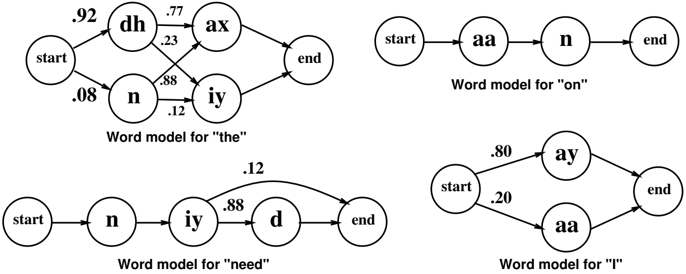
Viterbi algorithm



Simplified pronunciation networks [Jurafsky and Martin(2000)]

- Each word is modeled as a Finite State Machine
- Individual phoneme HMMs are trained from a corpus that does not contain all the words
- A pronunciation dictionary contains all word models
- Transition probabilities are "trained" from a transcribed speech corpus

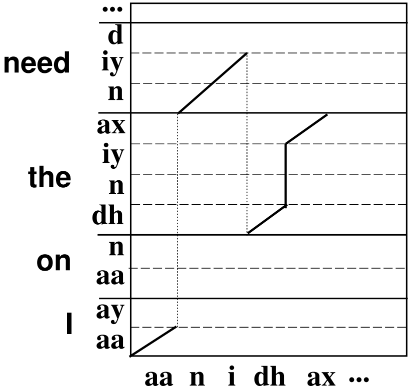
Viterbi algorithm



Simplified pronunciation networks [Jurafsky and Martin(2000)]

- Each word is modeled as a Finite State Machine
- Individual phoneme HMMs are trained from a corpus that does not contain all the words
- A pronunciation dictionary contains all word models
- Transition probabilities are "trained" from a transcribed speech corpus

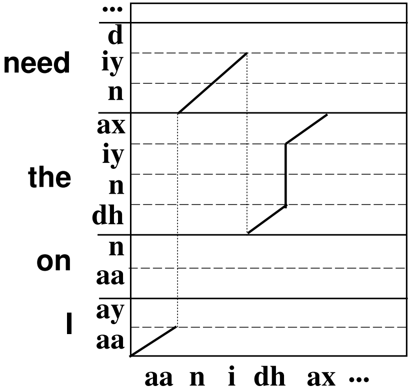
Viterbi algorithm



Viterbi algorithm result *"for I need a"* [Jurafsky and Martin(2000)]

- Whole sequence on **X** axis
- All word models on the other axis
- Switch to (any) new word after reaching the end of the current word
- Word switching cost based on the language model

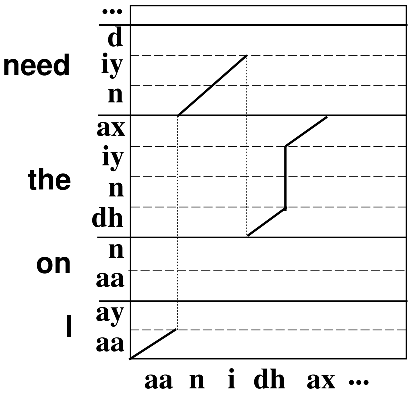
Viterbi algorithm



Viterbi algorithm result *"for I need a"* [Jurafsky and Martin(2000)]

- Whole sequence on **X** axis
- All word models on the other axis
- Switch to (any) new word after reaching the end of the current word
- Word switching cost based on the language model

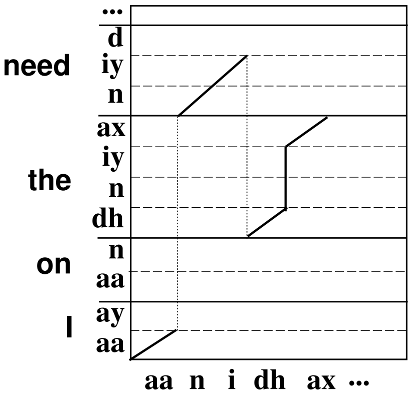
Viterbi algorithm



- Whole sequence on **X** axis
- All word models on the other axis
- Switch to (any) new word after reaching the end of the current word
- Word switching cost based on the language model

Viterbi algorithm result *"for I need a"* [Jurafsky and Martin(2000)]

Viterbi algorithm



- Whole sequence on **X** axis
- All word models on the other axis
- Switch to (any) new word after reaching the end of the current word
- Word switching cost based on the language model

Viterbi algorithm result *"for I need a"* [Jurafsky and Martin(2000)]

Viterbi algorithm

I need	0.0016	need need	0.000047	# Need	0.000018
I the	0.00018	need the	0.012	# The	0.016
I on	0.000047	need on	0.000047	# On	0.00077
I I	0.039	need I	0.000016	# I	0.079
the need	0.00051	on need	0.000055		
the the	0.0099	on the	0.094		
the on	0.00022	on on	0.0031		
the I	0.00051	on I	0.00085		

Bigram probabilities [Jurafsky and Martin(2000)]

- Word switching in Viterbi searches uses probabilities
- Switch to a new word with bigram probability cost
- Does not work with trigram probabilities

Viterbi algorithm

I need	0.0016	need need	0.000047	# Need	0.000018
I the	0.00018	need the	0.012	# The	0.016
I on	0.000047	need on	0.000047	# On	0.00077
I I	0.039	need I	0.000016	# I	0.079
the need	0.00051	on need	0.000055		
the the	0.0099	on the	0.094		
the on	0.00022	on on	0.0031		
the I	0.00051	on I	0.00085		

Bigram probabilities [Jurafsky and Martin(2000)]

- Word switching in Viterbi searches uses probabilities
- Switch to a new word with bigram probability cost
- Does not work with trigram probabilities

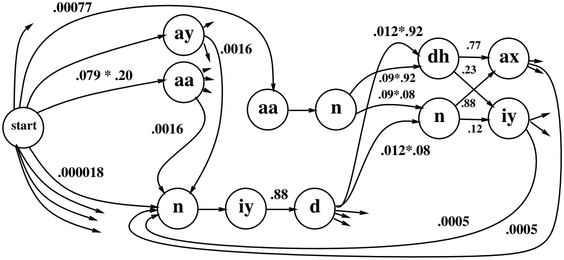
Viterbi algorithm

I need	0.0016	need need	0.000047	# Need	0.000018
I the	0.00018	need the	0.012	# The	0.016
I on	0.000047	need on	0.000047	# On	0.00077
I I	0.039	need I	0.000016	# I	0.079
the need	0.00051	on need	0.000055		
the the	0.0099	on the	0.094		
the on	0.00022	on on	0.0031		
the I	0.00051	on I	0.00085		

Bigram probabilities [Jurafsky and Martin(2000)]

- Word switching in Viterbi searches uses probabilities
- Switch to a new word with bigram probability cost
- Does not work with trigram probabilities

Viterbi algorithm

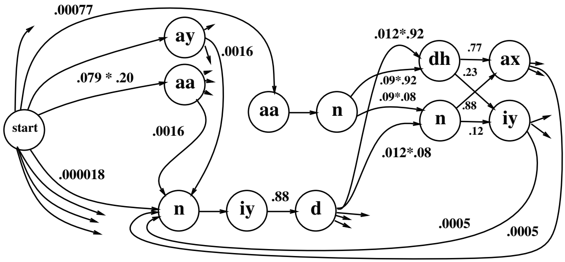


Single pronunciation automaton for *I*, *need*, *on*, and *the*

[Jurafsky and Martin(2000)]

- Bigram probabilities connect the word models
- Merge **start** and **end** states of connected words
- Need for *pruning* is apparent

Viterbi algorithm

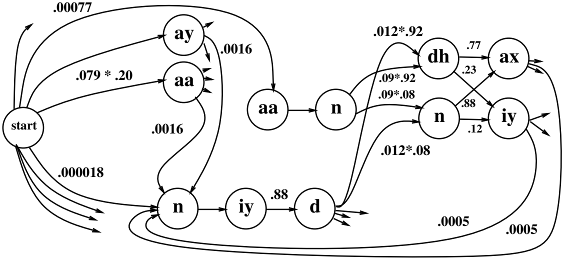


Single pronunciation automaton for *I*, *need*, *on*, and *the*

[Jurafsky and Martin(2000)]

- Bigram probabilities connect the word models
- Merge **start** and **end** states of connected words
- Need for *pruning* is apparent

Viterbi algorithm



Single pronunciation automaton for *I*, *need*, *on*, and *the*

[Jurafsky and Martin(2000)]

- Bigram probabilities connect the word models
- Merge **start** and **end** states of connected words
- Need for *pruning* is apparent

Viterbi algorithm

function VITERBI(*observations* of len *T*,*state-graph*) **returns** *best-path*

num-states ← NUM-OF-STATES(*state-graph*)
Create a path probability matrix *viterbi[num-states+2,T+2]*
viterbi[0,0] ← 1.0
for each time step t **from** 0 **to** T **do**
  for each state s **from** 0 **to** *num-states* **do**
    for each transition s' from s specified by *state-graph*
      new-score ← *viterbi*[s, t] * $a[s,s']$ * $b_{s'}(o_t)$
      if ((*viterbi*[s',$t+1$] = 0) || (*new-score* > *viterbi*[s', $t+1$]))
        then
          viterbi[s', $t+1$] ← *new-score*
          back-pointer[s', $t+1$] ← s
Backtrace from highest probability state in the final column of *viterbi[]* and return path

Extended version of the edit distance [Jurafsky and Martin(2000)]

- $a[s, s'] = P(s \rightarrow s')$
- $b_{s'}(o_t) = P(o_t|s')$

Viterbi algorithm

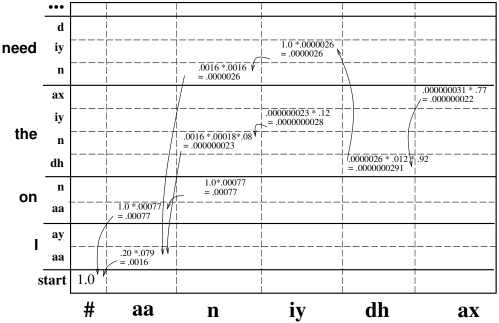
```
function VITERBI(observations of len T,state-graph) returns best-path

  num-states ← NUM-OF-STATES(state-graph)
  Create a path probability matrix viterbi[num-states+2,T+2]
  viterbi[0,0] ← 1.0
  for each time step t from 0 to T do
    for each state s from 0 to num-states do
      for each transition s' from s specified by state-graph
        new-score ← viterbi[s, t] * a[s,s'] * b_s'(o_t)
        if ((viterbi[s',t+1] = 0) || (new-score > viterbi[s', t+1]))
          then
                viterbi[s', t+1] ← new-score
                back-pointer[s', t+1] ← s
  Backtrace from highest probability state in the final column of viterbi[] and
return path
```

Extended version of the edit distance [Jurafsky and Martin(2000)]

- $a[s, s'] = P(s \rightarrow s')$
- $b_{s'}(o_t) = P(o_t|s')$

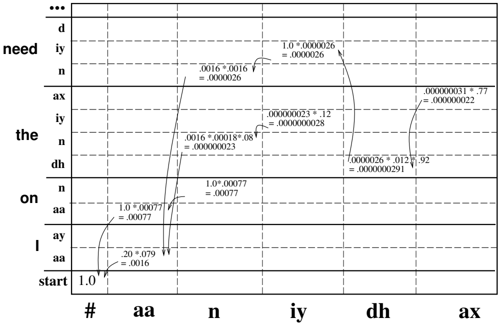
Viterbi algorithm



Individual state columns in Viterbi algorithm [Jurafsky and Martin(2000)]

- The actual entries for the Automaton
- Note the problems for a 20,000 word dictionary

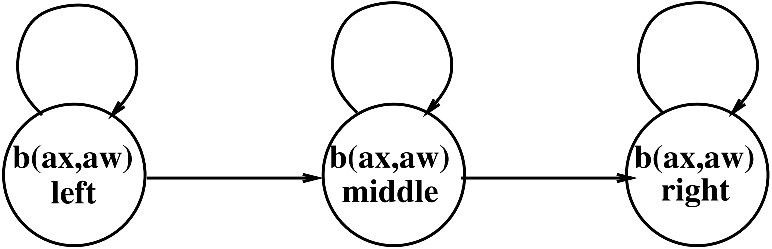
Viterbi algorithm



Individual state columns in Viterbi algorithm [Jurafsky and Martin(2000)]

- The actual entries for the Automaton
- Note the problems for a 20,000 word dictionary

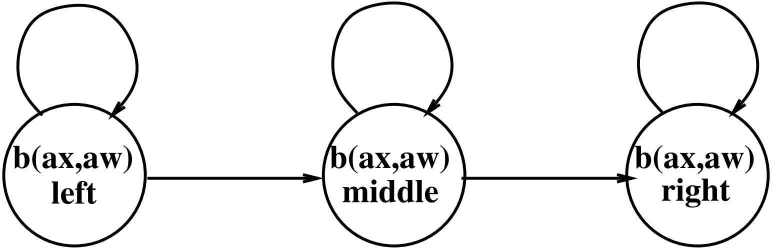
Viterbi algorithm: Subphones revisited [Jurafsky and Martin(2000)]



Use structured, context sensitive phone units

- Single phone units perform bad due to coarticulation
- *Begin* differs from *End* (eg, /d/)
- 60 context dependent triphones ⇒ $60^3 = 216000$ models
- Cluster contexts,eg, on manner and place of articulation

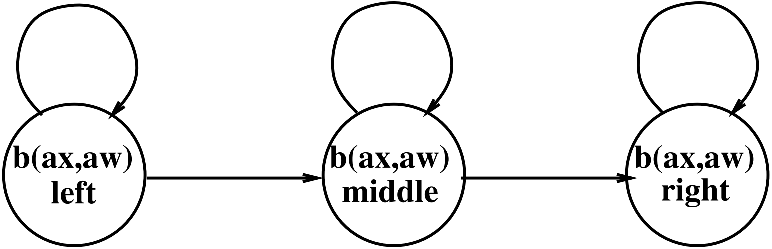
Viterbi algorithm: Subphones revisited [Jurafsky and Martin(2000)]



Use structured, context sensitive phone units

- Single phone units perform bad due to coarticulation
- *Begin* differs from *End* (eg, /d/)
- 60 context dependent triphones ⇒ $60^3 = 216000$ models
- Cluster contexts,eg, on manner and place of articulation

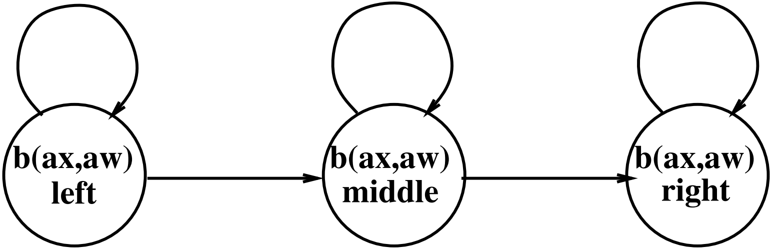
Viterbi algorithm: Subphones revisited [Jurafsky and Martin(2000)]



Use structured, context sensitive phone units

- Single phone units perform bad due to coarticulation
- *Begin* differs from *End* (eg, /d/)
- 60 context dependent triphones $\Rightarrow 60^3 = 216000$ models
- Cluster contexts,eg, on manner and place of articulation

Viterbi algorithm: Subphones revisited [Jurafsky and Martin(2000)]



Use structured, context sensitive phone units

- Single phone units perform bad due to coarticulation
- *Begin* differs from *End* (eg, /d/)
- 60 context dependent triphones ⇒ $60^3 = 216000$ models
- Cluster contexts,eg, on manner and place of articulation

Other approaches to decoding: Introduction

The standard HMM model has limitations

- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

Other approaches to decoding: Introduction

The standard HMM model has limitations

- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

Other approaches to decoding: Introduction

The standard HMM model has limitations

- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

Other approaches to decoding: Introduction

The standard HMM model has limitations

- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

Other approaches to decoding: Introduction

The standard HMM model has limitations

- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

Other approaches to decoding: Introduction

The standard HMM model has limitations

- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

Other approaches to decoding: Introduction

The standard HMM model has limitations

- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

Other approaches to decoding: Introduction

The standard HMM model has limitations

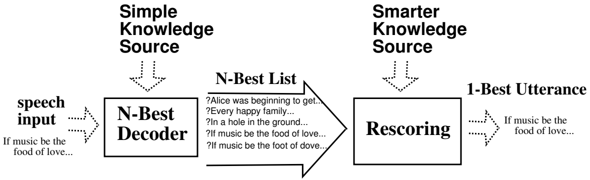
- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

Other approaches to decoding: Introduction

The standard HMM model has limitations

- Viterbi decoder penalizes multiple pronunciations
- Viterbi decoder does not work for anything more complex than bigram
- It is not possible to include other linguistic knowledge
  - Phoneme duration (HMM have a Poison distribution)
  - Intonation
  - Semantics
  - Speaker identification
  - Expressive speech tags
  - Task related knowledge

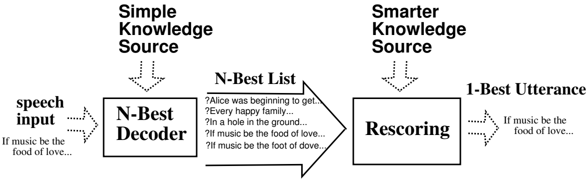
Other approaches to decoding



Two stage N-best decoding [Jurafsky and Martin(2000)]

- Keep N-best utterance list or word lattice
- Rescore the probabilities with the extra knowledge
  - A trigram or higher grammar
  - Phoneme duration probability Chapt 7 [Wang(1997)]
  - Parallel Intonation and Accent detector (HMM) example without N-best [Taylor et al.(1998)Taylor, King, Isard, and Wright]
  - Include semantic or task related knowledge
  - Multiple speakers and expressive speech tags
- Look up best path through rescored word lattice

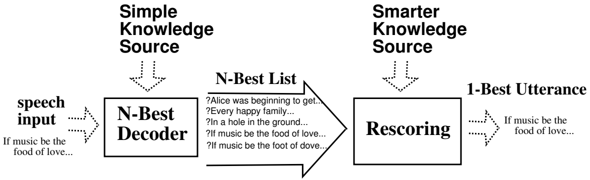
Other approaches to decoding



Two stage N-best decoding [Jurafsky and Martin(2000)]

- Keep N-best utterance list or word lattice
- Rescore the probabilities with the extra knowledge
  - A trigram or higher grammar
  - Phoneme duration probability Chapt 7 [Wang(1997)]
  - Parallel Intonation and Accent detector (HMM) example without N-best [Taylor et al.(1998)Taylor, King, Isard, and Wright]
  - Include semantic or task related knowledge
  - Multiple speakers and expressive speech tags
- Look up best path through rescored word lattice

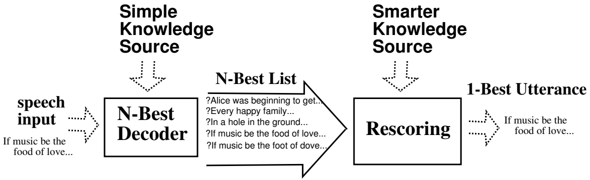
Other approaches to decoding



Two stage N-best decoding [Jurafsky and Martin(2000)]

- Keep N-best utterance list or word lattice
- Rescore the probabilities with the extra knowledge
  - A trigram or higher grammar
  - Phoneme duration probability Chapt 7 [Wang(1997)]
  - Parallel Intonation and Accent detector (HMM) example without N-best [Taylor et al.(1998)Taylor, King, Isard, and Wright]
  - Include semantic or task related knowledge
  - Multiple speakers and expressive speech tags
- Look up best path through rescored word lattice

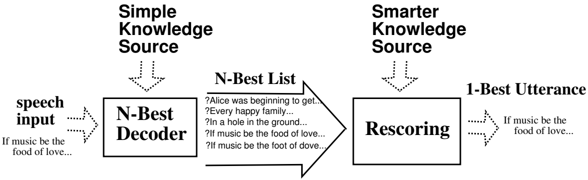
Other approaches to decoding



Two stage N-best decoding [Jurafsky and Martin(2000)]

- Keep N-best utterance list or word lattice
- Rescore the probabilities with the extra knowledge
  - A trigram or higher grammar
  - Phoneme duration probability Chapt 7 [Wang(1997)]
  - Parallel Intonation and Accent detector (HMM) example without N-best [Taylor et al.(1998)Taylor, King, Isard, and Wright]
  - Include semantic or task related knowledge
  - Multiple speakers and expressive speech tags
- Look up best path through rescored word lattice

Other approaches to decoding



Two stage N-best decoding [Jurafsky and Martin(2000)]

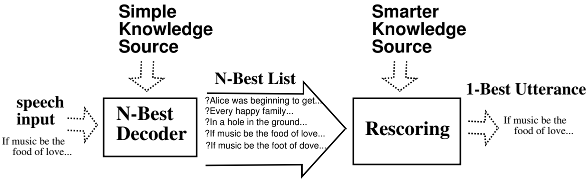
- Keep N-best utterance list or word lattice
- Rescore the probabilities with the extra knowledge
  - A trigram or higher grammar
  - Phoneme duration probability Chapt 7 [Wang(1997)]
  - Parallel Intonation and Accent detector (HMM) example without N-best [Taylor et al.(1998)Taylor, King, Isard, and Wright]
  - Include semantic or task related knowledge
  - Multiple speakers and expressive speech tags
- Look up best path through rescored word lattice

Other approaches to decoding

Two stage N-best decoding [Jurafsky and Martin(2000)]

- Keep N-best utterance list or word lattice
- Rescore the probabilities with the extra knowledge
  - A trigram or higher grammar
  - Phoneme duration probability Chapt 7 [Wang(1997)]
  - Parallel Intonation and Accent detector (HMM) example without N-best [Taylor et al.(1998)Taylor, King, Isard, and Wright]
  - Include semantic or task related knowledge
  - Multiple speakers and expressive speech tags
- Look up best path through rescored word lattice

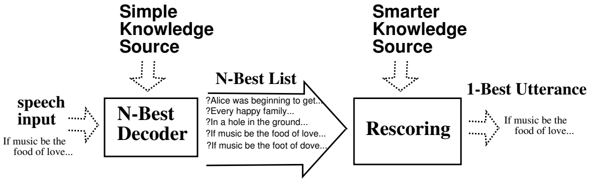
Other approaches to decoding



Two stage N-best decoding [Jurafsky and Martin(2000)]

- Keep N-best utterance list or word lattice
- Rescore the probabilities with the extra knowledge
  - A trigram or higher grammar
  - Phoneme duration probability Chapt 7 [Wang(1997)]
  - Parallel Intonation and Accent detector (HMM) example without N-best [Taylor et al.(1998)Taylor, King, Isard, and Wright]
  - Include semantic or task related knowledge
  - Multiple speakers and expressive speech tags
- Look up best path through rescored word lattice

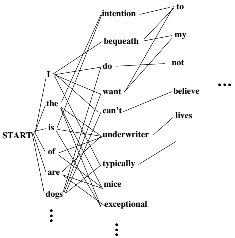
Other approaches to decoding



Two stage N-best decoding [Jurafsky and Martin(2000)]

- Keep N-best utterance list or word lattice
- Rescore the probabilities with the extra knowledge
    - A trigram or higher grammar
    - Phoneme duration probability Chapt 7 [Wang(1997)]
    - Parallel Intonation and Accent detector (HMM) example without N-best [Taylor et al.(1998)Taylor, King, Isard, and Wright]
    - Include semantic or task related knowledge
    - Multiple speakers and expressive speech tags
- Look up best path through rescored word lattice

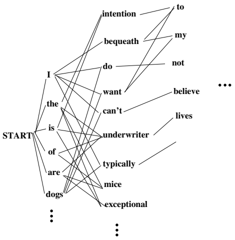
Other approaches to decoding: A^*



Stack, or A^*, decoding [Jurafsky and Martin(2000)]

- Viterbi uses best path upto position t to get to $t + 1$
- A^* uses complete forward algorithm (exact likelihoods)
- A^* searches potential utterances best-first

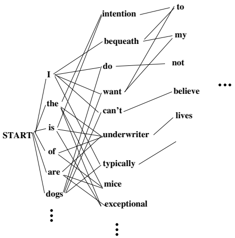
Other approaches to decoding: A^*



Stack, or A^*, decoding [Jurafsky and Martin(2000)]

- Viterbi uses best path upto position t to get to $t+1$
- A^* uses complete forward algorithm (exact likelihoods)
- A^* searches potential utterances best-first

Other approaches to decoding: A^*



Stack, or A^*, decoding [Jurafsky and Martin(2000)]

- Viterbi uses best path upto position t to get to $t + 1$
- A^* uses complete forward algorithm (exact likelihoods)
- A^* searches potential utterances best-first

Other approaches to decoding: A^*

function STACK-DECODING() **returns** *min-distance*

Initialize the priority queue with a null sentence.
Pop the best (highest score) sentence s off the queue.
If (s is marked end-of-sentence (EOS)) output s and terminate.
Get list of candidate next words by doing fast matches.
For each candidate next word w:
  Create a new candidate sentence $s+w$.
  Use forward algorithm to compute acoustic likelihood L of $s+w$
  Compute language model probability P of extended sentence $s+w$
  Compute "score" for $s+w$ (a function of L, P, and ???)
  if (end-of-sentence) set EOS flag for $s+w$.
  Insert $s+w$ into the queue together with its score and EOS flag

Stack decoding [Jurafsky and Martin(2000)]

- At each point, the A^* looks for the most likely next word
- Acoustic likelihood is part of the criterium
- Use the forward probability of preceding words

Other approaches to decoding: A^*

function STACK-DECODING() **returns** *min-distance*

Initialize the priority queue with a null sentence.
Pop the best (highest score) sentence s off the queue.
If (s is marked end-of-sentence (EOS)) output s and terminate.
Get list of candidate next words by doing fast matches.
For each candidate next word w:
  Create a new candidate sentence $s+w$.
  Use forward algorithm to compute acoustic likelihood L of $s+w$
  Compute language model probability P of extended sentence $s+w$
  Compute "score" for $s+w$ (a function of L, P, and ???)
  if (end-of-sentence) set EOS flag for $s+w$.
  Insert $s+w$ into the queue together with its score and EOS flag

Stack decoding [Jurafsky and Martin(2000)]

- At each point, the A^* looks for the most likely next word
- Acoustic likelihood is part of the criterium
- Use the forward probability of preceding words

Other approaches to decoding: A^*

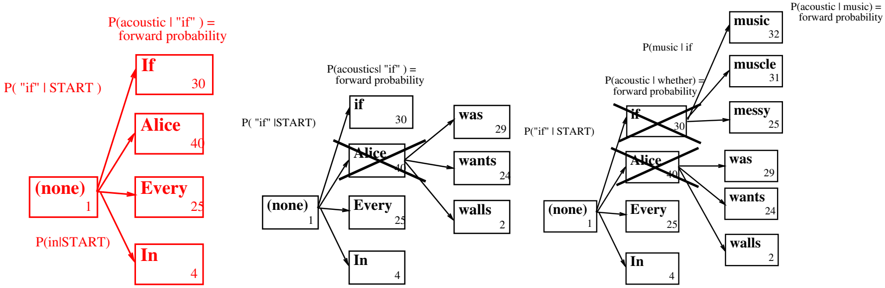
function STACK-DECODING() **returns** *min-distance*

Initialize the priority queue with a null sentence.
Pop the best (highest score) sentence s off the queue.
If (s is marked end-of-sentence (EOS)) output s and terminate.
Get list of candidate next words by doing fast matches.
For each candidate next word w:
    Create a new candidate sentence $s+w$.
    Use forward algorithm to compute acoustic likelihood L of $s+w$
    Compute language model probability P of extended sentence $s+w$
    Compute "score" for $s+w$ (a function of L, P, and ???)
    if (end-of-sentence) set EOS flag for $s+w$.
    Insert $s+w$ into the queue together with its score and EOS flag

Stack decoding [Jurafsky and Martin(2000)]

- At each point, the A^* looks for the most likely next word
- Acoustic likelihood is part of the criterium
- Use the forward probability of preceding words

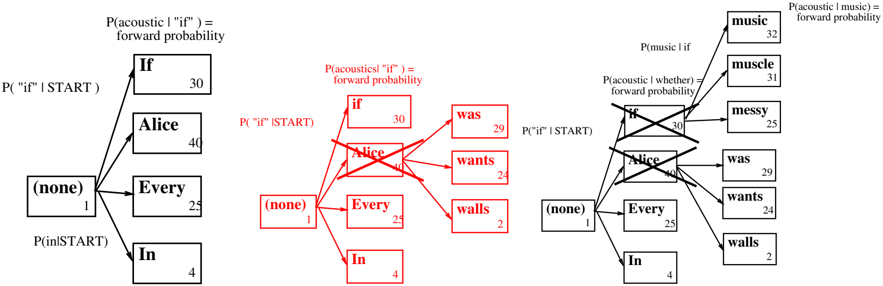
Other approaches to decoding: A^*



If music be the food of love [Jurafsky and Martin(2000)]

- *"Start Alice"* has highest score: 40
- *"Start if"* has highest score: 30
- *"Start if music"* has highest score: 32

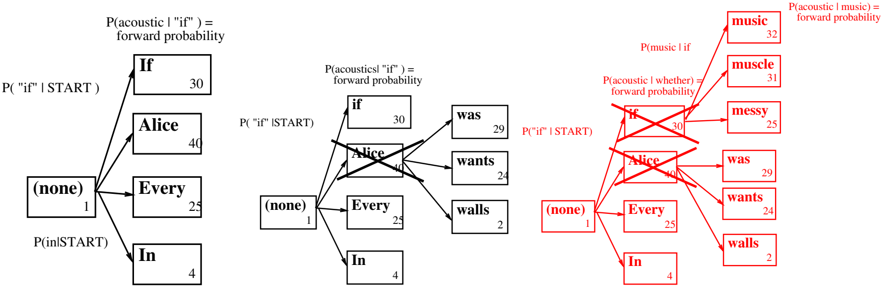
Other approaches to decoding: A^*



If music be the food of love [Jurafsky and Martin(2000)]

- *"Start Alice"* has highest score: 40
- *"Start if"* has highest score: 30
- *"Start if music"* has highest score: 32

Other approaches to decoding: A^*



If music be the food of love [Jurafsky and Martin(2000)]

- *"Start Alice"* has highest score: 40
- *"Start if"* has highest score: 30
- *"Start if music"* has highest score: 32

Other approaches to decoding: A^*

Remarks

- Use fast match heuristics for selecting next words
- Longer utterances have lower probabilities, score should correct for this
- A^* evaluation function: $f^*(p) = g(p) + h^*(p)$
- $g(partial\ path) = P(O|Words) \cdot P(Words)$, i.e. the likelihood until now
- $h^*(p)$ something that correlates with number of words in the rest of the utterance
- Defining a good $h^*(p)$ is an interesting (unsolved) problem

Other approaches to decoding: A^*

Remarks

- Use fast match heuristics for selecting next words
- Longer utterances have lower probabilities, score should correct for this
- A^* evaluation function: $f^*(p) = g(p) + h^*(p)$
- $g(partial\ path) = P(O|Words) \cdot P(Words)$, i.e. the likelihood until now
- $h^*(p)$ something that correlates with number of words in the rest of the utterance
- Defining a good $h^*(p)$ is an interesting (unsolved) problem

Other approaches to decoding: A^*

Remarks

- Use fast match heuristics for selecting next words
- Longer utterances have lower probabilities, score should correct for this
- A^* evaluation function: $f^*(p) = g(p) + h^*(p)$
- $g(partial\ path) = P(O|Words) \cdot P(Words)$, i.e. the likelihood until now
- $h^*(p)$ something that correlates with number of words in the rest of the utterance
- Defining a good $h^*(p)$ is an interesting (unsolved) problem

Other approaches to decoding: A^*

Remarks

- Use fast match heuristics for selecting next words
- Longer utterances have lower probabilities, score should correct for this
- A^* evaluation function: $f^*(p) = g(p) + h^*(p)$
- $g(partial\ path) = P(O|Words) \cdot P(Words)$, i.e. the likelihood until now
- $h^*(p)$ something that correlates with number of words in the rest of the utterance
- Defining a good $h^*(p)$ is an interesting (unsolved) problem

Other approaches to decoding: A^*

Remarks

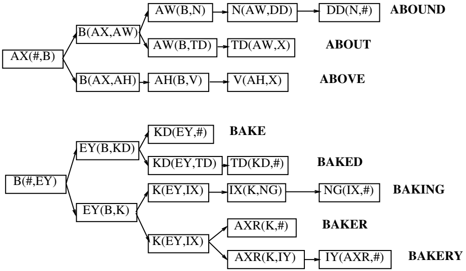
- Use fast match heuristics for selecting next words
- Longer utterances have lower probabilities, score should correct for this
- A^* evaluation function: $f^*(p) = g(p) + h^*(p)$
- $g(partial\ path) = P(O|Words) \cdot P(Words)$, i.e. the likelihood until now
- $h^*(p)$ something that correlates with number of words in the rest of the utterance
- Defining a good $h^*(p)$ is an interesting (unsolved) problem

Other approaches to decoding: A^*

Remarks

- Use fast match heuristics for selecting next words
- Longer utterances have lower probabilities, score should correct for this
- A^* evaluation function: $f^*(p) = g(p) + h^*(p)$
- $g(partial\ path) = P(O|Words) \cdot P(Words)$, i.e. the likelihood until now
- $h^*(p)$ something that correlates with number of words in the rest of the utterance
- Defining a good $h^*(p)$ is an interesting (unsolved) problem

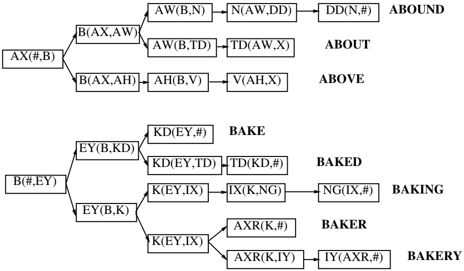
Other approaches to decoding: A^* fast match



A tree structured lexicon from SPHINX [Gouvêa()][Jurafsky and Martin(2000)]

- Need to get forward probabilities of potential continuations *fast*
- Tree lexicon shares forward probabilities between words
- Allows early pruning of search trees

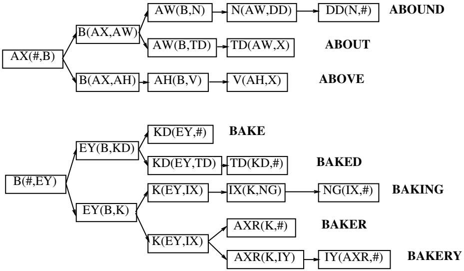
Other approaches to decoding: A^* fast match



A tree structured lexicon from SPHINX [Gouvêa()][Jurafsky and Martin(2000)]

- Need to get forward probabilities of potential continuations *fast*
- Tree lexicon shares forward probabilities between words
- Allows early pruning of search trees

Other approaches to decoding: A^* fast match



A tree structured lexicon from SPHINX [Gouvêa()][Jurafsky and Martin(2000)]

- Need to get forward probabilities of potential continuations *fast*
- Tree lexicon shares forward probabilities between words
- Allows early pruning of search trees

Training acoustic models: Introduction

Determine $P(Observation|Words)$, i.e. the transition probability between phone states a_{ij} and the acoustic likelihood of the speech vectors $b_j(o_k)$

- Large, "transcribed" speech corpus (on text level)
- Coverage of speakers and language types
- Recorded under the same conditions as intended use, eg, over the phone or in a driving car
- Use the same microphone etc.
- Using a simulated task (Wizard of Oz or Green curtain) to elicit the same kind of speech

Training acoustic models: Introduction

Determine $P(Observation|Words)$, i.e. the transition probability between phone states a_{ij} and the acoustic likelihood of the speech vectors $b_j(o_k)$

- Large, "transcribed" speech corpus (on text level)
- Coverage of speakers and language types
- Recorded under the same conditions as intended use, eg, over the phone or in a driving car
- Use the same microphone etc.
- Using a simulated task (Wizard of Oz or Green curtain) to elicit the same kind of speech

Training acoustic models: Introduction

Determine $P(Observation|Words)$, i.e. the transition probability between phone states a_{ij} and the acoustic likelihood of the speech vectors $b_j(o_k)$

- Large, “transcribed” speech corpus (on text level)
- Coverage of speakers and language types
- Recorded under the same conditions as intended use, eg, over the phone or in a driving car
- Use the same microphone etc.
- Using a simulated task (Wizard of Oz or Green curtain) to elicit the same kind of speech

Training acoustic models: Introduction

Determine $P(Observation|Words)$, i.e. the transition probability between phone states a_{ij} and the acoustic likelihood of the speech vectors $b_j(o_k)$

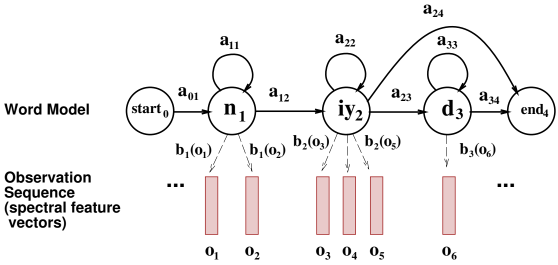
- Large, "transcribed" speech corpus (on text level)
- Coverage of speakers and language types
- Recorded under the same conditions as intended use, eg, over the phone or in a driving car
- Use the same microphone etc.
- Using a simulated task (Wizard of Oz or Green curtain) to elicit the same kind of speech

Training acoustic models: Introduction

Determine $P(Observation|Words)$, i.e. the transition probability between phone states a_{ij} and the acoustic likelihood of the speech vectors $b_j(o_k)$

- Large, "transcribed" speech corpus (on text level)
- Coverage of speakers and language types
- Recorded under the same conditions as intended use, eg, over the phone or in a driving car
- Use the same microphone etc.
- Using a simulated task (Wizard of Oz or Green curtain) to elicit the same kind of speech

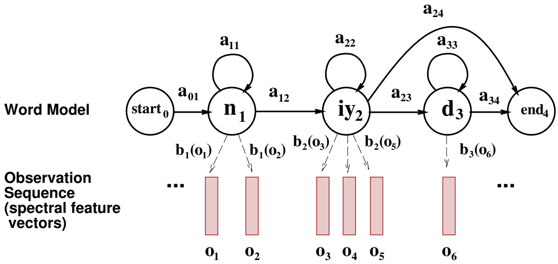
Training acoustic models



If all states were known [Jurafsky and Martin(2000)]

- $a_{ij} = \frac{\#S_{ij}}{\#S_{i*}}$ (count transitions and states)
- $b_i(O_k) = \frac{\#(O_k \& S_i)}{\#S_i}$ (for discrete O_k)

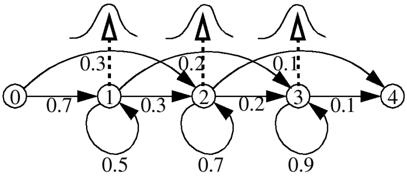
Training acoustic models



If all states were known [Jurafsky and Martin(2000)]

- $a_{ij} = \frac{\#S_{ij}}{\#S_{i*}}$ (count transitions and states)
- $b_i(O_k) = \frac{\#(O_k \& S_i)}{\#S_i}$ (for discrete O_k)

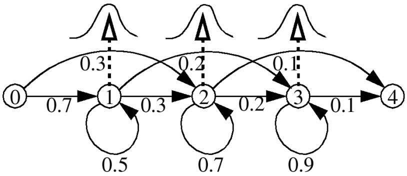
Training acoustic models



If observations are continuous vectors [SPH()]

- $b_i(O_t) \Rightarrow N\{\hat{\mu}_i, \hat{\Sigma}_i\}$
- $\hat{\mu}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} O_t$
- $\hat{\Sigma}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} [(O_t - \hat{\mu}_i)'(O_t - \hat{\mu}_i)]$

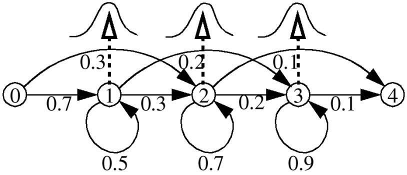
Training acoustic models



If observations are continuous vectors [SPH()]

- $b_i(O_t) \Rightarrow N\{\hat{\mu}_i, \hat{\Sigma}_i\}$
- $\hat{\mu}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} O_t$
- $\hat{\Sigma}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} [(O_t - \hat{\mu}_i)'(O_t - \hat{\mu}_i)]$

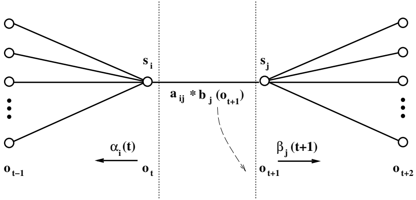
Training acoustic models



If observations are continuous vectors [SPH()]

- $b_i(O_t) \Rightarrow N\{\hat{\mu}_i, \hat{\Sigma}_i\}$
- $\hat{\mu}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} O_t$
- $\hat{\Sigma}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} [(O_t - \hat{\mu}_i)'(O_t - \hat{\mu}_i)]$

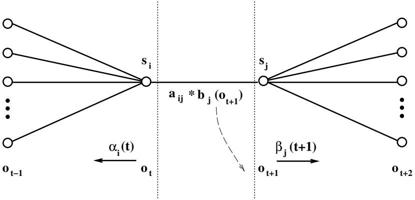
Training acoustic models



States have to be estimated. Use an iterative procedure App D [Jurafsky and Martin(2000)]

- Run the recognizer on the corpus with the known words
- Calculate $\hat{a}_{ij} = \frac{expected\ \#S_i \rightarrow S_j}{expected\ \#S_i \rightarrow S_*}$
- Calculate $\hat{b}_j(v_k) = \frac{expected\ \#S_j\ observing\ v_k}{expected\ \#S_j}$
- Update all values and start again

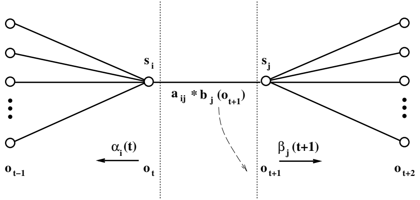
Training acoustic models



States have to be estimated. Use an iterative procedure App D [Jurafsky and Martin(2000)]

- Run the recognizer on the corpus with the known words
- Calculate $\hat{a}_{ij} = \frac{expected\ \#S_i \rightarrow S_j}{expected\ \#S_i \rightarrow S_*}$
- Calculate $\hat{b}_j(v_k) = \frac{expected\ \#S_j\ observing\ v_k}{expected\ \#S_j}$
- Update all values and start again

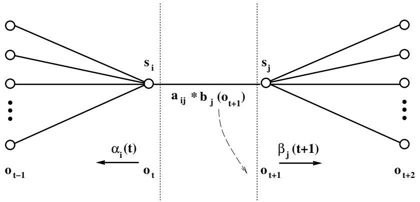
Training acoustic models



States have to be estimated. Use an iterative procedure App D [Jurafsky and Martin(2000)]

- Run the recognizer on the corpus with the known words
- Calculate $\hat{a}_{ij} = \frac{expected\ \#S_i \rightarrow S_j}{expected\ \#S_i \rightarrow S_*}$
- Calculate $\hat{b}_j(v_k) = \frac{expected\ \#S_j\ observing\ v_k}{expected\ \#S_j}$
- Update all values and start again

Training acoustic models



States have to be estimated. Use an iterative procedure App D [Jurafsky and Martin(2000)]

- Run the recognizer on the corpus with the known words
- Calculate $\hat{a}_{ij} = \frac{expected\ \#S_i \rightarrow S_j}{expected\ \#S_i \rightarrow S_*}$
- Calculate $\hat{b}_j(v_k) = \frac{expected\ \#S_j\ observing\ v_k}{expected\ \#S_j}$
- Update all values and start again

FLOSS resources

Free and Open Source ASR systems

- SPHINX (CMU) [Gouvêa()] [Singh(2005)]
- CMU Statistical Language Modeling Toolkit [Rosenfeld()]
- CMU Pronouncing Dictionary [Lenzo()]
- Internet-Accessible Speech Recognition Technology project (ISIP, Mississippi State University) [ISIP(2004)]
- Open Mind Speech [Valin()]

FLOSS resources

Free and Open Source ASR systems

- SPHINX (CMU) [Gouvêa()] [Singh(2005)]
- CMU Statistical Language Modeling Toolkit [Rosenfeld()]
- CMU Pronouncing Dictionary [Lenzo()]
- Internet-Accessible Speech Recognition Technology project (ISIP, Mississippi State University) [ISIP(2004)]
- Open Mind Speech [Valin()]

FLOSS resources

Free and Open Source ASR systems

- SPHINX (CMU) [Gouvêa()] [Singh(2005)]
- CMU Statistical Language Modeling Toolkit [Rosenfeld()]
- CMU Pronouncing Dictionary [Lenzo()]
- Internet-Accessible Speech Recognition Technology project (ISIP, Mississippi State University) [ISIP(2004)]
- Open Mind Speech [Valin()]

FLOSS resources

Free and Open Source ASR systems

- SPHINX (CMU) [Gouvêa()] [Singh(2005)]
- CMU Statistical Language Modeling Toolkit [Rosenfeld()]
- CMU Pronouncing Dictionary [Lenzo()]
- Internet-Accessible Speech Recognition Technology project (ISIP, Mississippi State University) [ISIP(2004)]
- Open Mind Speech [Valin()]

FLOSS resources

Free and Open Source ASR systems

- SPHINX (CMU) [Gouvêa()] [Singh(2005)]
- CMU Statistical Language Modeling Toolkit [Rosenfeld()]
- CMU Pronouncing Dictionary [Lenzo()]
- Internet-Accessible Speech Recognition Technology project (ISIP, Mississippi State University) [ISIP(2004)]
- Open Mind Speech [Valin()]

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use --help to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the --count option. Try --count -1 on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt "<sentence>"
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Assignment: Week 8 Statistical Language Models

Construct your own language model

- Download texts from the internet, eg, [Project Gutenberg(2005)]
- Use a single author or a single genre
- Use `--help` to see instructions of the programs
- Construct unigram and bigram word tables with *Ngramcount.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/Ngramcount.pl`
- perl Ngramcount.pl 1 <filename1> <filename2> ... > unigramtable.txt
- perl Ngramcount.pl 2 <filename1> <filename2> ... > bigramtable.txt
- Inspect the table files. What are the most frequent words and bigrams?
- Calculate the probabilities of sentences with *ngramprobability.pl*
  `http://www.fon.hum.uva.nl/rob/Courses/Taaltechnologien/ngramprobability.pl`
- perl ngramprobability.pl –count 5 –verbose bigramtable.txt ”<sentence>”
- Enter some sentences and inspect the resulting probabilities
- Experiment with the `--count` option. Try `--count -1` on a sentence that contains unknown word combinations

Further Reading I

P. Boersma.
Praat, a system for doing phonetics by computer.
Glot International, 5:341–345, 2001.
URL http://www.Praat.org/.

P. Boersma and D. Weenink.
Praat 4.2: doing phonetics by computer.
Computer program: http://www.Praat.org/, 2004.
URL http://www.Praat.org/.

CSLU.
CSLU Toolkit.
Web.
URL http://cslu.cse.ogi.edu/toolkit/index.html.

FSF.
GNU General Public License.
Web, June 1991.
URL http://www.gnu.org/licenses/gpl.html.

Joshua T. Goodman.
A bit of progress in language modeling.
Computer Speech and Language, 15:403–434, 2001.
URL http://arxiv.org/abs/cs.CL/0108005.
URL is extended preprint.

Further Reading II

E. Gouvêa.
The CMU Sphinx Group Open Source Speech Recognition Engines.
Web.
URL http://cmusphinx.sourceforge.net/html/cmusphinx.php.

ISIP.
The Mississippi State ISIP public domain speech recognizer.
Web, August 2004.
URL http://www.cavs.msstate.edu/hse/ies/projects/speech/index.html.

Daniel Jurafsky and James H. Martin.
Speech and Language Processing.
Prentice-Hall, 2000.
ISBN 0-13-095069-6.
URL http://www.cs.colorado.edu/~martin/slp.html.
Updates at http://www.cs.colorado.edu/

Kevin Lenzo.
The CMU Pronouncing Dictionary.
Web.
URL http://www.speech.cs.cmu.edu/SLM_info.html.

Project Gutenberg.
Project gutenberg free ebook library.
Web, 2005.
URL http://www.gutenberg.org/.

Further Reading III

Roni Rosenfeld.
The CMU Statistical Language Modeling (SLM) Toolkit.
Web.
URL http://www.speech.cs.cmu.edu/SLM_info.html.

Rita Singh.
Robust group's open source tutorial learning to use the cmu sphinx automatic speech recognition system.
Web, 2005.
URL http://www.cs.cmu.edu/~robust/Tutorial/opensource.html.

Manual for the Sphinx-III recognition system.
SPHINX-CMU.
URL http://fife.speech.cs.cmu.edu/sphinxman/.

Paul A. Taylor, S. King, S. D. Isard, and H. Wright.
Intonation and dialogue context as constraints for speech recognition.
Language and Speech, 41:493–512, 1998.
URL http://www.cstr.ed.ac.uk/downloads/publications/1998/Taylor_1998_b.pdf.

Jean-Marc Valin.
Open mind speech.
Web.
URL http://freespeech.sourceforge.net/.

Xue Wang.
incorporating knowledge on segmental duration in hmm-based continuous speech recognition.
PhD thesis, LOT Netherlands Graduate School of Linguistics, 04 1997.
URL http://www.fon.hum.uva.nl/wang/ThesisWangXue/TOCINDEX.html.

Appendix A

Copyright License

The GNU General Public License I

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

The GNU General Public License II

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.
Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.
The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".
   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

The GNU General Public License III

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
   2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
   3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

The GNU General Public License IV

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.
In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
    1. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
    2. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
    3. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The GNU General Public License V

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

The GNU General Public License VI

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.
   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.
   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

The GNU General Public License VII

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

The GNU General Public License VIII

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> *one line to give the program's name and a brief idea of what it does.*
> *Copyright (C) yyyy name of author*
> *This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.*
>
> *This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.*
>
> *You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.*

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> *Gnomovision version 69, Copyright (C) yyyy name of author*
> *Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.*
> *This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.*

The GNU General Public License IX

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.
You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> *Yoyodyne, Inc., hereby disclaims all copyright interest in the program*
> *'Gnomovision' (which makes passes at compilers) written by James Hacker.*
> *signature of Ty Coon, 1 April 1989*
> *Ty Coon, President of Vice*

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.