MPhil in Machine Learning and Machine Intelligence

Module MLMI2: Speech Recognition

L9: LVCSR Search

Phil Woodland
pcw@eng.cam.ac.uk

Michaelmas 2021

Cambridge University Engineering Department

Introduction

In speech recognition we find the word sequence, **W**, that maximises $P(\mathbf{W}|\mathbf{O})$ where **O** is the observed speech. The search (**decoder**) component finds this optimal **W**.

With HMMs/*N*-gram **all the knowledge** (constraints) can be represented in a network form:

- Phone based HMMs (possibly context dependent)
- Pronunciation lexicon (dictionary)
- Language model

and in principle all the knowledge present can be compiled into a large HMM and the best (most likely) path found through the network.

Here we discuss issues important for large vocabulary recognition, esp with *N*-gram LMs. Topics covered include:

- Search network structure
- Language Model Application
- Decoding strategies

In many cases **multiple recognition outputs** are required. These can have many uses including allowing the use of multiple acoustic models and language models. We discuss:

- Generating Multiple Solutions
- Word Lattices
- Confusion Networks
- System (Hypothesis) Combination / Ensemble Methods

The ideal decoder

An ideal decoder would be:

Efficient - Computation resources must be reasonable. For interactive use decoding must be better than real-time. Already seen **beam search** and **token passing**. Also need to consider **delay** or **latency** to get output.

Accurate - Decoder should always find the most likely state sequence- **admissible**. In order to keep the compute time down (e.g. using beam-search) almost always not possible. This results in **search errors**, which should be kept to a minimum. In practice, there is a trade-off between compute-time and WER.

Scalable - Should be able to handle large vocabulary sizes.

Versatile - Needs to be able to handle a variety of knowledge sources. For example *N*-gram language models and context dependent acoustic models.

Results - Should be able to produce 1-best, *N*-best or lattices.

Time synchronous decoders using Viterbi recognition will be mainly discussed - in particular using the **token passing** paradigm.

Simple Systems

To illustrate some of the search issues a simple 4 word vocabulary system will be used.

Lexicon (monophones):

AND	/ **ae n d** /
BILL	/ **b ih l** /
BIT	/ **b ih t** /
BEN	/ **b eh n** /

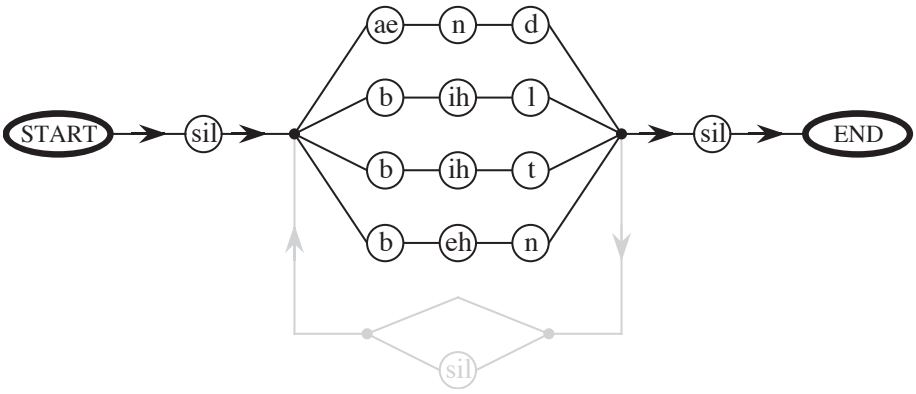
Using this example consider the problems involved in:

- Finite-state network (and unigram language model)
- Bigram language model
- Trigram language model
- Cross-word triphone context dependent models

Finite-State Network

Initially consider a system with:

- Monophone acoustic models + Finite state grammar



This is shown in the figure (link in grey is extension to continuous speech)

To extend this to use a unigram language model:

- Add the language model probability (e.g. $\log(P_{BEN})$) to the log-likelihood of the token at the start of each word.
- Language model information used as soon as possible to aid pruning (beam search).

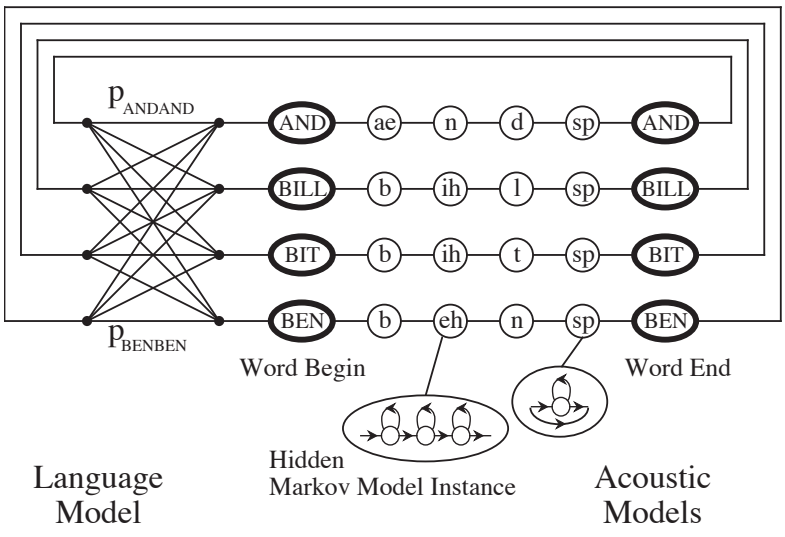
Bigram Language Model

Expanding to use bigrams is more complex:

- Language model probability is dependent on the previous word

Cannot have a single return silence loop.

- Apply bigram probability to start of each word
- Add separate loop back for each word
- Add optional silence model (*sp*) to end of each word.



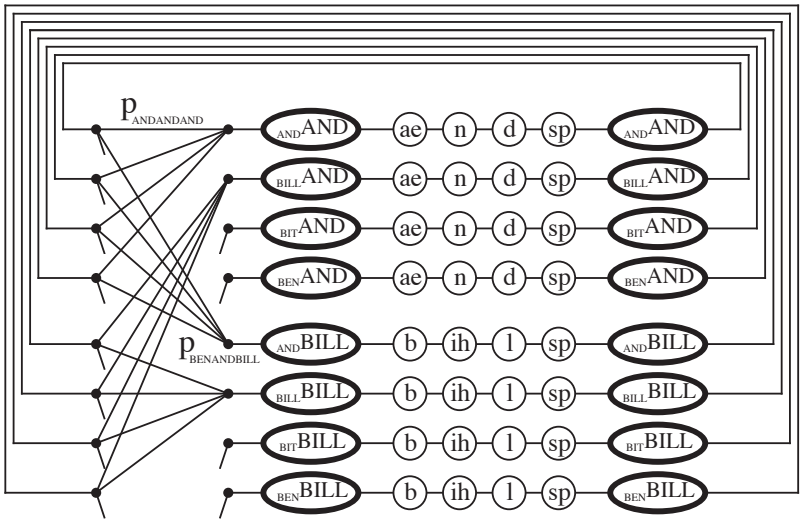
- Extension to word internal triphones is simple.

Trigram Language Model

Expanding to use trigrams is even more complex:

- Language model probability is dependent on the previous two words

- Duplicate each word according to previous word
- Apply trigram probability to start of each “duplicate” word
- Add separate loop back for each “duplicate” word



- Extension to word internal triphones is simple.

Cross Word Models

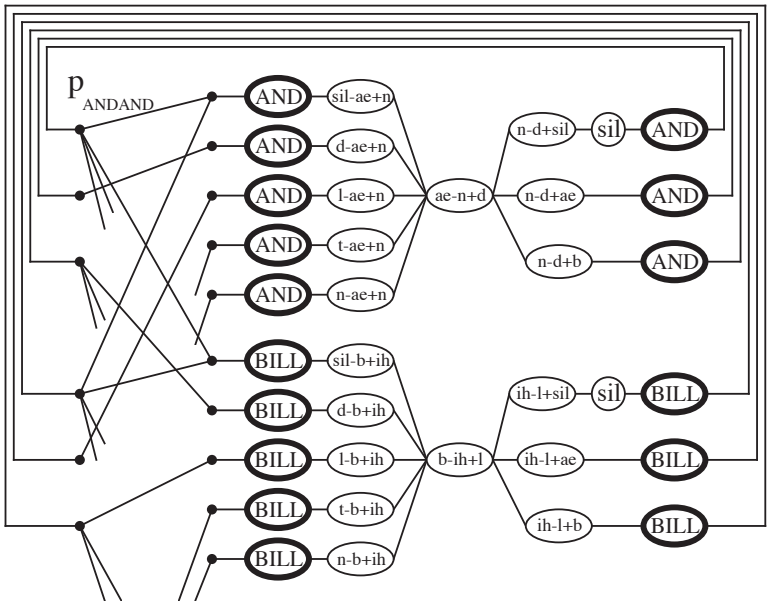
Consider a system with:

- Cross word triphone acoustic models + Bigram language model

Modify previous bigram network

- Make use of inter-word silence explicit in network
- Duplicate first phone of word according to the last phone of the previous word
- Duplicate last phone of word according to the first phone of the next word

`HVite` automatically expands search networks for cross-word triphones.



Asymmetry in Pruned Search

When a decoder uses pruning with a fully-connected *N*-gram LM, found that most of the search effort is concentrated in the first phones of each word and relatively little at the word ends.

For example the average number of active phones at different word positions for a 5000 word WSJ task (word-internal triphones with bigram) is shown below.

Model position in word	First	Second	Last but one	Last	Word End
Number active	3539	866	265	91	43
Proportion active	65.4%	16.0%	4.9%	1.7%	0.8%
Relative computation	76.0%	18.6%	5.7%	1.9%	

It can be seen that 95% of the computation is in the first two phones.

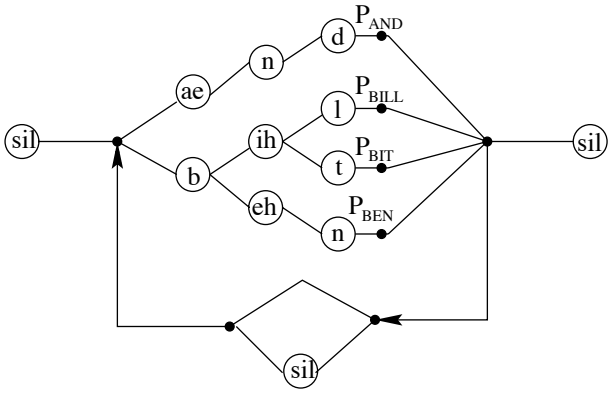
To make an efficient decoder it is important to reduce the computation at the *start* of words.

- tree-structure the decoding network
- words no longer have a unique start in the network

Tree Structured Lexicon

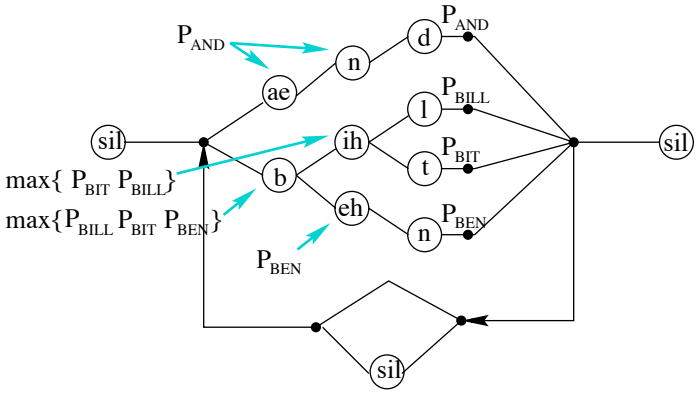
Problem with standard implementation

- At the end of each word there are typically many possible following words
- Most of these paths are rapidly pruned away.



Tree structuring the lexicon reduces search cost but

- the LM application is delayed until the end of the word (reduces effectiveness of pruning).
- may be overcome by early, approximate, LM application.



Decoding Strategies/Refinements

1. **Weighted Finite-State Transducers** Use a WFST-based approach to optimise the state-level network structure, and search this structure with the Viterbi algorithm.
2. **Dynamic Decoders** Generate only the parts of the search network as needed: rely on pruning for this to not get too large.
3. **Fast-match procedures** Look ahead a few frames and calculate an "approximate" match (e.g. for monophones). Historically this used different acoustic models but can be useful when using neural network based models as these monophone outputs can be computed from the context dependent DNN outputs. This can discard unlikely paths (don't allow paths to be extended).
4. **Stack-decoders** modify form of decoder to run in a *time asynchronous* fashion.
5. **Multi-pass search** - run over data more than once. For example:
   - Initially use word internal models/bigram.
   - Use these simple models to generate multiple hypotheses in the form of ***N*-best list** or **lattices**.
   - **Rescore** hypotheses (N-best/lattice) with more accurate acoustic e.g. cross-word models and longer span language models

   It is possible to run the second pass backwards to generate the final answers (and use a backwards dictionary, backwards data and a backwards LM)!

In addition to the above there are various combinations of the above strategies that have been investigated.

Dynamic Network Decoding

A full trigram network with cross-word triphones can become very large.

Rather than using a static network decoder, a **dynamic network** decoder may be used i.e. create new network nodes as required and rely on pruning of the search space to keep the number of active paths small.

This allows more complex acoustic and language models and still integrates all knowledge in a single pass.

Basic description:

- tree structure network where possible
- create new network nodes as needed
- expand network to ensure enough context for language models and acoustic models
- cross-word triphone models only slightly more expensive than word internal models;
- relies of rapid pruning of search space
- for efficient pruning early application of the language model is required.
- it is possible to construct a general purpose dynamic network decoder can use arbitrary N-gram language model and acoustic model constraints e.g. pentaphone (quinphone) HMMs with a 4-gram LM.

WFST Based Decoders

Aim is to expend most effort on optimising the state-level search network by representing the various elements as WFSTs, and then optimising the result to minimise the overall size.

The following are all represented as **weighted transducers**

- Hidden Markov Models, H
- Context Dependent models , C
- Lexicon (list of words/pronunciations), L
- N-gram grammar, G

Compose transducers

$$H \circ C \circ L \circ G$$

and determinise / minimise the size of the resulting network using standard operations.

There are a number of WFST toolkits that help with these operations, notably **OpenFST** (`www.openfst.org`)

Most new large vocabulary decoders are based on WFSTs (since can be very fast).

However there are various issues:

- Network construction complex (can need large amounts of memory).
- Addition of new words
- incorporation of large language models (normally via lattice rescoring)
- Generation of good word lattices

However with current high-accuracy ANN acoustic models it is possible to use a WFST with a weak language model and generate lattices which can be rapidly rescored (see below).

Generating/Using Multiple Solutions

Often required to produce **multiple hypotheses** rather than just the 1-Best output.
We will discuss how this can be done and some ways in which such output can be used.

Topics covered include:

- Lattice & word-dependent N-best
- Multiple token search
- Lattice Quality & Lattice Pruning
- Acoustic Model & LM Rescoring
- "N-Best paradigm" to integrate varied knowledge sources

In a multi-pass system we might want to use a more complex acoustic or language model to rescore multiple "good" paths, not just the best path (or what's the point!).

The multiple paths are either stored as

- **N-best list**
- **Word Lattice**

For most LVCSR systems lattices are preferred: give a far more compact representation. We will look at how these multiple hypotheses can be generated.

- Lattice N-best
- Sentence dependent N-best
- Word dependent N-best

Lattice *N*-Best

A simple modification to the basic token-passing Viterbi search that alters the `Record Decisions` function.

At each frame a WLR is created for the *N*-Best tokens entering a word instance in the network.

Within a word only the single best token is propagated, however this can cause search errors in finding *N*-Best alternatives.

```
for each entry state at each t do
   Select the best token Q
   Create a new WLR w containing:
      (1) a copy of Q
      (2) time t
      (3) identity of emitting word
   Q.link = w
   for 2nd to Nth best token, q, do
      Create a new WLR v containing:
         (1) a copy of q
         (2) time t
         identity of emitting word
      chain v to w;
   end;
end;
```

At the end of the sentence the trace-back procedure is used to convert the history information to a **word lattice**.

A word lattice is a finite state network structure containing

- acoustic and language model scores
- word identities and start/end times

This structure can then be used to find the *N*-Best hypotheses.

Sentence-Dependent & Word-Dependent *N*-Best

Sentence Dependent *N*-Best:

- Modify token passing approach is modified so that each state can hold **multiple tokens**.
- After propagating tokens merge tokens with same word history (keep best)
- On completion, *N*-best sentences are available from *N* tokens in the final state.
- Guaranteed to find best *N* paths through network.
- Computationally very expensive. Main problem is cost of identifying sets of tokens with identical histories (need to have *N* quite large, e.g. 100)

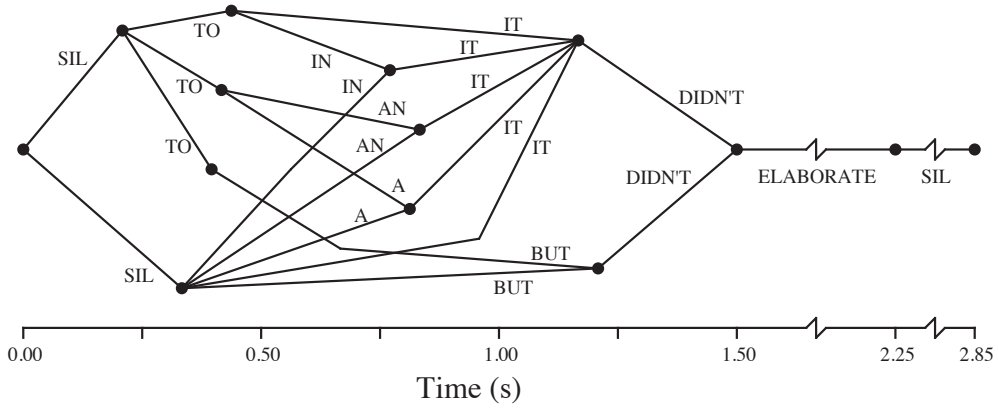
Word-Dependent *N*-Best

- Intermediate between lattice *N*-best & sentence dependent *N*-best (exact) algorithm.
- Uses the **word-pair** approximation: word start time depends only on the previous word
- Only store small number of tokens *n* in each state, after propagation, merge tokens with the same previous word & keep *n* best
- Typically *n* is 4–16
- Create WLRs in the same way as for lattice N-best
- At the end will be able to trace back and form large number of paths/lattices

Note that multiple token search can be used in a hybrid static/dynamic decoder with basic vocabulary / acoustic models in a tree-structured static network and multiple tokens used to allow N-gram LMs (e.g. HTK decoder `HDecode`)

Word Lattices

A typical word lattice structure is shown.



A general word lattice structure contains:

- A set of nodes that correspond to points in time (or word-ends)
- A set of arcs that are labelled with the word identifies and the acoustic scores of words between nodes and also the language model scores

Note that if acoustic score information is included then many arcs will be replicated due to slightly different acoustic context / timing information.

Lattice Accuracy & Lattice Pruning

The quality of the lattices can be measured by finding the **minimum possible error rate** of the lattices with respect to the actual sentence spoken.

- Most accurate path through the lattice is found using a best first dynamic programming search that minimises the total number of word errors.
- When the correct sentence exists in the lattice, it will be found otherwise the total number of insertion, deletion and substitution errors will be calculated.
- The number of word/sentence errors can be used as lattice performance figures.
- Often referred to as the **oracle** word / sentence error rates.
- If alternatives are stored as list then lowest error rate alternative gives N -Best error rate.

Lattices generated can be very large

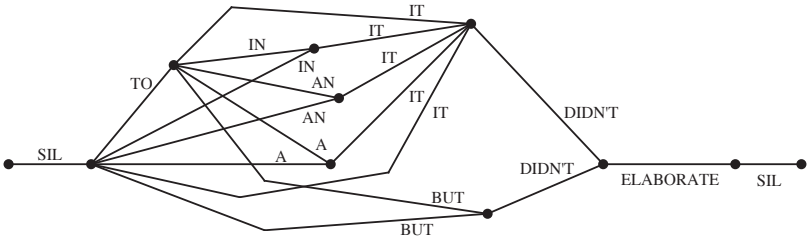
- depends on the beamwidths and the number of tokens per state
- some parts very unlikely to be important & can be **pruned**!

Lattice pruning

- Finds best possible path through the complete lattice and compares it to the most likely path through a lattice node/arc.
- Uses a lattice-level forward-backward algorithm
- Prunes based on arc/node posterior
- Very effective: significantly reduce size without damaging lattice accuracy too much.

Acoustic Model Rescoring of Lattices

- Use as finite-state grammar for acoustic rescoring with more complex acoustic models
  - e.g.Generated with word-internal context dependent models, rescore with x-wrd
  - Need only fairly likely word sequences along with LM scores
  - Don't need to keep different word times/contexts



- Word lattices can be used for multi-pass decoding
  - Can be useful in system development for rapid decoding with new acoustic models
  - Also can be used in discriminative sequence training of HMMs (see later lecture ...)

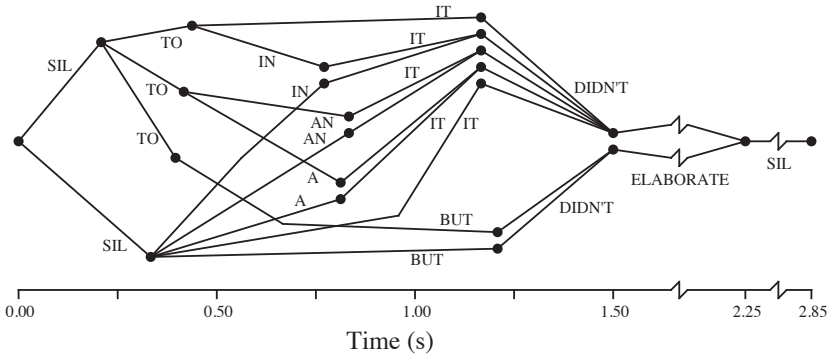
Language Model Rescoring

The basic lattices can also be used to evaluate the effect of **new language models** without further acoustic processing (using the acoustic scores from the lattice).

A very useful operation is **lattice expansion** with a new LM.

- For example a lattice generated with a bigram model could be expanded to a trigram lattice (& is then pruned)
- Can be used for acoustic rescoring or for finding the best path subject to the new LM.

An example trigram lattice generated from the previous bigram lattice is shown below.



An alternative (e.g. if entire sentence required for LM probability calculation) is to generate an *N*-Best list of sentences and then use each entry to find a score according to the new LM.

N-Best paradigm

N-Best rescoring is particularly useful for knowledge sources that are difficult to apply to a lattice directly.

General *N*-Best approach is

1. Generate *N*-Best list of alternatives (best-first search from lattice)
2. Rescore each alternative hypotheses with *K* different knowledge sources and generate a log likelihood L_k for each of the *N* alternatives
3. Re-rank the alternatives by using a weighted sum of the different knowledge source scores

Optimal weights (to maximise accuracy) of the knowledge sources can be found by using some development data and a grid-search to minimise the word error rate.

Powerful and general framework

- Very many different knowledge sources can be used that would be hard to integrate directly into the original search
  - Language models that require complete hypotheses
  - Other types of information (e.g.relative duration, prosodic scores)
- However it requires a high-quality *N*-best list to start.

Confusion Network Decoding

Normally speech is recognised to find the most likely sentence

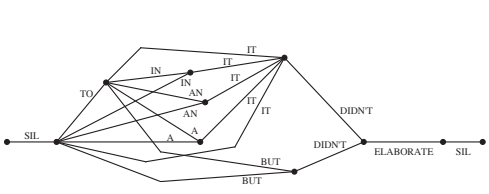
$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \{P(\mathbf{W}|\boldsymbol{O};\lambda)\}$$

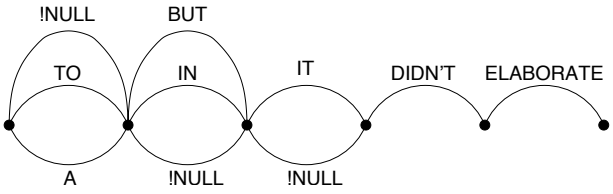
This will minimise the **sentence error rate**. In the Viterbi algorithm we approximate the word sequence by that of the best state sequence.

If confusions can be made explicit at the word-level (each with a posterior probability) then could use the following:

$$\hat{\mathbf{W}} = \sum_{i=1}^{L} \arg\max_{W_i} \{P(W_i)|\boldsymbol{O};\lambda)\}$$

this should minimise the WER rather than sentence error rate.





Confusion networks (also known as “sausages”) are one approach to this

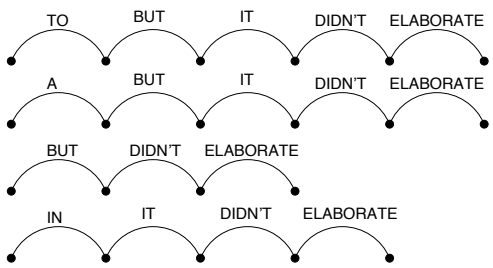
- use standard HMM decoder to generate word lattice;
- iteratively merge links to form confusion networks (CN) from word lattice.

System (Hypothesis) Combination

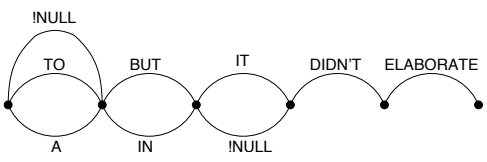
- Can improve the performance by combining multiple system outputs
- A standard machine learning is to use an **ensemble** of classifiers: in general with different strengths / weaknesses
- LVCSR systems can include "system combination" e.g.
  - different pronunciation dictionaries / phone sets incl. graphemic models
  - different acoustic model types e.g. DNN hybrid vs Tandem, various types of DNN
  - completely different systems / implementations
- Combination can be explicit: use **ROVER or CNC** (see below)
- Combination can be at different levels (e.g. word, phone, frame)
- **Frame-level combination**: combine acoustic scores from different models
  - **joint decoding** with multiple acoustic models
  - easier if use same decision trees (same state distributions)
  - may need to weight combination
  - can lead to more more effective pruning
- Implicit combination via **cross-adaptation**: use output from one system to adapt another

ROVER

- Introduced by NIST (Fiscus, 1997) and was then widely used
- ROVER takes the output from multiple recognition systems (1-best, optionally with confidence scores) then:
  - convert outputs into **Word Transition Networks** (WTNs)
  - align and combine (WTNs) in a pre-specified order
  - can use both **confidence scores** and **voting** to choose the best output for each word position from the aligned hypotheses
  - if (only) use confidence scores then this is a confusion network!
- A simple example output: `BUT IT DIDN'T ELABORATE`



Multiple System WTNs



Aligned/Combined

Confusion Network Combination (CNC)

In contrast to ROVER, align and combine CNs

- use all word posteriors from systems rather than using the 1-best only
- combines the posterior estimates from all input confusion networks
- combined "posterior" found by

$$P(W_i|\boldsymbol{O};\lambda^{(1)},\ldots,\lambda^{(S)}) = \sum_{s=1}^{S} P(s)P(W_i|\boldsymbol{O};\lambda^{(s)})$$

  $P(s)$ can be used to represent the global confidence in system s
- New combined confusion network: pick best posterior as usual in CN decoding

CNC generally works slightly better than ROVER

- system word posteriors, rather than 1-best helps
- **but** alignment more complex:
  - difficult to use with multiple segmentations of acoustic data (automatic segmentation)

Note that an alternative to CN/CNC decoding is to do direct Minimum Bayes' Risk (MBR) decoding on word lattices which can also be used for combination from parallel lattices.

- usually produces similar WER results to CNC.

Summary

- Viterbi search can be used for a variety of model types for continuous speech
- Can integrate all the models into a single search pass but
  - Network can get (much) more complex for different acoustic and language models
  - Network can be optimised using weighted finite state transducers
  - An alternative dynamically generates network as needed (within pruning beam).
- Generate multiple solutions rather than 1-best
  - store multiple tokens in each state
  - word-pair or language-model state approximation widely used
  - N-best or lattices with simpler acoustic/language models generated
- Rather than single-pass decoding, can use multi-pass search
  - Generate N-best or lattices with simpler acoustic/language models
  - Rescore lattices with more variety of complex models
  - Can also include unsupervised adaptation (future lecture ...)
  - Often used transcription systems (esp. where latency is less important)
  - Lattices are widely used in various ways in speech recognition systems
- Lattices can be further processed into e.g. confusion networks
  - Perform posterior decoding
  - Can be used when combining an ensemble of outputs: "system combination"