[Return to Main](#)

[Objectives](#)

Stack Decoding:
[Best-First](#)
[Admissible Heuristics](#)
[Fast Match](#)

Multi-pass Search:
[N-Best Generation](#)
[Lattice Generation](#)

Other:
Course Evaluations

On-Line Resources:
[JA: Stack](#)
[AJR: Search](#)
[SR: Search](#)

LECTURE 36: STACK DECODING

- Objectives:
  - Best-first search with admissible heuristics
  - Fast matching
  - Cross-word decoding and lexical trees
  - N-best and word graph generation

This lecture follows the course textbook closely:

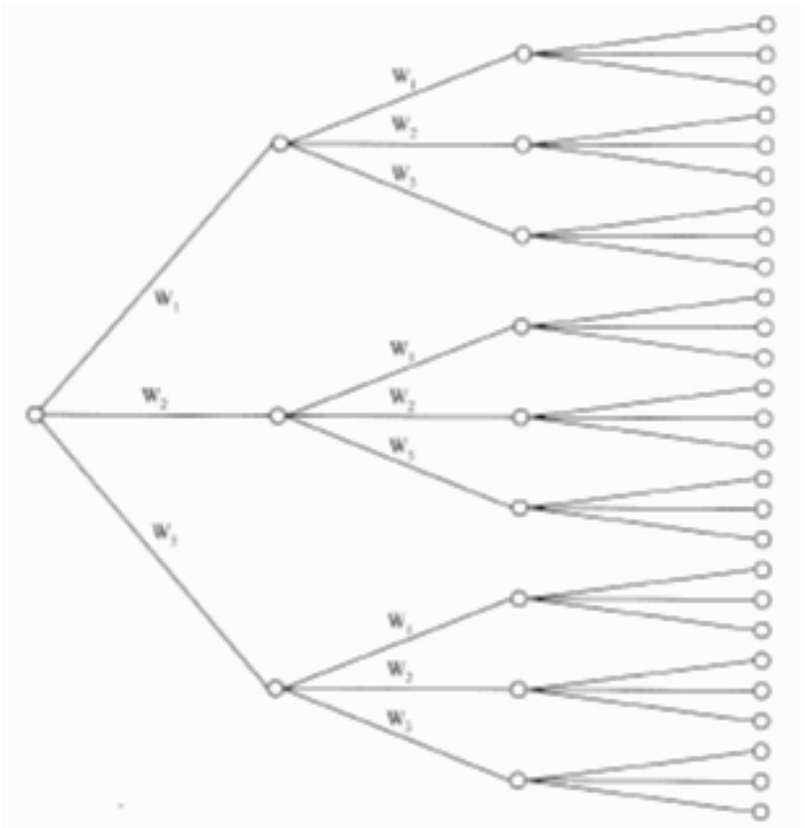
> X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

Another good source for some of this information is:

> F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Boston, Massachusetts, USA, ISBN: 0-262-10066-5, 1998.

STACK DECODING (A* SEARCH)

- If some heuristics are available to guide the decoding, the search can be done in a depth-first fashion around the best path.
- We can avoid wasting computation on unpromising paths via time synchronous decoding.
- Such a heuristic function is very difficult to attain in speech recognition since it must combine elements of acoustic and language model scoring.
- **Stack decoding** is a variant of tree search.



- Note that the Viterbi search finds the optimal state sequence while stack decoding focuses on the optimal word sequence.
- The search process can be summarized as follows:
  - Add all possible one-word sequences to the OPEN list.
  - Remove the best path from the OPEN list; all paths from it are extended, evaluated, and placed back in the OPEN list (sorted).
  - Continue until a complete path that is guaranteed to be better than any path on the OPEN list has been found.
- Two key operations:
  - Finding an effective heuristic function for estimating the probability of the "future" part of the path.
  - Determining when to extend the search to the next word/phone.

ADMISSIBLE HEURISTICS

- Recall the general form of our evaluation function:

  $$f(H) = g(H) + h(H)$$

  Where g() represents the evaluation function for the partial path up to time t, and h() represents the estimate of the remaining path.

- An admissible heuristic function is one that always underestimates the true cost of the remaining path (e.g., a zero function).

- The evaluation function can simply be the forward probability.

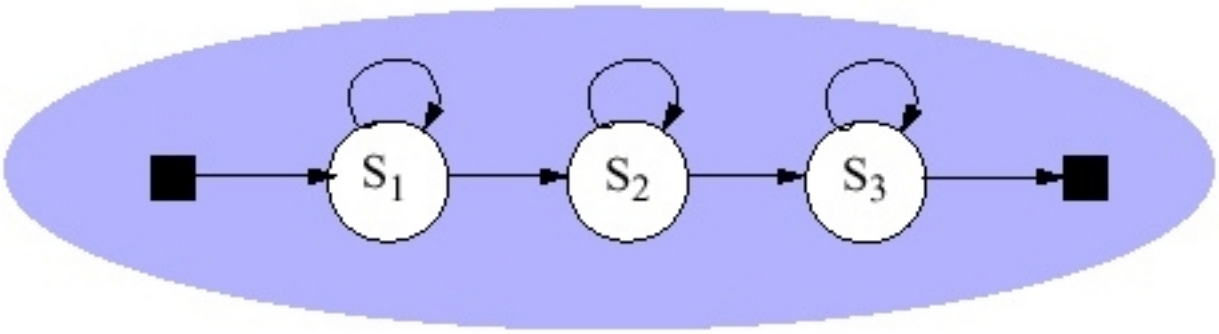
- The expected cost of the remaining part of the path can be estimated by gathering statistics from the training data:

  $$h(H) = (T-t)P_{min}$$

- It can be shown that this same heuristic can be applied to the problem of extending the path into the next word.

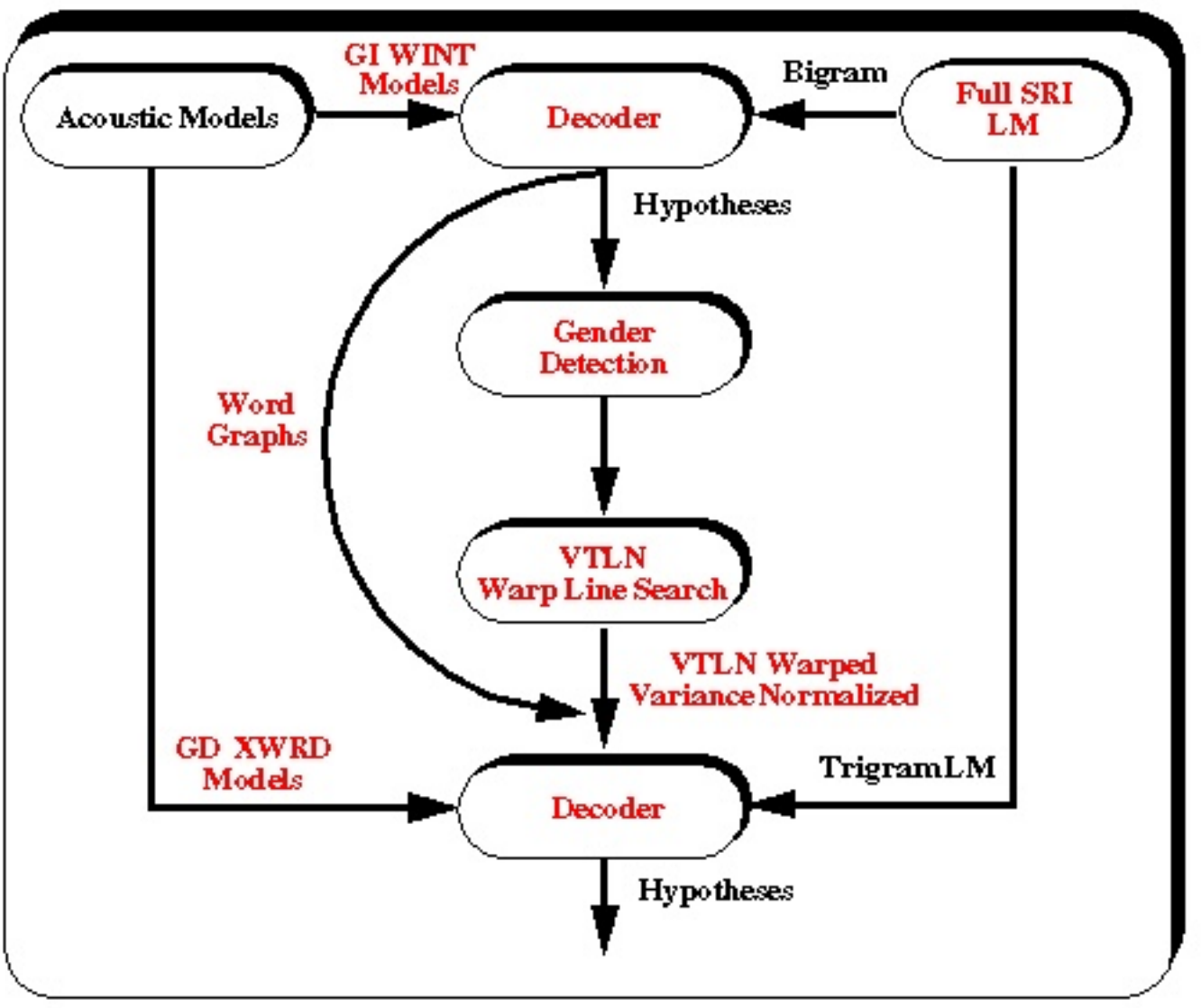
FAST MATCH IS CRITICAL IN STACK DECODING

- An effective underestimated heuristic function for the remaining portion of speech is very difficult to derive.
- In asynchronous stack decoding, the most expensive step is to extend the best subpath.
- **Fast match** is a method for the rapid computation of a list of candidates that constrain successive search phases.
- Fast match can be regarded as an additional pruning theshold to meet before a new word or phone can be started.
- A fast match method is called admissible if it never prunes away the optimal path.
- One popular fast match approach is to estimate the probability a phone model by using the "straight-thru" path:



N-BEST DECODING IS USEFUL FOR INTEGRATING KNOWLEDGE SOURCES

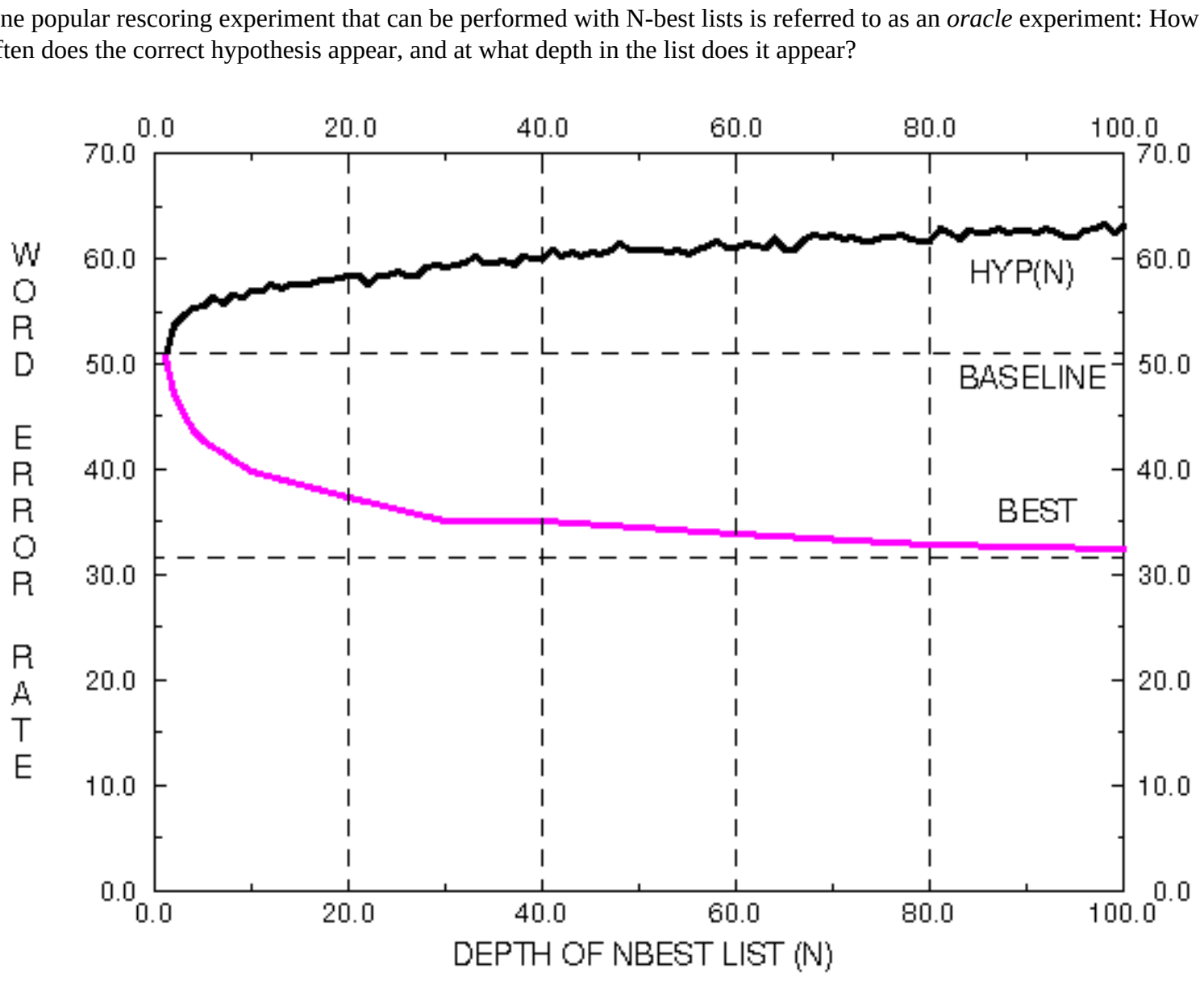
- Often the search space becomes unmanageable for real problems due to complex language model constraints (e.g., trigrams) and acoustic models (e.g., cross-word context-dependent phones), and our interest in integrating multiple knowledge sources (e.g., phrase structured grammars).
- A pragmatic alternative is to use a multipass search approach:



- It is common to perform a first pass of decoding with a bigram language model (or a word-internal triphone/trigram system), and postprocess (or rescore) the output with a more sophisticated system. We refer to this process as **multipass** decoding.
- Stack decoding is naturally suited to generating N-best lists. Consider this example from the North American Business (NAB) Corpus:

1.	I will tell you would I think in my office
2.	I will tell you what I think in my office
3.	I will tell you when I think in my office
4.	I would sell you would I think in my office
5.	I would sell you what I think in my office
6.	I would sell you when I think in my office
7.	I will tell you that I think in my office
8.	I will tell you why I think in my office
9.	I will tell you would I think on my office
10.	I Wilson you think on my office

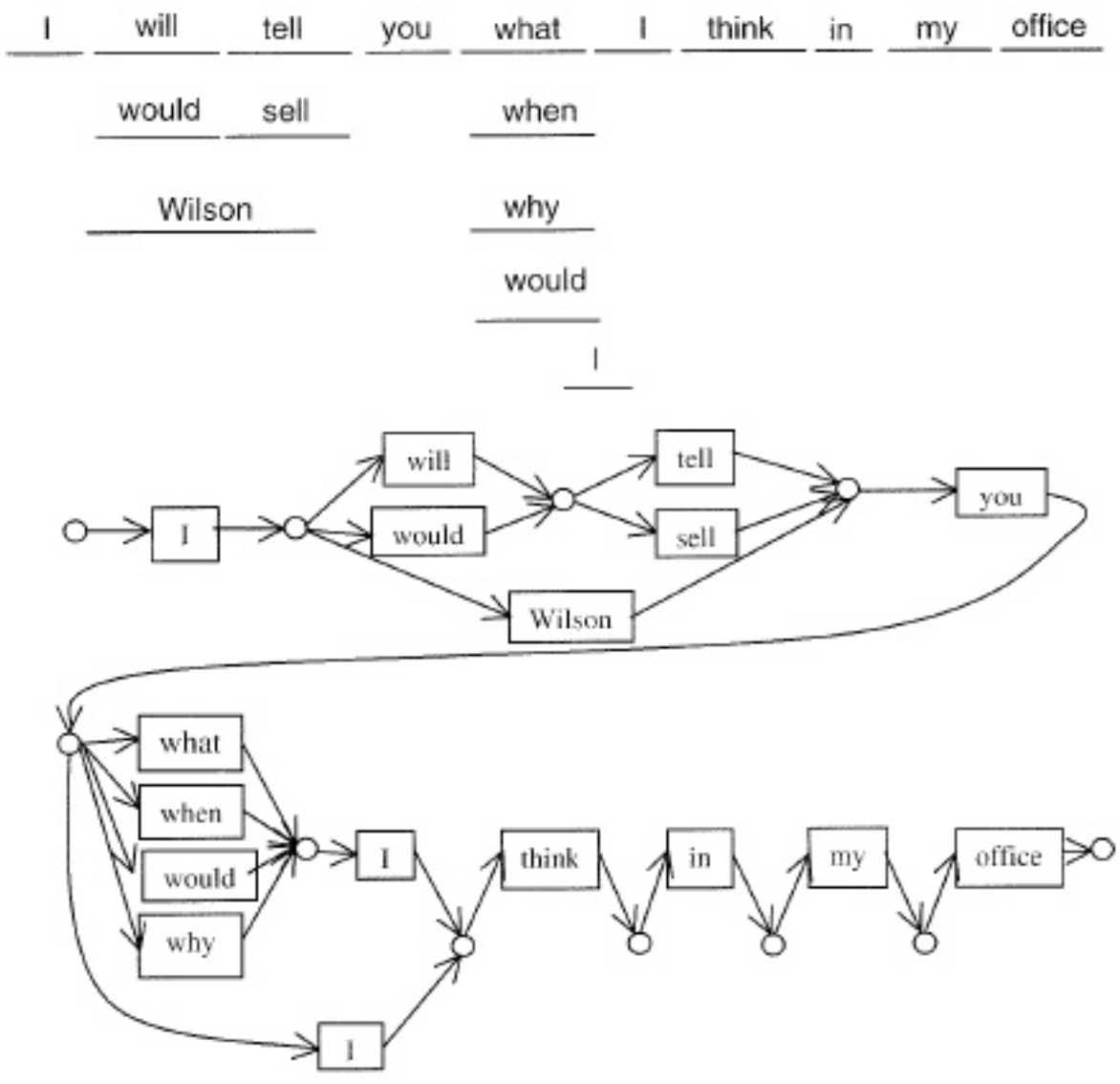
- N-best lists are very compact representations of the search space since timing information is discarded.
- One popular rescoring experiment that can be performed with N-best lists is referred to as an *oracle* experiment: How often does the correct hypothesis appear, and at what depth in the list does it appear?



- Oracle experiments are one form of a *cheating* experiment. Cheating experiments are very important diagnostic tools.

WORD GRAPH GENERATION AND RESCORING

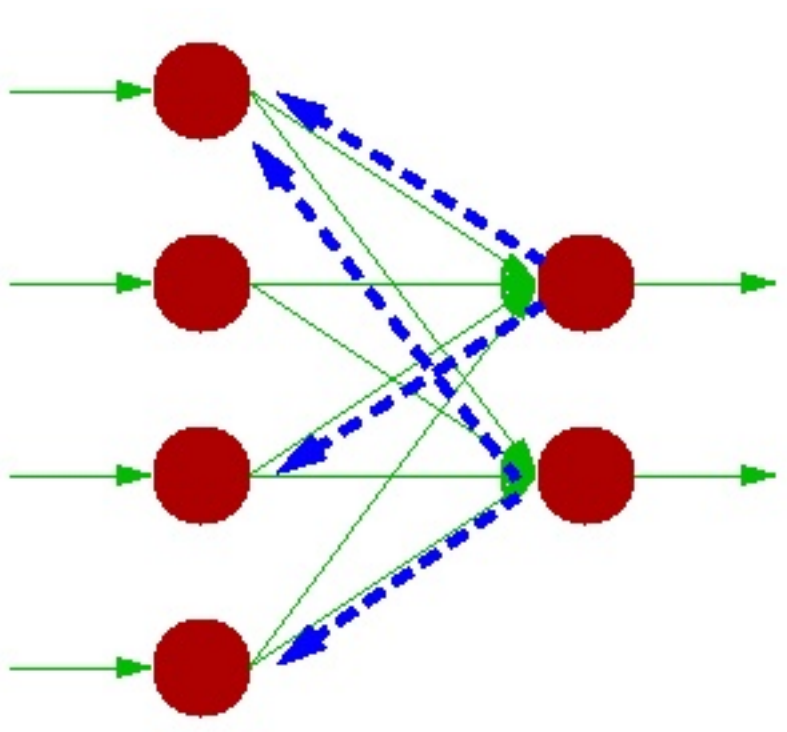
- An N-best list can be viewed as a graph:



This representation is often referred to as a **lattice**.

- How might we generate such a graph using time synchronous Viterbi decoding?

- Solution: keep multiple choices at each node in the graph during the dynamic programming step:



- It is hard to underestimate the impact word graph rescoring has had on speech recognition research. However. these graphs are very large and take at least an order of magnitude more time to generate (than the one-best choice). Why are these worth the trouble?

- Word graphs can be very large: 10 to 50 MBytes per file; 1 Gbyte or more per corpus.

- What figure of merit can we use to describe such graphs? (Hint: lattice word error rate) Explain the significance of this measure.