Disclaimer

- The material provided in this document is not my original work and is a summary of some one else's work(s).
- A simple Google search of the title of the document will direct you to the original source of the material.
- I do not guarantee the accuracy, completeness, timeliness, validity, non-omission, merchantability or fitness of the contents of this document for any particular purpose.
- Downloaded from najeebkhan.github.io

Chapter # 12: Networks, Dictionaries and Language Models

HMM TOOL KIT HTK

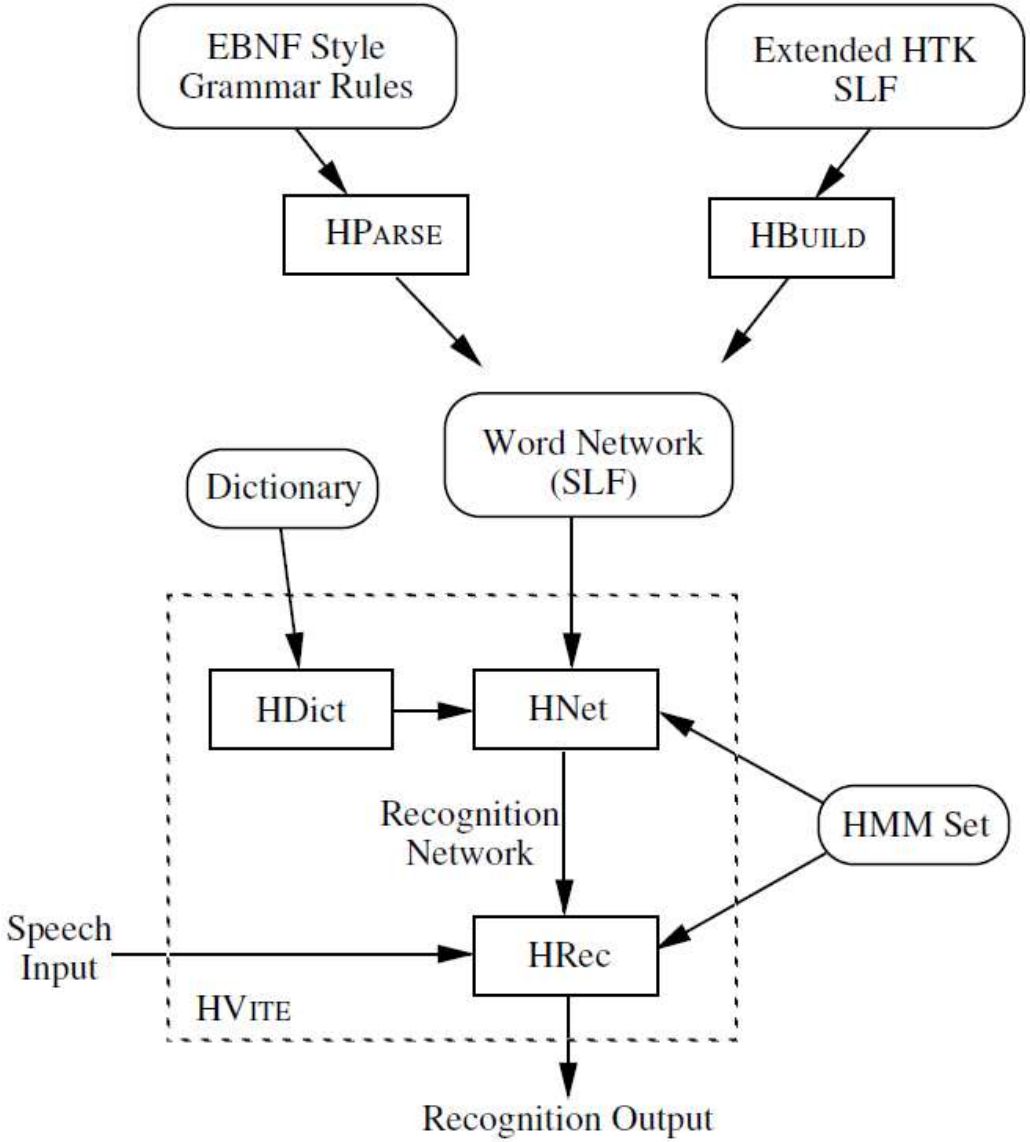
Outline

- Introduction
- How Networks are Used
- Word Networks and Standard Lattice Format
- Building a Word Network with Hparse
- Bigram Language Models
- Building a Word Network with Hbuild
- Testing a Word Network using HSGen
- Constructing a Dictionary
- Word Network Expansion

Introduction

- This chapter is concerned with building a speech recognizer using HTK
- This chapter focuses on the use of networks and dictionaries
- A network describes the sequence of words that can be recognized
- Networks are specified using the HTK Standard Lattice Format (SLF)

How Networks are Used

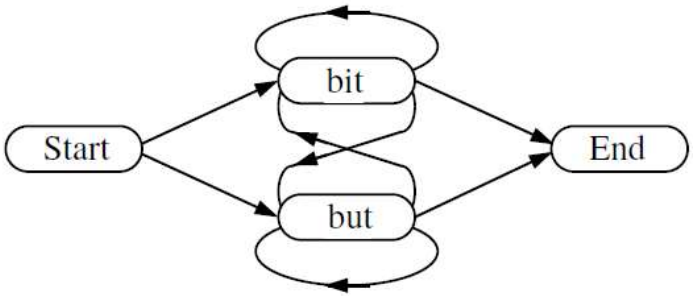


Word Networks and Standard Lattice Format

- A word network in SLF consists of a list of nodes and a list of arcs
- The nodes represent words and the arcs represent the transition between words

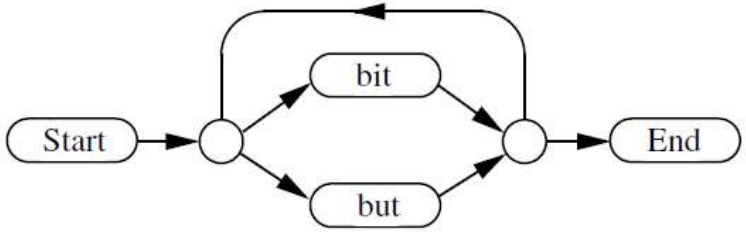
Word Networks and Standard Lattice Format

```
# Define size of network: N=num nodes and L=num arcs
N=4 L=8
# List nodes: I=node-number, W=word
I=0 W=start
I=1 W=end
I=2 W=bit
I=3 W=but
# List arcs: J=arc-number, S=start-node, E=end-node
J=0 S=0 E=2
J=1 S=0 E=3
J=2 S=3 E=1
J=3 S=2 E=1
J=4 S=2 E=3
J=5 S=3 E=3
J=6 S=3 E=2
J=7 S=2 E=2
```



Word Networks and Standard Lattice Format

```
# Network using null nodes
N=6 L=7
I=0 W=start
I=1 W=end
I=2 W=bit
I=3 W=but
I=4 W=!NULL
I=5 W=!NULL
J=0 S=0 E=4
J=1 S=4 E=2
J=2 S=4 E=3
J=3 S=2 E=5
J=4 S=3 E=5
J=5 S=5 E=4
J=6 S=5 E=1
```



Word Networks and Standard Lattice Format

- By default, all arcs are equally likely
- The optional field l=x can be used to attach the log transition probability x to an arc
- If the word "but" was twice as likely as "bit", the arcs numbered 1 and 2 in the last example could be changed to

```
J=1 S=4 E=2 l=-1.1
J=2 S=4 E=3 l=-0.4
```

Building a Word Network with HParse

- In earlier versions of HTK, a high level grammar notation based on extended Backus-Naur Form (EBNF) was used to specify recognition grammars
- This HParse format was read-in directly by the recognizer and compiled into a finite state recognition network at run-time
- HParse format grammar consists of an extended form of regular expression enclosed within parentheses

Building a Word Network with HParse

| denotes alternatives

[ ] encloses options

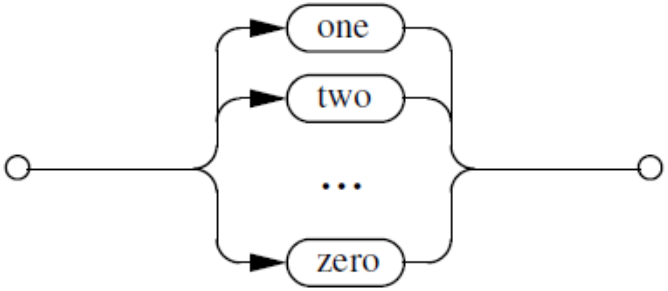
{ } denotes zero or more repetitions

< > denotes one or more repetitions

<< >> denotes context-sensitive loop

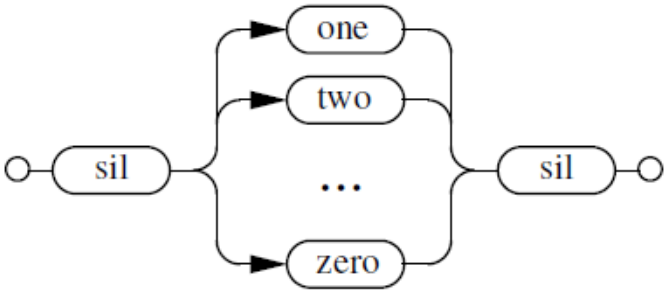
```
(
   one | two | three | four | five |
   six | seven | eight | nine | zero
)
```

```
HParse digitsyn digitnet
```



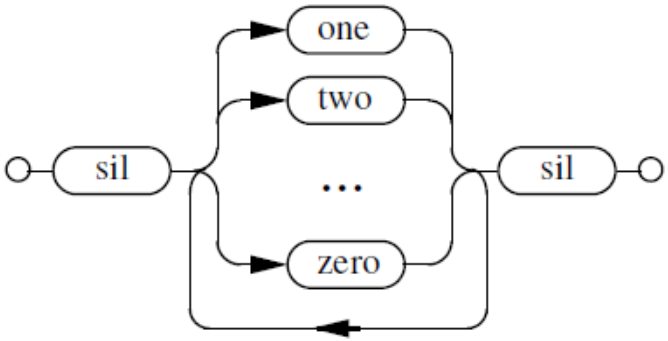

```
(
   sil (one | two | three | four | five |
   six | seven | eight | nine | zero) sil
)
```

```
HParse digitsyn digitnet
```



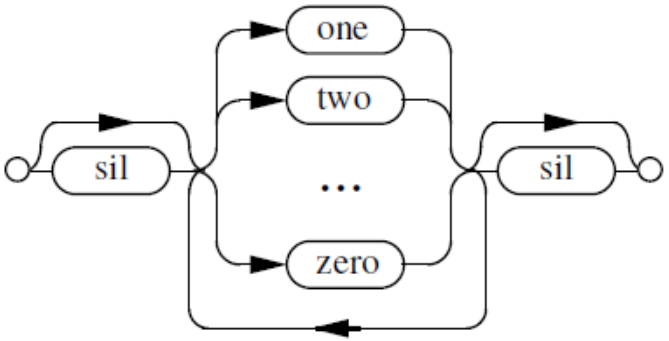

```
(
   sil < one | two | three | four | five |
   six | seven | eight | nine | zero > sil
)
```

```
HParse digitsyn digitnet
```




```
$digit = one | two | three | four | five |
         six | seven | eight | nine | zero;
(
   [sil] < $digit > [sil]
)
```

```
HParse digitsyn digitnet
```



Building a Word Network with HParse

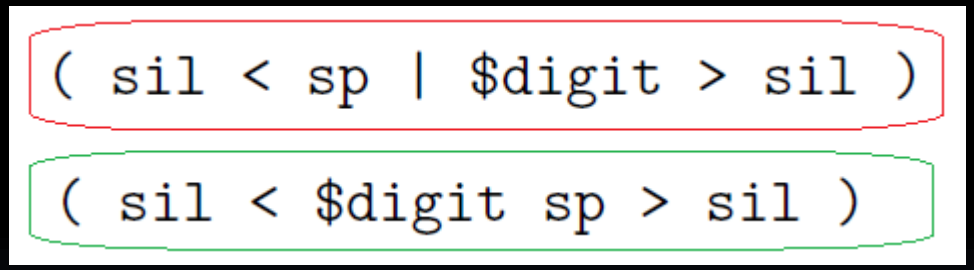
- HParse format grammars are a convenient way of specifying task grammars for interactive voice interfaces

Building a Word Network with HParse

```
$digit  = one | two | three | four | five |
          six | seven | eight | nine | zero;
$number = $digit { [pause] $digit};
$scode  = shortcode $digit $digit;
$telnum = $scode | $number;
$cmd    = dial $telnum |
          enter $scode for $number |
          redial | cancel;
$noise  = lipsmack | breath | background;
( < $cmd | $noise > )
```

Building a Word Network with HParse

- Any network containing an unbroken loop of one or more tee-models will generate an error



Bigram Language Models

- A bigram language model can be built using HLStats invoked as follows where it is a assumed that all of the label files used for training are stored in an MLF called labs

```
HLStats -b bigfn -o wordlist labs
```

- This will build a table of bigram counts in memory, and then output a back-off bigram to the file bigfn

Bigram Language Models

- The basic formulae for bigram models is given by

$$p(i,j) = \begin{cases} (N(i,j) - D)/N(i) & \text{if } N(i,j) > t \\ b(i)p(j) & \text{otherwise} \end{cases}$$

where $N(i,j)$ is the number of times word j follows word i

$N(i)$ is the number of times that word i appears

- Essentially, a small part of the available probability mass is deducted from the higher bigram counts and distributed amongst the infrequent bigrams
- When a bigram count falls below the threshold t, the bigram is backed-off to the unigram probability suitably scaled by a back-off weight in order to ensure that all bigram probabilities for a given history sum to one

Bigram Language Models

```
\data\
ngram 1=<num 1-grams>
ngram 2=<num 2-ngrams>

\1-grams:
P(!ENTER)       !ENTER   B(!ENTER)
P(W1)            W1      B(W1)
P(W2)            W2      B(W2)
...
P(!EXIT)        !EXIT    B(!EXIT)

\2-grams:
P(W1 | !ENTER)   !ENTER W1
P(W2 | !ENTER)   !ENTER W2
P(W1 | W1)       W1      W1
P(W2 | W1)       W1      W2
P(W1 | W2)       W2      W1
....
P(!EXIT | W1)    W1      !EXIT
P(!EXIT | W2)    W2      !EXIT
\end\
```

Bigram Language Models

!ENTER	0	P(W1 | !ENTER)	P(W2 | !ENTER)	.....
W1	0	P(W1 | W1)	P(W2 | W1)	.....
W2	0	P(W1 | W2)	P(W2 | W2)	.....
...				
!EXIT	0	PN	PN	.....

If there are a total of N words in the vocabulary then PN in the above is set to $1/(N+1)$

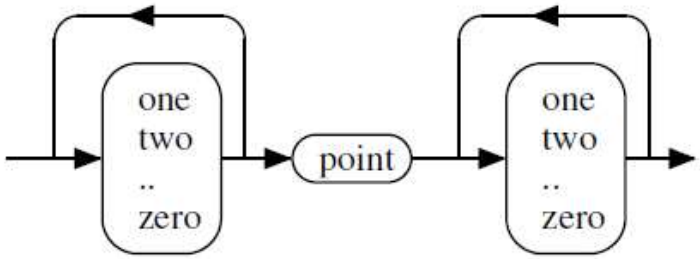
Building a Word Network with HBuild

- The main function of HBuild is to allow a word-level network to be constructed from a main lattice and a set of sub-lattices
- Any lattice can contain node definitions which refer to other lattices
- This allows a word-level recognition network to be decomposed into a number of sub-networks which can be reused at different points in the network

Building a Word Network with HBuild

```
# Digit network
SUBLAT=digits
N=14 L=21
# define digits
I=0  W=zero
I=1  W=one
I=2  W=two
...
I=9  W=nine
I=10 W=!NULL
I=11 W=!NULL
I=12 W=!NULL
I=13 W=!NULL
# null->null->digits
J=0 S=10 E=11
J=1 S=11 E=0
J=2 S=11 E=1
...
J=10 S=11 E=9
```

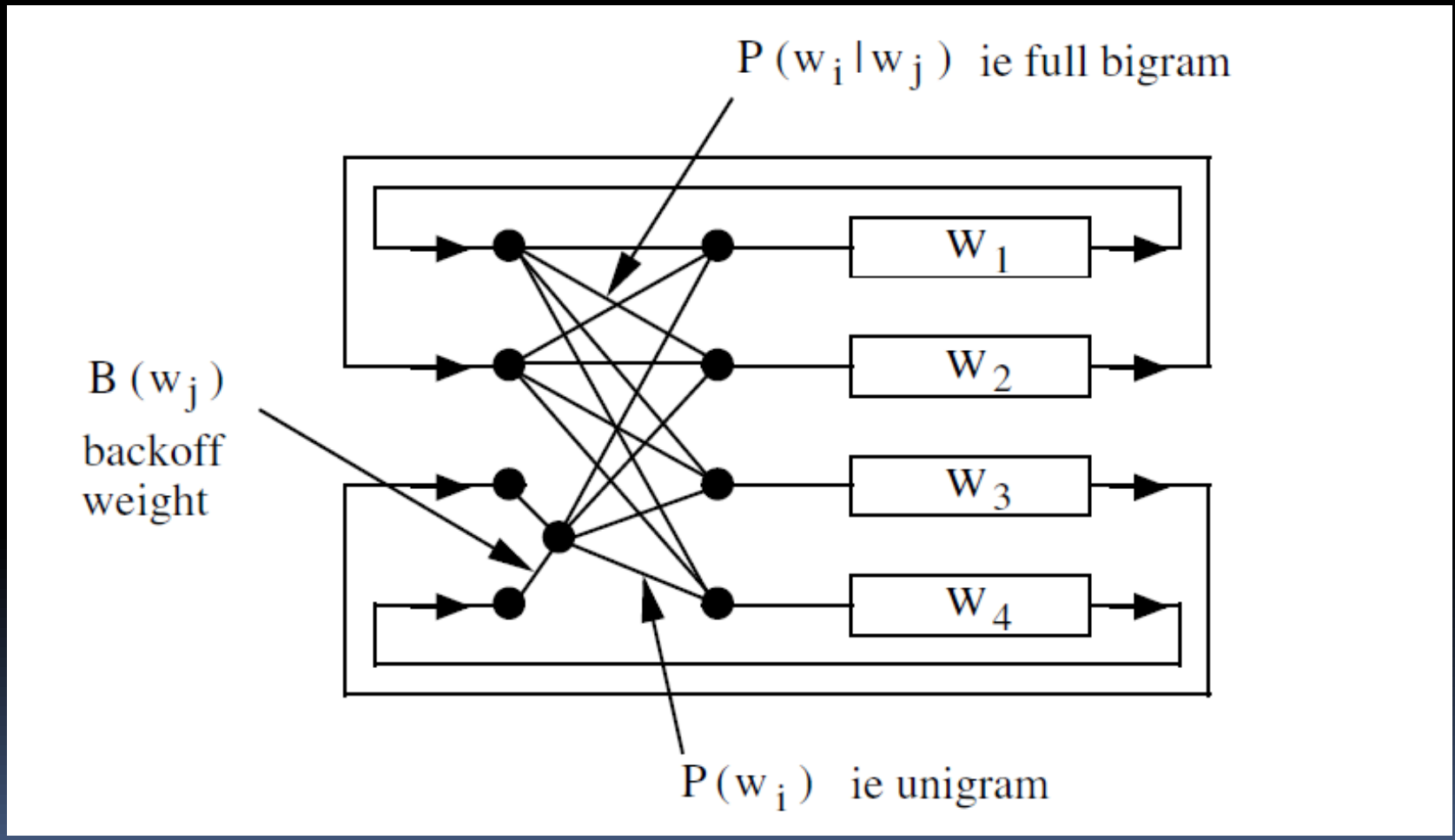
```
# digits->null->null
J=11 S=0 E=12
...
J=19 S=9 E=12
J=20 S=12 E=13
# finally add loop back
J=21 S=12 E=11
.
# Decimal netork
N=5 L=4
# digits -> point -> digits
I=0 W=start
I=1 L=digits
I=2 W=pause
I=3 L=digits
I=4 W=end
# digits -> point -> digits
J=0 S=0 E=1
J=1 S=1 E=2
J=2 S=2 E=3
J=3 S=3 E=4
```



Building a Word Network with HBuild

- One of the common form of recognition network is the word-loop where all vocabulary items are placed in parallel with a loop-back to allow any word sequence to be recognized
- HBuild can build such a loop automatically from a list of words
- It can also read in a bigram in either ARPA MIT-LL format or HTK matrix format and attach a bigram probability to each word transition
- Using a full bigram language model means that every distinct pair of words must have its own unique loop-back transition

Building a Word Network with HBuild



Testing a Word Network using HSGen

- The network can be used as a generator by randomly traversing it and outputting the name of each word node encountered
- HTK provides a very simple tool called HSGen for doing this

```
HSGen bnet bdic
start bit but bit bit bit end
start but bit but but end
start bit bit but but end
.... etc
```

Testing a Word Network using HSGen

- HSGen will also estimate the empirical entropy by recording the probability of each sentence generated

```
HSGen -s -n 1000 -q bnet bdic

Number of Nodes = 4 [0 null], Vocab Size = 4
Entropy = 1.156462,  Perplexity = 2.229102
1000 Sentences: average len = 5.1, min=3, max=19
```

Constructing a Dictionary

- The way in which each word is expanded is determined from a dictionary
- Dictionary format

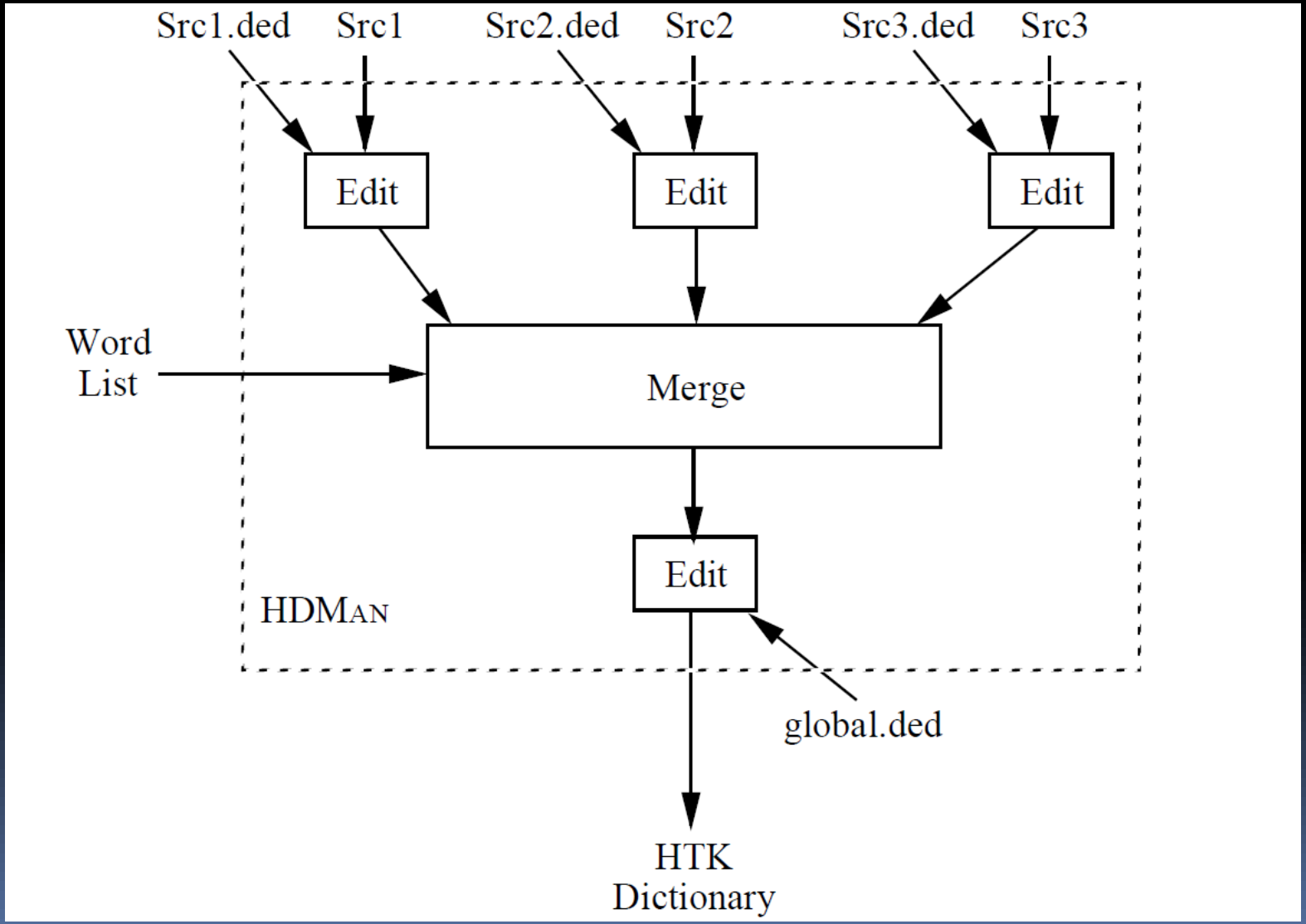
```
WORD [ '['OUTSYM']' ] [PRONPROB] P1 P2 P3 P4 ....
```

bit		b	ih	t
but		b	ah	t
dog	[woof]	d	ao	g
cat	[meow]	k	ae	t
start	[]	sil		
end	[]	sil		

the	th	iy
the	th	ax

bit	b+ih	b-ih+t	ih-t
but	b+ah	b-ah+t	ah-t

Constructing a Dictionary



Constructing a Dictionary

```
ih0 d  #    ->    ax d
ih0         ->    ih  (otherwise)

MP axd0 ih0 d #
SP axd0 ax d #
RP ih ih0
```

all examples of `ax l` (as in "bottle") are to be replaced by the single phone `el` provided that the immediately following phone is a non-vowel

```
DC nonv l r w y .... m n ng #
MP axl ax l
CR el * axl nonv
SP axl ax l
```

convert all phones to context-dependent form and append a short pause model `sp` at the end

```
TC
AS sp
```

```
BAT b ah t
    ⇩
BAT b+ah b-ah+t ah-t sp
```

Word Network Expansion

- The conversion of word level networks to model-based recognition networks is performed by the HTK module Hnet
- HNet attempts to infer the required expansion from the contents of the dictionary and the associated list of HMMs
- The expansion proceeds in four stages

Word Network Expansion

- Context definition:
- The first step is to determine how model names are constructed from the dictionary entries and whether cross-word context expansion should be performed
  - Context Free: the phone is skipped when determining context
  - Context Independent: The phone only exists in context-independent form
  - Context Dependent: This classification depends on whether a phone appears in the context part of the name and whether any context dependent versions of the phone exist in the HMMSet

Word Network Expansion

- Determination of network type
- If the dictionary is closed (every phone name appears in the HMM list), then no expansion of phone names is performed
- If the dictionary is not closed, then if word internal context expansion would find each model in the HMM set then word internal context expansion is used
- Otherwise, full cross-word context expansion is applied

Word Network Expansion

- Network expansion
- Each word in the word network is transformed into a word-end node preceded by the sequence of model nodes corresponding to the word's pronunciation
- For cross word context expansion, the initial and final context dependent phones are duplicated as many times as is necessary to cater for each different cross word context

Word Network Expansion

- Linking of models to network nodes
- Each model node is linked to the corresponding HMM definition
- FORCECXTEXP is true
  - Construct the context-dependent name and see if the corresponding model exists
- ALLOWCXTEXP is false
  - Construct the context-independent name and see if the corresponding model exists

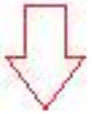
Word Network Expansion

sil aa r sp y uw sp sil

sil sil-aa+r aa-r+y sp r-y+uw y-uw+sil sp sil

CFWORDBOUNDARY set true

aa r sp y uw sp



aa+r aa-r sp y+uw y-uw sp

Setting CFWORDBOUNDARY false would produce

aa+r aa-r+y sp r-y+uw y-uw sp

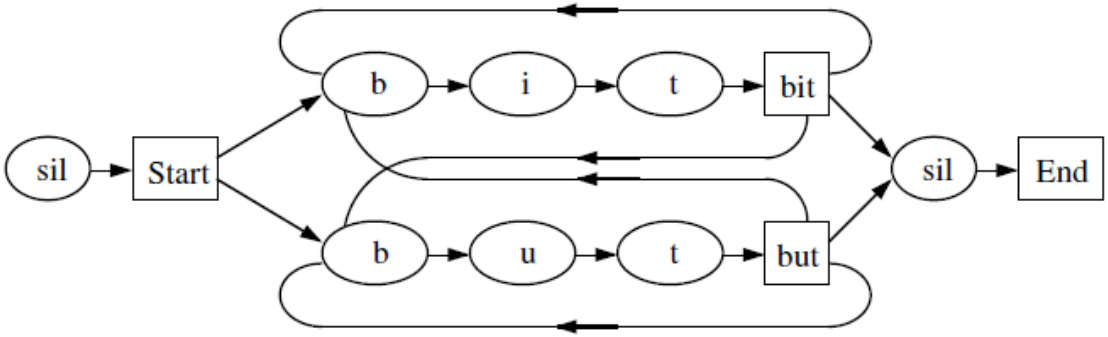
Word Network Expansion

Dictionary

bit	b i t
but	b u t
start	sil
end	sil

HMM Set

b i t u sil

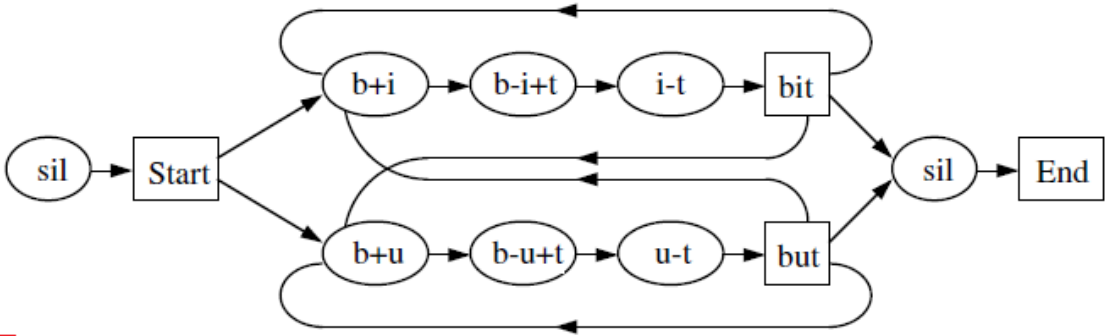


Dictionary

bit	b+i	b-i+t	i-t
but	b+u	b-u+t	u-t
start	sil		
end	sil		

HMM Set

b+i b-i+t i-t b+u b-u+t u-t sil



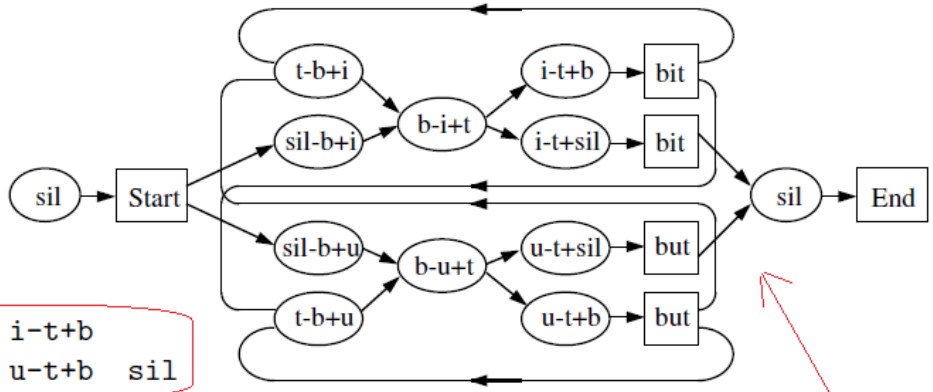
Word Network Expansion

Dictionary

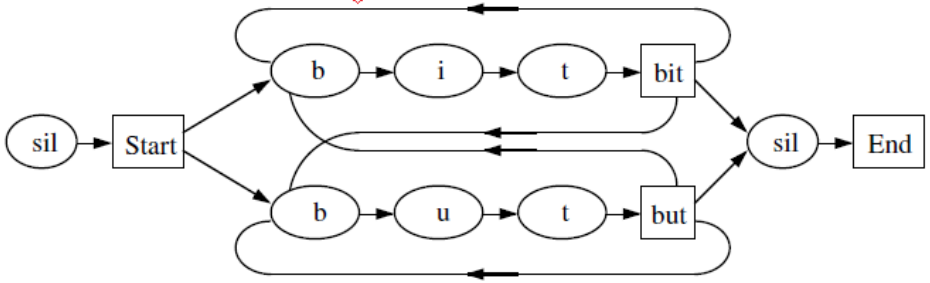
```
bit         b  i  t
but         b  u  t
start       sil
end         sil
```

HMM Set

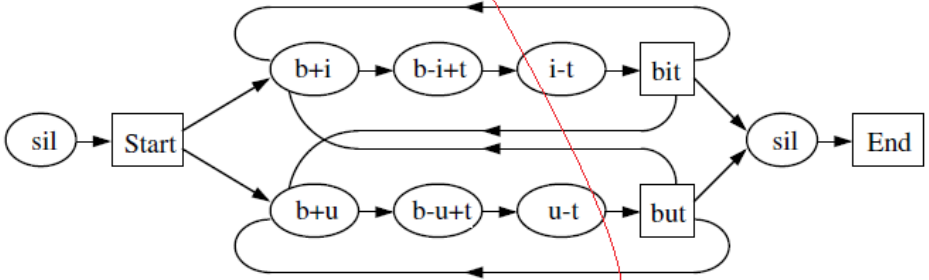
```
sil-b+i  t-b+i  b-i+t  i-t+sil  i-t+b
sil-b+u  t-b+u  b-u+t  u-t+sil  u-t+b  sil
```



If HMM Set contains all monophones, biphones and triphones



FORCEXTEXP is True and ALLOWXWRDEXP is false



FORCEXTEXP and ALLOWXWRDEXP are both True

Thank You