Lexical Translation

- How to translate a word → look up in dictionary
  - **Haus** — house, building, home, household, shell.

- Multiple translations
  - some more frequent than others
  - for instance: house, and building most common
  - special cases: Haus of a snail is its shell
Collect Statistics

- Suppose a parallel corpus, with German sentences paired with English sentences, and suppose people inspect this marking how Haus is translated.
  
  
  
  
  
  
  das Haus ist klein  the house is small
  
- Hypothetical table of frequencies

<table>
<thead>
<tr>
<th>Translation of Haus</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>house</td>
<td>8,000</td>
</tr>
<tr>
<td>building</td>
<td>1,600</td>
</tr>
<tr>
<td>home</td>
<td>200</td>
</tr>
<tr>
<td>household</td>
<td>150</td>
</tr>
<tr>
<td>shell</td>
<td>50</td>
</tr>
</tbody>
</table>

IBM models

- the so-called IBM models seek a probabilistic model of translation one of whose ingredients is this kind of lexical translation probability.
- there’s a sequence of models of increasing complexity (models 1-5). The simplest models pretty much just use lexical translation probability
- parallel corpora are used (eg. pairing German sentences with English sentences) but crucially there is no human inspection to find how given German words are translated to English words, ie. info is of form
  
  
  
  
  das Haus ist klein  the house is small
  
- though originally developed as models of translation, these models are now used as models of alignment, providing crucial training input for so-called ‘phrase-based SMT’

Estimation of Translation Probabilities

- from this could use relative frequencies as estimate of translation probabilities \( t(e|Haus) \)
- technically this is a maximum likelihood estimate – there could be others
- outcome would be

\[
tr(e|Haus) = \begin{cases} 
  0.8 & \text{if } e = \text{house}, \\
  0.16 & \text{if } e = \text{building}, \\
  0.02 & \text{if } e = \text{home}, \\
  0.015 & \text{if } e = \text{household}, \\
  0.005 & \text{if } e = \text{shell}. 
\end{cases}
\]

Notation

- For reasons that will become apparent, we will use \( \mathcal{O} \) for the language we want to translate from \( \mathcal{S} \) for the language we want to translate to
- \( o \) is a single sentence from \( \mathcal{O} \), and is a sequence \( (o_1 \ldots o_j \ldots o_\ell_o) \); \( \ell_o \) is length \( o \)
- \( s \) is a single sentence from \( \mathcal{S} \), and is a sequence \( (s_1 \ldots s_i \ldots s_\ell_s) \); \( \ell_s \) is length \( s \)
- the set of all possible words of language \( \mathcal{O} \) is \( \mathcal{V}_o \)
- the set of all possible words of language \( \mathcal{S} \) is \( \mathcal{V}_s \)
- comments on notation in Koehn, J&M
The sparsity problem

- Suppose for two languages you have large sentence-aligned corpus \( d \). Say the two languages are \( O \) and \( S \).
- in principle for any sentence \( o \in O \) could work out the probabilities of its various translations \( s \) by relative frequency

\[
p(s|o) = \frac{\text{count}(\langle o, s \rangle \in d)}{\sum_{s'} \text{count}(\langle o, s' \rangle \in d)}
\]

- but even in very large corpora the vast majority of possible \( o \) and \( s \) occur zero times. So this method gives uselessly bad estimates.

The Noisy-Channel formulation

- recalling Bayesian classification, finding \( s \) from \( o \):

\[
\begin{align*}
\text{arg max}_s P(s|o) &= \text{arg max}_s \frac{P(s, o)}{P(o)} \\
&= \text{arg max}_s P(s, o) \\
&= \text{arg max}_s P(o|s) \times P(s)
\end{align*}
\]

- can then try to factorise \( P(o|s) \) and \( P(s) \) into clever combination of other probability distributions (not sparse, learnable, allowing solution of arg-max problem). IBM models 1-5 can be used for \( P(o|s) \); \( P(s) \) is the topic of so-called ‘language models’.
- The reason for the notation \( s \) and \( o \) is that (3) is the defining equation of Shannons ‘noisy-channel’ formulation of decoding, where an original ‘source’ \( s \) has to be recovered from a noisy observed signal \( o \), the noisiness defined by \( P(o|s) \)

Alignments (informally)

- When \( s \) and \( o \) are translations of each other, usually can say which pieces of \( s \) and \( o \) are translations of each other. eg.

```
1 2 3 4
das Haus ist klein
1 2 3 4
the house is small
```

- In SMT such a piece-wise correspondence is called an alignment
- warning: there are quite a lot of varying formal definitions of alignment
Hidden Alignment

- key feature of the IBM models is to assume there is a hidden alignment, a between o and s
- so a pair \( \langle o, s \rangle \) from a sentence-aligned corpus is seen as a partial version of the fully observed case:

\[
\langle o, a, s \rangle
\]

- A model is essentially made of \( p(o, a|s) \), and having this allows other things to be defined
- best translation:

\[
\arg\max_s P(s, o) = \arg\max_s (\sum_a p(o, a|s)) \times p(s)
\]

- best alignment:

\[
\arg\max_a [p(o, a|s)]
\]

Some weirdness about directions

- Note here o is English, and s is German
- the alignment goes up the page, English-to-German,
- they will be used though in a model of \( P(o|s) \), so down the page, German-to-English

IBM Alignments

- Define alignment with a function, from posn \( j \) in o to posn. \( i \) in s so \( a: j \rightarrow i \)
- the picture

\[
\begin{array}{cccc}
\text{das} & \text{Haus} & \text{ist} & \text{klein} \\
1 & | & | & |
\end{array}
\]

represents

\[
a: \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4\}
\]

Comparison to 'edit distance' alignments

- in case you have ever studied 'edit distance' alignments . . .
- like edit-dist alignments, its a function: so can’t align 1 o words with 2 s words
- like edit-dist alignments, some s words can be unmapped to (cf. insertions)
- like edit-dist alignments, some o words can be mapped to nothing (cf. deletions)
- unlike edit-dist alignments, order not preserved: so \( j < j' \), \( a(j) \neq a(j') \)
N-to-1 Alignment (ie. 1-to-N Translation)

- \( a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\} \)
- \( N \) words of \( o \) can be aligned to 1 word of \( s \)
  (needed when 1 word of \( s \) translates into \( N \) words of \( o \))

Reordering

- \( a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\} \)
- alignment does not preserve \( o \) word order
  (needed when \( s \) words reordered during translation)

s words not mapped to (ie. dropped in translation)

- \( a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 5\} \)
- some \( s \) words are not mapped-to by the alignment
  (needed when \( s \) words are dropped during translation)
  (here the German flavouring particle ‘ja’ is dropped)

o words mapped to nothing (ie. inserting in translation)

- \( a : \{1 \rightarrow 1, 2 \rightarrow 0, 3 \rightarrow 3, 4 \rightarrow 2, 5 \rightarrow 4, 6 \rightarrow 4, 7 \rightarrow 5\} \)
- some \( o \) word are mapped to nothing by the alignment
  (needed when \( o \) words have no clear origin during translation)
  The is no clear origin in German of the English ‘do’
  formally represented by alignment to special NULL token
IBM Model 1

- basically a hidden variable $a$, aligning $o$ to $s$ is assumed.
- in more detail, IBM model 1 will define a probability model of
  
  \[ P(o, a, L, s) \]
  
  where $L$ is length for $o$ sentences, and $a$ is an alignment from $o$ sentences of length $L$ to $s$.
- $o$, $a$, $L$ are intended to be synchronized in the sense that if $L$ is not the $\ell_o$ the probability is zero. Similarly if $a$ is not an alignment function from length $L$ sequences to length $\ell_s$ sequences, the probability is 0. So we will write
  
  \[ P(o, a, \ell_o, s) \]

Length dependency

- first without any assumptions, via the chain rule:
  \[ P(o, a, \ell_o, s) = P(o, a, \ell_o | s) \times P(s) \]
  
  the IBM model1 assumptions are all about $P(o, a, \ell_o | s)$. The assumptions can be shown by a succession of applications of the chain rule concerning $(o, a, \ell_o)$
- concerning $\ell_o$, still without any particular assumptions
  \[ P(o, a, \ell_o | s) = P(o, a | \ell_o, s) \times p(\ell_o | s) \]
  
  An assumption of IBM Model 1 is that the dependency $p(\ell_o | s)$ can be expressed as a dependency just on the length $\ell_s$, so by some distribution $p(L|\ell_s)$.
- Usually it's stated that $p(L|\ell_s)$ is uniform: ie. all $L$ equally likely
- We will see in a while that for many of the vital calculations for training the model, the actually values of $p(L|\ell_s)$ are irrelevant.

Alignment dependency

- we have so far
  
  \[ P(o, a, \ell_o | s) = P(o, a | \ell_o, s) \times p(\ell_o | \ell_s) \]

- analysing $P(o, a | \ell_o, s)$, a further application of the chain rule gives
  
  \[ P(o, a | \ell_o, s) = P(o | a, \ell_o, s) \times P(a | \ell_o, s) \]

  (4)

- The next assumption is that the dependency $P(a | \ell_o, s)$ can be expressed as dependency just on $\ell_s$ and $\ell_o$, and furthermore that the distribution of possible alignments from length $\ell_o$ sequences to length $\ell_s$ sequences is a uniform distribution
- There are $\ell_o$ members of $o$ to be aligned, and for each there are $\ell_s + 1$ possibilities (including NULL mappings), so there are $(\ell_s + 1)^{\ell_o}$ possible alignments, so this means
  
  \[ p(a | \ell_o, \ell_s) = \frac{1}{(\ell_s + 1)^{\ell_o}} \]

Observed words dependency

- this means the formula for $P(o, a | \ell_o, s)$ from (4) now looks like this
  \[ P(o, a | \ell_o, s) = P(o | a, \ell_o, s) \times \frac{1}{(\ell_s + 1)^{\ell_o}} \]  
  (5)

- finally concerning $P(o | a, \ell_o, s)$ it is assumed that this probability takes a particularly simple multiplicative form, with each $o_j$ treated as independent of everything else given the word in $s$ that it is aligned to, that is, $s_{a(j)}$, so
  
  \[ p(o | a, \ell_o, s) = \prod_j [p(o_j | s_{a(j)})] \]

- and $P(o, a | \ell_o, s)$ becomes
  \[ P(o, a | \ell_o, s) = \prod_j [p(o_j | s_{a(j)})] \times \frac{1}{(\ell_s + 1)^{\ell_o}} \]  
  (6)
The final IBM Model 1 formula

\[ P(o, a, \ell_o|s) = \prod_j [p(o_j|s_{a(j)})] \times \frac{1}{(\ell_s + 1)^{\ell_o}} \times p(\ell_o|\ell_s) \]

or slightly more compactly

\[ P(o, a, \ell_o|s) = \frac{p(\ell_o|\ell_s)}{(\ell_s + 1)^{\ell_o}} \times \prod_j [p(o_j|s_{a(j)})] \quad (7) \]

Another way to arrive at the formula is via the following so-called 'generative story' for generating \( o \) from \( s \):

1. Choose a length \( \ell_o \), according to a distribution \( p(\ell_o|\ell_s) \)
2. Choose an alignment \( a \) from \( 1 \ldots \ell_o \rightarrow 0, 1 \ldots \ell_s \), according to distribution \( p(a|\ell_s, \ell_o) = \frac{1}{(\ell_s+1)^{\ell_o}} \)
3. For \( j = 1 \) to \( j = \ell_o \), choose \( o_j \) according to distribution \( p(o_j|s_{a(j)}) \)

Example

Suppose \( s \) is \( das \ haus \ ist \ klein \) and \( o \) is \( the \ house \ is \ small \). Recall the alignment from \( o \) to \( s \) shown earlier:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\text{das} & \text{Haus} & \text{ist} & \text{klein} \\
1 & 2 & 3 & 4 \\
\text{the} & \text{house} & \text{is} & \text{small}
\end{array}
\]

we will illustrate the value of \( p(o, a, \ell_o|s) \) in this case, according to the formula (7)

\[
P(o, a, \ell_o|s) = \frac{p(\ell_o|\ell_s)}{(\ell_s + 1)^{\ell_o}} \times \prod_j [p(o_j|s_{a(j)})]
\]

Example cntd

Let \( \epsilon \) represent the \( P(\ell_o = 4|\ell_s = 4) \) term

\[
p(o, a, \ell_o|s) = \epsilon \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein})
\]

\[
= \frac{\epsilon}{5^4} \times 0.7 \times 0.8 \times 0.8 \times 0.4
\]

\[
= 0.00028672 \epsilon
\]

\(^1\text{see p87 Koehn book}\)