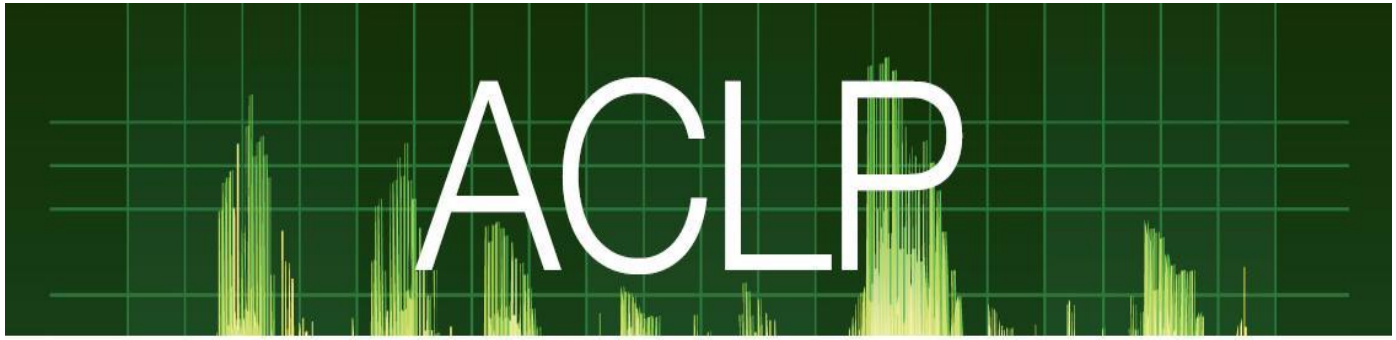
An Efficient Distance Measure for Comparing Phoneme Sequences



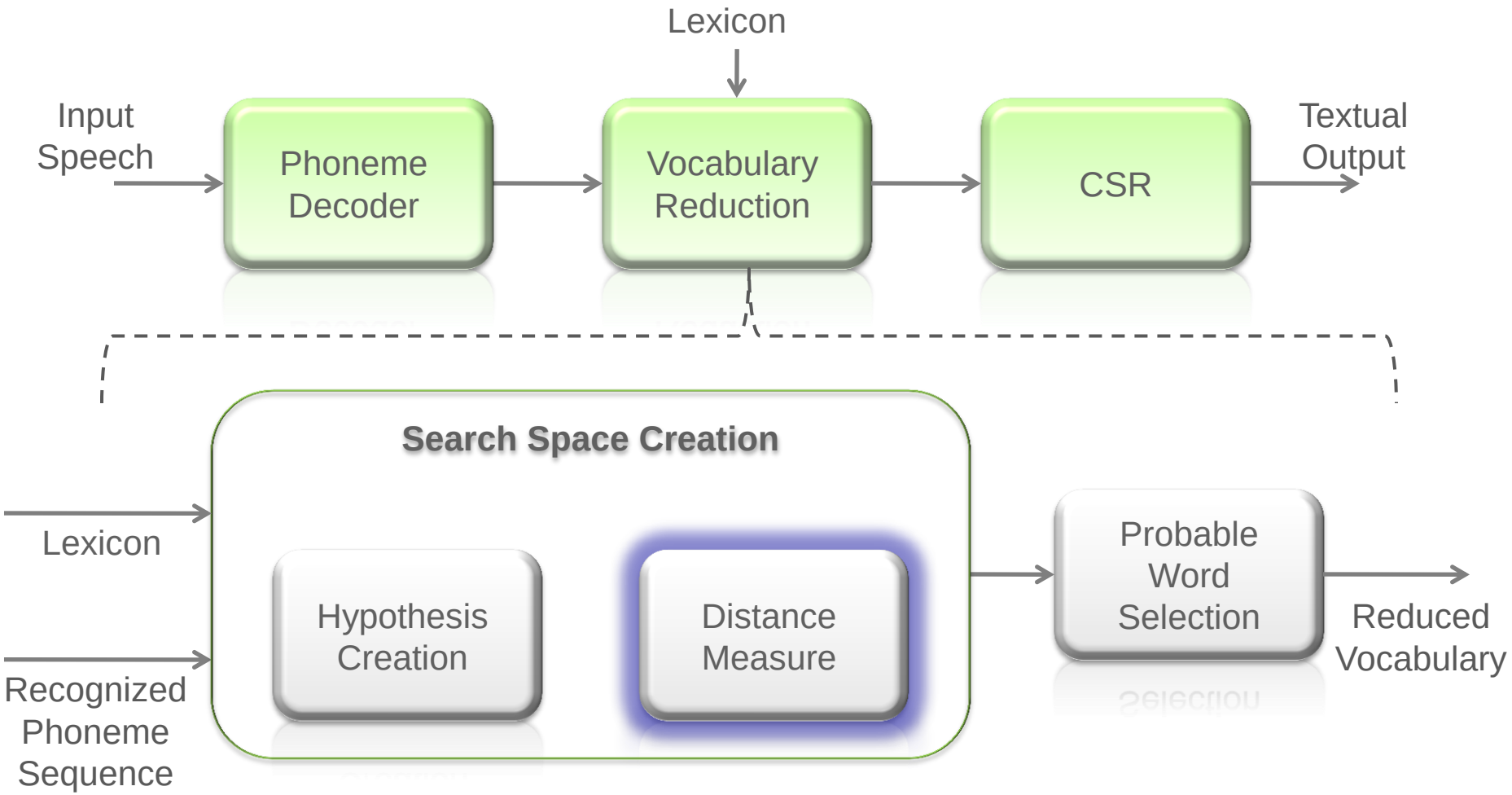
Afeka Center for Language Processing (ACLP)

Baruchi Har-Lev

Speech Recognition Day – 8/6/2010

AFEKA TEL-AVIV ACADEMIC COLLEGE OF ENGINEERING

Multi-Stage LVCSR



Overview
The Problem

Distance measure
$O(n^2)$

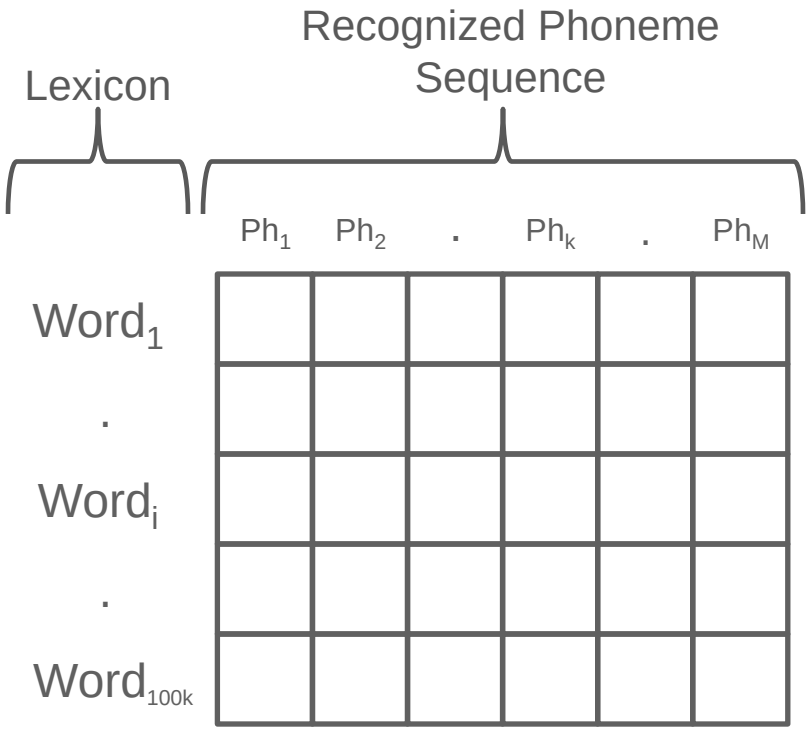
Example

Lexicon size = 100k

Phoneme sequence length = 250

Grid size = 25M

Avg. hypothesis length = 7



Number of Operations > 1,000,000,000

Overview

The Goal

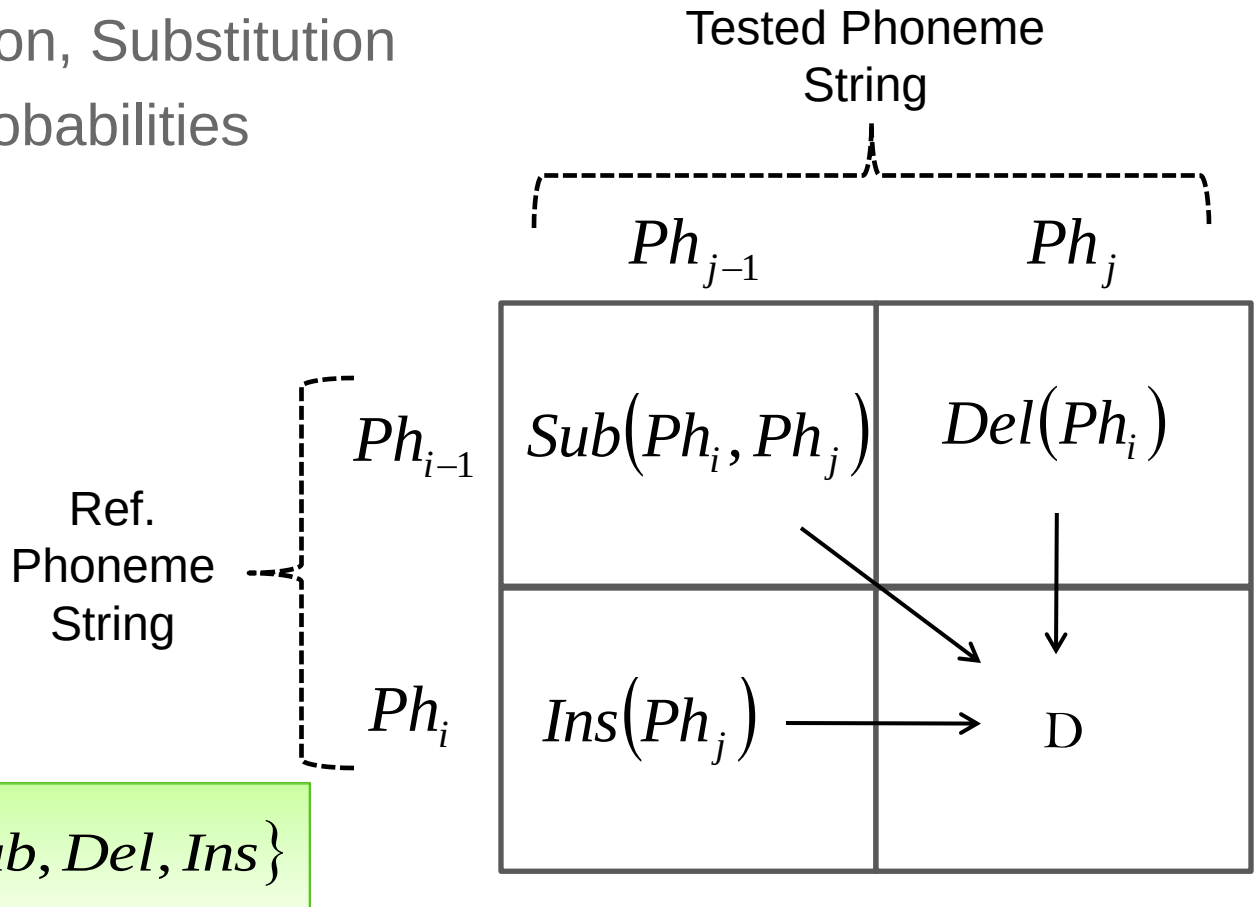
- Reduce computational complexity of the distance measure while maintaining minimum performances loss

Test Data Base

Parameter	Macrophone	VoiceMail
# of Utterances	3140	2485
Utterance Length	3.5 sec	20+ sec
# of Words in Utterance	8-9	70-80
# of Phonemes in Utterance	40	250

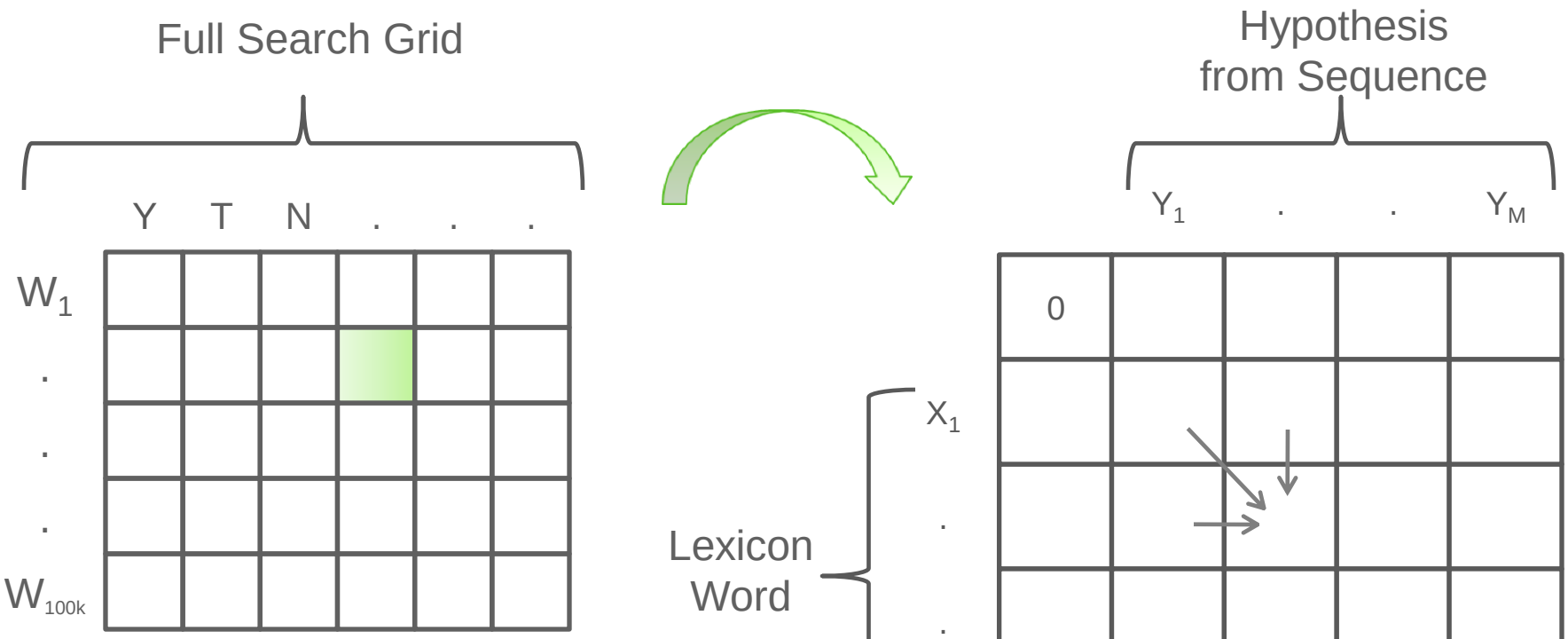
Levenshtein Distance

- Insertion, Deletion, Substitution
- Equal weight probabilities
- $O(NM) \approx O(n^2)$



$$D(Ph_i, Ph_j) = Min\{Sub, Del, Ins\}$$

Levenshtein Distance in the Exhaustive Process



$$Number\ of\ D(W_i, Hyp_j):$$

$$Grid\ Size \times O(n^2)$$

Results

DB	Mean Sequence Size [Phones]	Lexicon Size	Reduced Vocabulary Size
Macrophone	41	100k	50k

Distance Method	Creation Time [sec]	Coverage [%]
Levenshtein Distance	**8**	**85%**

A-Priori Distance Differences

- Choose a canonic word - Word0
- Compute distances (Levenshtein)
  - A-priori distance computation
  - Real time computation
  - Sum of results

A-priori

One Time in RT

$$D(Wi, Hyp) \leq \overbrace{D(Wi, Word_0)}^{\text{A-priori}} + \overbrace{D(Word_0, Hyp)}^{\text{One Time in RT}}$$

A-Priori Distance Differences

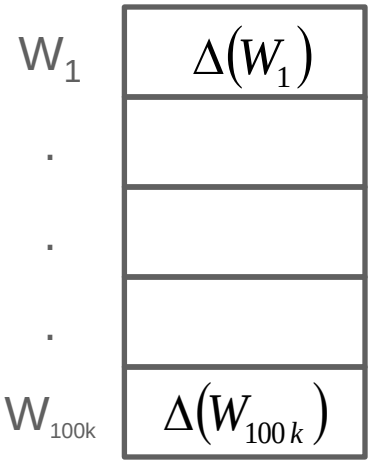
Full Search

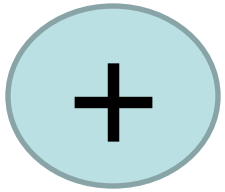
A-priori

RT

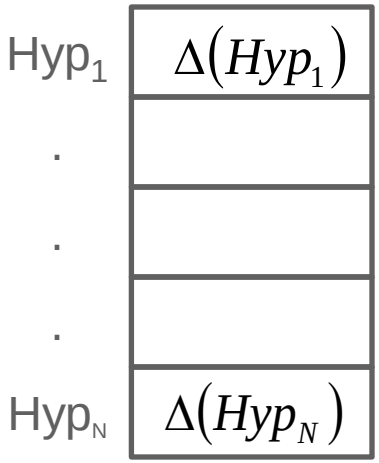
$D(Wi, Word_0)$

$D(Hyp_i, Word_0)$





O(Grid Size)



O(#Hyp) → O(1)

Results

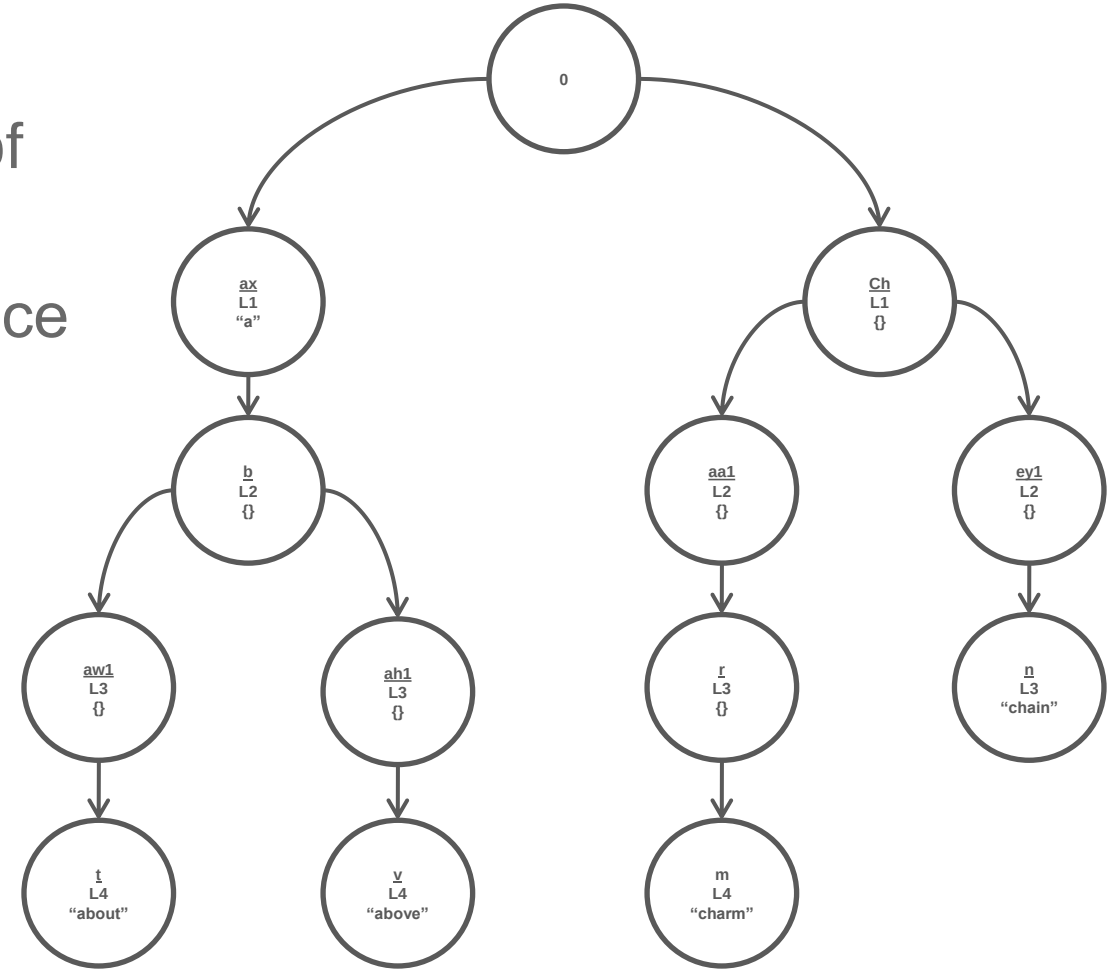
DB	Mean Sequence Size [Phones]	Lexicon Size	Reduced Vocabulary Size
Macrophone	41	100k	50k

Distance Method	Creation Time [sec]	Coverage [%]
Levenshtein Distance	8	85%
A-priori Differences	**0.32**	**53%**

96% Time Reduction, 32% Coverage Loss

Tree Distance

- Advantages:
  - Reduces number of hypotheses
  - “Breaks” the distance measure
- Disadvantages:
  - Complex interface



Results

DB	Mean Sequence Size [Phones]	Lexicon Size	Reduced Vocabulary Size
Macrophone	41	100k	50k

Distance Method	Creation Time [sec]	Coverage [%]
Levenshtein Distance	8	85%
A-priori Differences	0.32	53%
Tree Distance	**8**	**85%**

No Improvement

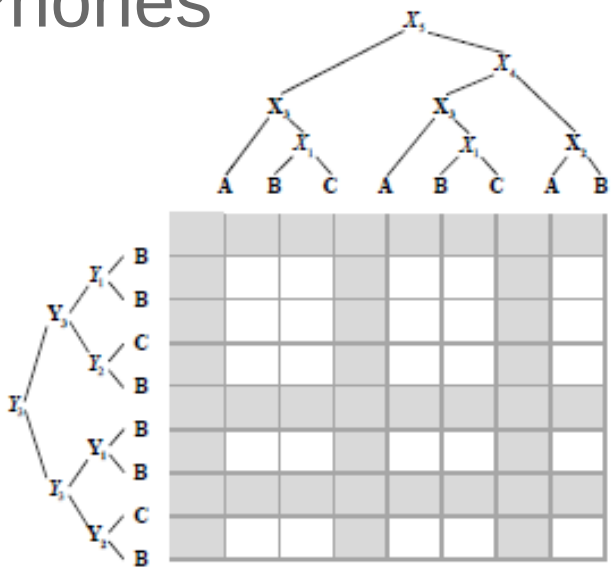
A Glimpse at Different Disciplines

Stringology – Computer science

- $O(n^{1.4}N^{1.2})$
  - Good compression rate
- Avg. length (Lexicon Word) ≈ 7 Phones
  - Bad compression rate

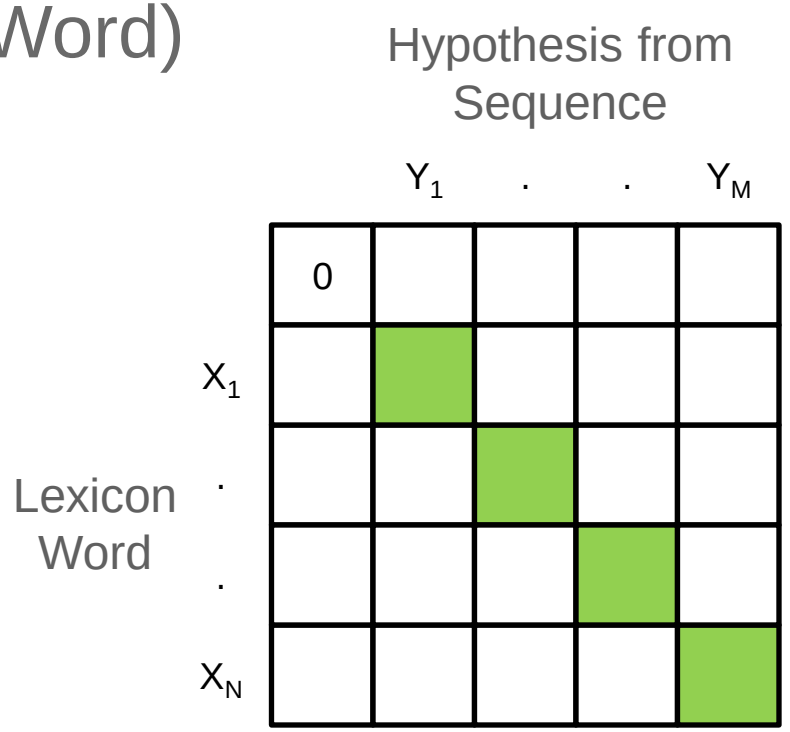
Biology

- Blast
  - Computer network



Diagonal Distance

- Length(Hyp.) = Length(Word)
- Basic comparisons
- Uses substitutions only
- Does not use:
  - Insertion
  - Deletion



Results

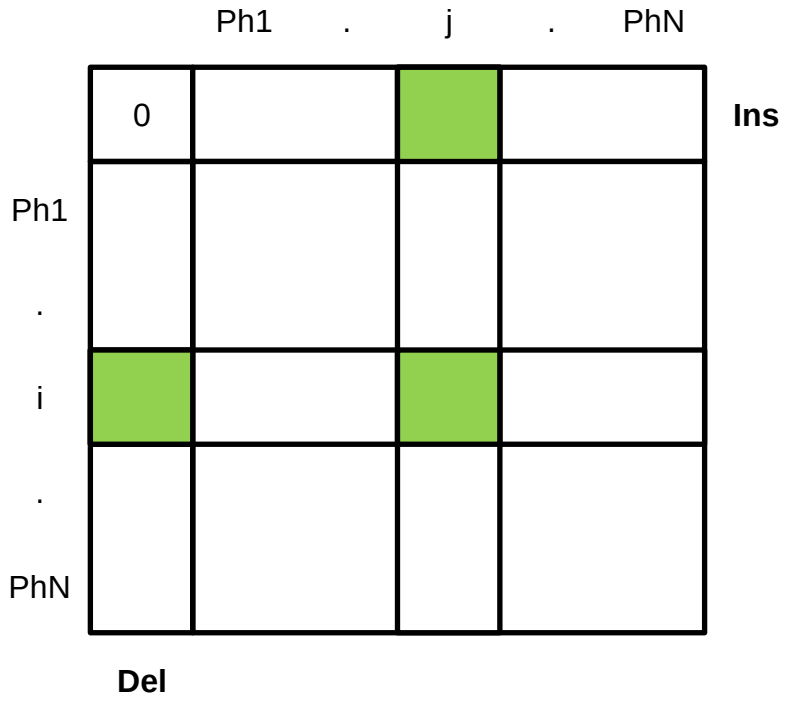
DB	Mean Sequence Size [Phones]	Lexicon Size	Reduced Vocabulary Size
Macrophone	41	100k	50k

Distance Method	Creation Time [sec]	Coverage [%]
Levenshtein Distance	8	85%
A-priori Differences	0.32	53%
Tree Distance	8	85%
Diagonal Distance	**0.9**	**83%**

90% Time Reduction, 2% Coverage Loss

Weighted Distance

- Additional data - Phoneme engine characteristics
  - Confusion matrix
  - Posteriori probabilities
  - Larger distinction between phones
- Disadvantage
  - Sensitivity to phoneme engine characteristics



Results

DB	Mean Sequence Size [Phones]	Lexicon Size	Reduced Vocabulary Size
Macrophone	41	100k	50k

Distance Method	Creation Time [sec]	Coverage [%]
Levenshtein Distance	8	85%
A-priori Differences	0.32	53%
Tree Distance	8	85%
Diagonal Distance	0.9	83%
Diagonal Weighted Distance	**0.9**	**90%**

90% Time Reduction, 5% Coverage Improvement

Summary

- Vocabulary reduction
  - Problems and Goals
- Levenshtein distance method
  - Very complex in exhaustive search
- Improving the distance method
  - Best method – Weighted diagonal distance method
- Next step

THANK YOU
FOR YOUR ATTENTION