Decoding

Brute force enumeration: by number of phrase-pairs
Enumeration by $o$-coverage

'Decoding' means trying to solve $\arg \max_{\bar{s}, \tau} p(\bar{O}, \tau, \bar{s})$, which is

\[
\arg \max_{\bar{s}, \tau} \left[ \prod_{k=1}^{K} tr(\bar{O}_{\tau(k)}|\bar{s}_k) d(\bar{O}_{\tau(k-1)}, \bar{O}_{\tau(k)}) \right] LM(s)
\]

- the decoding constructs $\bar{s}$ in sequence $\rightarrow$ need the score for a complete hypothesis $(\tau, \bar{s})$ to computable from scores for \textit{partial} hypotheses $(\tau_1, \bar{s}_{1:k})$
- we still need to 'fold in' the \textit{LM} contribution, but that is easily done. Let $s_{1:let(\bar{s}_k)}$ be the source sequence underlying the phrase sequence $\bar{s}_{1:k}$, and $\Phi_{LM}(\bar{s}_k)$ be the parts of the $LM(s_{1:let(\bar{s}_k)})$ mentioning the words in $\bar{s}_k$. This is

\[
\Phi_{LM}(\bar{s}_k) = \prod_{i=let(\bar{s}_k)}^{i=let(\bar{s}_k)} p(s_i|s_{i-(n-1)}, \ldots, s_{i-1})
\]

- the fully incremental equation is then

\[
\arg \max_{\bar{s}, \tau} \left[ \prod_{k=1}^{K} tr(\bar{O}_{\tau(k)}|\bar{s}_k) d(\bar{O}_{\tau(k-1)}, \bar{O}_{\tau(k)}) \Phi_{LM}(\bar{s}_k) \right]
\]
The Translation Options from the phrase-table

For er geht ja nicht nach hause the above depicts a subset of the relevant phrase pairs (\(\vec{o}, \vec{s}\))
- in Europarl phrase table: 2727 matching phrase pairs for this sentence
- by pruning to the top 20 per phrase, 202 translation options remain
- picture shows top 4 for each \(\vec{o}\) German ‘phrase’

Exhaustive enumeration

- though not realistic, its useful to first consider what would be involved in exhaustively enumerating all the possibilities, exploiting the incremental, LtoR nature of the defining formula

Exhaustive enumeration cntd

- initial hypothesis: no observed words used up, no source words produced; as observed words are used up will black out corresponding box in strip at top of box; will write in the box the chosen source phrase

Translation Options

- The correct translation chooses the highlighted phrase pairs above, and chooses an ordering
- has to somehow do this guided by the fact that \(s_{post}\) is defined as the source part of

\[
\arg\max_{\vec{s}} \prod_{k=1}^{K} \phi(\vec{o}(\vec{s}_{k}); \vec{s}_{k}) d(\vec{o}(\vec{s}_{k-1}), \vec{o}(\vec{s}_{k})) \Phi_{LM}(\vec{s}_{k})
\]
Exhaustive enumeration contd

- put in all other hypotheses for first source phrase and matched observed phrase (only a few are shown)
- NOTE: all these partial hypotheses can be assigned a score as you go

Exhaustive enumeration contd

- from all the hypotheses concerning 1st source phrase, make all hypotheses about 2nd source phrase, then for 3rd source phrase etc.
- in each case new score easily obtained using score of ancestor hypothesis
- as you go forward, for each hypothesis the available possible phrase pairs which could be used is going down as observed words are struck off. So the process is going to end.

Computational Complexity

- however, this outlined procedure is **exponentially costly**; for example if there are 2000 relevant entries from the translation tables for \( o = \text{er geht ja nicht nach hause} \), then there will already be approx 4,000,000 hypotheses for the first 2 phrases of the source.
- can actually prove that there is no non-exponential algorithm which is guaranteed to find the best translation as defined by formula (??) : 'Machine translation decoding is NP-complete'
- people use **pruning** in some fashion to kill the vast majority of hypotheses early on
- inherently risky as however you do it, you may prune something which looks insufficiently promising though it would recover as you go further
A different enumeration: by o-coverage

- in the above enumeration the hypotheses grew in stages, from all hypotheses about \( \bar{s}_{1,k} \) (the first \( k \) source phrases) to all hypotheses about \( \bar{s}_{1,k+1} \) (the first \( k + 1 \) source phrases). The simplest kind of pruning would set some max \( h_{\text{max}} \) and at each \( k \) keep only the best \( h_{\text{max}} \) hypotheses.
- this does not work very well though because the hypotheses concern wildly different numbers of source words and observed words, and so are not really comparable.
- an alternative is to change the book-keeping and progression a little so that all hypotheses that share the same amount \( L \) of coverage of observed words are grouped – say they are stored in \( \text{bin}_L \).

**generate:** if you have found all (or \( h_{\text{max}} \) best) hypotheses covering \( L \) observed words – \( \text{bin}_L \) – then can generate all successor hypotheses, adding each to relevant later \( \text{bin}_{L+x} \), where \( x \) is length of the chosen \( \bar{o} \) phrase of the successor hypothesis, and then move on to \( \text{bin}_{L+1} \).

**pruning:** can keep the size of such \( L \)-organised bins down by pruning, and in so doing be ranking more comparable things.

***er geht nach haus*** example

To see this evolution with bins organised by o-coverage, suppose for ***er geht nach haus*** we have the following minute phrase table:

<table>
<thead>
<tr>
<th></th>
<th>er</th>
<th>geht</th>
<th>nach</th>
<th>haus</th>
</tr>
</thead>
<tbody>
<tr>
<td>he</td>
<td>goes</td>
<td>home</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Also without giving all the details of language model lets suppose the following source bigrams are impossible:

- he home  
- goes he  
- home goes

Suppose for ***er geht nach haus*** all bins initialised with single phrase possibilities (green above), and \( \text{bin}_1 \) is to be extended.

- above shows 2 successor hypotheses to the \( \langle \bar{s}_1 = \text{he}, \tau(1) = \text{er} \rangle \) hypothesis one adds to \( \text{bin}_1 \): \( \langle \bar{s}_2 = \text{goes}, \tau(2) = \text{geht} \rangle \) one adds to \( \text{bin}_2 \): \( \langle \bar{s}_3 = \text{does not}, \tau(2) = \text{ja nicht} \rangle \)
- Once all \( \text{bin}_1 \) hypotheses have been extended, move to \( \text{bin}_2 \)
- general fact: a new hypothesis is dropped into a bin further down.

***er geht nach haus***: initial stage

The empty hypothesis is extended via possibilities in phrase table: creates some hypotheses in \( \text{bin}_1 \) and \( \text{bin}_2 \).
**er geht nach haus: extend bin₁**

Hypotheses in bin₁ now extended; creates additions to bin₂ and bin₃
impossible bigrams rule out
extension of he with home
extension of goes with he

---

**er geht nach haus: extend bin₂**

Hypotheses in bin₂ now extended; creates additions to bin₃ and bin₄
impossible bigrams rule out
extension of home with goes

---

**er geht nach haus: extend bin₃**

Hypotheses in bin₃ now extended; creates additions to bin₄
Tracing back from bin₄ there are 3 possible translations
he goes home; goes home he; home he goes
We did not include the scores from the model; these translations would be
distinguished by the source lang. factor LM and by the distortion factors,
favouring he goes home

to get a further feel for what this alternative enumeration of the hypothesis
space via amount of o-coverage looks like the following pictures² depict
schematically an evolution of contents of bins; for the moment there is no
pruning
to keep things compact just showing schematically the observed words
consumed and not the associated chosen source phrase, nor the pointers back
to the ancestor hypothesis

the example comes from a Chinese/English case. Though for the schematic
pictures the actual translation options are not important, for completeness they
are:

```
Bush and held a talk
bushi yu shalong juxing le huitan
Bush and held meeting
```

²From Watanabe
empty hypothesis, no observed words consumed

extend the empty bin₀ hypothesis, adding to bin₁ and later...

bin₁ is complete

extend bin₁ hypotheses, adding to bin₂ and later...
**bin₂ is complete**

**extend bin₂ hypotheses, adding to bin₃ and later...**

**bin₃ is complete**

**extend bin₃ hypotheses, adding to bin₄ and later...**
Beam search algorithm

With hypotheses stored in bins associated with number of $o$ words used, have a better basis for pruning and leads to the following simple so-called 'Beam search' version

1. Place empty hypothesis into bin$_0$
2. For $L = 0$ to length($o$) - 1 {
   a. For each hypotheses in bin$_L$ {
      i. For each applicable pair $\bar{s}, \bar{o}$ {
         1. Create new hypothesis (including score)
         2. Place in bin$_{L+x}$ where $\bar{o}$ has length $x$
         3. Prune bin$_{L+x}$ if too big
      }
   }
}

why beam?

- 'beam': idea is that across the bins, which really should be exponentially growing, a fixed width 'beam' of good hypotheses is illuminated (ie. kept)
- It's a heuristic and while it strives to have the $h_{max}$ options stored in bin$_L$ be genuinely the $h_{max}$ best hypotheses that use $L$ observed words, it will not do this perfectly
- The width of the 'beam' – $h_{max}$ – is set as high as is computationally feasible and at the very end the best is chosen

Following pictures show again an evolution of bins, this time including pruning...
extend bin₀ hypotheses, adding to bin₁, ...
prune bin₂

extend bin₂ hypotheses, adding to bin₃, ...

prune bin₃

extend bin₃ hypotheses, adding to bin₄, ...
4CSLL5 “Advanced Computational Linguistics” Phrase Based Machine Trans
Decoding
Enumeration by n-coverage