Alignment

- In a parallel text (or when we translate), we align words in one language with the words in the other

1. das Haus ist klein
2. the house is small

- Word positions are numbered 1–4
Alignment Function

- Formalizing alignment with an alignment function

- Mapping an English target word at position $i$ to a German source word at position $j$ with a function $a : i \rightarrow j$

- Example

  $a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4\}$
Reordering

Words may be reordered during translation

\[ a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\} \]
## One-to-Many Translation

A source word may translate into multiple target words

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;das Haus ist klitzeklein&quot;</td>
<td>&quot;the house is very small&quot;</td>
</tr>
</tbody>
</table>

\[
a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\}
\]
Dropping Words

Words may be dropped when translated
(German article *das* is dropped)

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

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

• Words may be added during translation
  – The English just does not have an equivalent in German
  – We still need to map it to something: special NULL token

$$a: \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 0, 5 \rightarrow 4\}$$
IBM Model 1

- Generative model: break up translation process into smaller steps
  - IBM Model 1 only uses lexical translation

- Translation probability
  - for a foreign sentence $\mathbf{f} = (f_1, \ldots, f_{l_f})$ of length $l_f$
  - to an English sentence $\mathbf{e} = (e_1, \ldots, e_{l_e})$ of length $l_e$
  - with an alignment of each English word $e_j$ to a foreign word $f_i$ according to the alignment function $a : j \rightarrow i$

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

- parameter $\epsilon$ is a normalization constant
Example

<table>
<thead>
<tr>
<th>das</th>
<th>Haus</th>
<th>ist</th>
<th>klein</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e$</td>
<td>$t(e</td>
<td>f)$</td>
<td>$e$</td>
</tr>
<tr>
<td>the</td>
<td>0.7</td>
<td>house</td>
<td>0.8</td>
</tr>
<tr>
<td>that</td>
<td>0.15</td>
<td>building</td>
<td>0.16</td>
</tr>
<tr>
<td>which</td>
<td>0.075</td>
<td>home</td>
<td>0.02</td>
</tr>
<tr>
<td>who</td>
<td>0.05</td>
<td>household</td>
<td>0.015</td>
</tr>
<tr>
<td>this</td>
<td>0.025</td>
<td>shell</td>
<td>0.005</td>
</tr>
</tbody>
</table>

\[
p(e, a|f) = \frac{\epsilon}{4^3} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein})
\]

\[
= \frac{\epsilon}{4^3} \times 0.7 \times 0.8 \times 0.8 \times 0.4
\]

\[
= 0.0028\epsilon
\]
em algorithm
EM Algorithm

• Incomplete data
  – if we had complete data, would could estimate model
  – if we had model, we could fill in the gaps in the data

• Expectation Maximization (EM) in a nutshell
  1. initialize model parameters (e.g. uniform)
  2. assign probabilities to the missing data
  3. estimate model parameters from completed data
  4. iterate steps 2–3 until convergence
EM Algorithm

... la maison ... la maison blue ... la fleur ...

... the house ... the blue house ... the flower ...

- Initial step: all alignments equally likely

- Model learns that, e.g., la is often aligned with the
EM Algorithm

... la maison ... la maison blue ... la fleur ...

... the house ... the blue house ... the flower ...

• After one iteration

• Alignments, e.g., between la and the are more likely
• After another iteration

• It becomes apparent that alignments, e.g., between *fleur* and *flower* are more likely (pigeon hole principle)
EM Algorithm

... la maison ... la maison bleu ... la fleur ...

... the house ... the blue house ... the flower ...

• Convergence

• Inherent hidden structure revealed by EM
EM Algorithm

... la maison ... la maison bleu ... la fleur ...

/ | X | |

... the house ... the blue house ... the flower ...

\[ p(\text{la}|\text{the}) = 0.453 \]
\[ p(\text{le}|\text{the}) = 0.334 \]
\[ p(\text{maison}|\text{house}) = 0.876 \]
\[ p(\text{bleu}|\text{blue}) = 0.563 \]

- Parameter estimation from the aligned corpus
IBM Model 1 and EM

- EM Algorithm consists of two steps

- Expectation-Step: Apply model to the data
  - parts of the model are hidden (here: alignments)
  - using the model, assign probabilities to possible values

- Maximization-Step: Estimate model from data
  - take assign values as fact
  - collect counts (weighted by probabilities)
  - estimate model from counts

- Iterate these steps until convergence
We need to be able to compute:

- Expectation-Step: probability of alignments
- Maximization-Step: count collection
IBM Model 1 and EM

• Probabilities

\[ p(\text{the}|\text{la}) = 0.7 \quad p(\text{house}|\text{la}) = 0.05 \]
\[ p(\text{the}|\text{maison}) = 0.1 \quad p(\text{house}|\text{maison}) = 0.8 \]

• Alignments

\[ \text{la} \rightarrow \text{the} \quad \text{la} \rightarrow \text{the} \quad \text{la} \rightarrow \text{the} \quad \text{la} \rightarrow \text{the} \]
\[ \text{maison} \rightarrow \text{house} \quad \text{maison} \rightarrow \text{house} \quad \text{maison} \rightarrow \text{house} \quad \text{maison} \rightarrow \text{house} \]

\[ p(\text{e}, \text{a}|\text{f}) = 0.56 \quad p(\text{e}, \text{a}|\text{f}) = 0.035 \quad p(\text{e}, \text{a}|\text{f}) = 0.08 \quad p(\text{e}, \text{a}|\text{f}) = 0.005 \]
\[ p(\text{a}|\text{e}, \text{f}) = 0.824 \quad p(\text{a}|\text{e}, \text{f}) = 0.052 \quad p(\text{a}|\text{e}, \text{f}) = 0.118 \quad p(\text{a}|\text{e}, \text{f}) = 0.007 \]

• Counts

\[ c(\text{the}|\text{la}) = 0.824 + 0.052 \]
\[ c(\text{house}|\text{la}) = 0.052 + 0.007 \]
\[ c(\text{the}|\text{maison}) = 0.118 + 0.007 \]
\[ c(\text{house}|\text{maison}) = 0.824 + 0.118 \]
• We need to compute $p(a|e, f)$

• Applying the chain rule:

$$p(a|e, f) = \frac{p(e, a|f)}{p(e|f)}$$

• We already have the formula for $p(e, a|f)$ (definition of Model 1)
We need to compute $p(\mathbf{e}|\mathbf{f})$

$$
p(\mathbf{e}|\mathbf{f}) = \sum_a p(\mathbf{e}, a|\mathbf{f})
$$

$$
= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} p(\mathbf{e}, a|\mathbf{f})
$$

$$
= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f + 1)^l_e} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})
$$
IBM Model 1 and EM: Expectation Step

\[
p(e|f) = \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})
\]

\[
= \frac{\epsilon}{(l_f + 1)^{l_e}} \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})
\]

\[
= \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)
\]

- Note the trick in the last line
  - removes the need for an exponential number of products
  → this makes IBM Model 1 estimation tractable
The Trick

(cases \( l_e = l_f = 2 \))

\[
\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} \frac{\epsilon}{3^2} \prod_{j=1}^{2} t(e_j|f_{a(j)}) = \\
= t(e_1|f_0) t(e_2|f_0) + t(e_1|f_0) t(e_2|f_1) + t(e_1|f_0) t(e_2|f_2) + \\
+ t(e_1|f_1) t(e_2|f_0) + t(e_1|f_1) t(e_2|f_1) + t(e_1|f_1) t(e_2|f_2) + \\
+ t(e_1|f_2) t(e_2|f_0) + t(e_1|f_2) t(e_2|f_1) + t(e_1|f_2) t(e_2|f_2) = \\
= t(e_1|f_0) (t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) + \\
+ t(e_1|f_1) (t(e_2|f_1) + t(e_2|f_1) + t(e_2|f_2)) + \\
+ t(e_1|f_2) (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) = \\
= (t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2)) (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2))
\]
IBM Model 1 and EM: Expectation Step

- Combine what we have:

\[
p(a|e, f) = \frac{p(e, a|f)}{p(e|f)}
\]

\[
= \frac{\epsilon}{(l_f+1)l_e} \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \\
\prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)}
\]

\[
= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)}
\]
Now we have to collect counts

Evidence from a sentence pair $e, f$ that word $e$ is a translation of word $f$:

$$c(e|f; e, f) = \sum_a p(a|e, f) \sum_{j=1}^{l_e} \delta(e, e_j) \delta(f, f_{a(j)})$$

With the same simplification as before:

$$c(e|f; e, f) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_f} \delta(f, f_i)$$
After collecting these counts over a corpus, we can estimate the model:

\[
t(e|f; e, f) = \frac{\sum_{(e,f)} c(e|f; e, f))}{\sum_e \sum_{(e,f)} c(e|f; e, f))}
\]
IBM Model 1 and EM: Pseudocode

**Input:** set of sentence pairs \((e, f)\)

**Output:** translation prob. \(t(e|f)\)

1: initialize \(t(e|f)\) uniformly

2: while not converged do

3: // initialize

4: count\((e|f)\) = 0 for all \(e, f\)

5: total\((f)\) = 0 for all \(f\)

6: for all sentence pairs \((e,f)\) do

7: // compute normalization

8: for all words \(e\) in \(e\) do

9: s-total\((e)\) = 0

10: for all words \(f\) in \(f\) do

11: s-total\((e)\) += \(t(e|f)\)

12: end for

13: end for

14: // collect counts

15: for all words \(e\) in \(e\) do

16: for all words \(f\) in \(f\) do

17: count\((e|f)\) += \(\frac{t(e|f)}{s\text{-}total(e)}\)

18: total\((f)\) += \(\frac{t(e|f)}{s\text{-}total(e)}\)

19: end for

20: end for

21: end for

22: // estimate probabilities

23: for all foreign words \(f\) do

24: for all English words \(e\) do

25: \(t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}\)

26: end for

27: end for

28: end while
## Convergence

<table>
<thead>
<tr>
<th>$e$</th>
<th>$f$</th>
<th>initial</th>
<th>1st it.</th>
<th>2nd it.</th>
<th>3rd it.</th>
<th>...</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>das</td>
<td>0.25</td>
<td>0.5</td>
<td>0.6364</td>
<td>0.7479</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>book</td>
<td>das</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1208</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>house</td>
<td>das</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1313</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>the</td>
<td>buch</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1208</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>book</td>
<td>buch</td>
<td>0.25</td>
<td>0.5</td>
<td>0.6364</td>
<td>0.7479</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>buch</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1313</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>book</td>
<td>ein</td>
<td>0.25</td>
<td>0.5</td>
<td>0.4286</td>
<td>0.3466</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>ein</td>
<td>0.25</td>
<td>0.5</td>
<td>0.5714</td>
<td>0.6534</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>haus</td>
<td>0.25</td>
<td>0.5</td>
<td>0.4286</td>
<td>0.3466</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>house</td>
<td>haus</td>
<td>0.25</td>
<td>0.5</td>
<td>0.5714</td>
<td>0.6534</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>
Perplexity

• How well does the model fit the data?

• Perplexity: derived from probability of the training data according to the model

\[
\log_2 PP = - \sum_s \log_2 p(e_s | f_s)
\]

• Example (\(\epsilon=1\))

<table>
<thead>
<tr>
<th></th>
<th>initial</th>
<th>1st it.</th>
<th>2nd it.</th>
<th>3rd it.</th>
<th>...</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p(\text{the haus}</td>
<td>\text{das haus}))</td>
<td>0.0625</td>
<td>0.1875</td>
<td>0.1905</td>
<td>0.1913</td>
<td>...</td>
</tr>
<tr>
<td>(p(\text{the book}</td>
<td>\text{das buch}))</td>
<td>0.0625</td>
<td>0.1406</td>
<td>0.1790</td>
<td>0.2075</td>
<td>...</td>
</tr>
<tr>
<td>(p(\text{a book}</td>
<td>\text{ein buch}))</td>
<td>0.0625</td>
<td>0.1875</td>
<td>0.1907</td>
<td>0.1913</td>
<td>...</td>
</tr>
<tr>
<td>perplexity</td>
<td>4095</td>
<td>202.3</td>
<td>153.6</td>
<td>131.6</td>
<td>...</td>
<td>113.8</td>
</tr>
</tbody>
</table>
Higher IBM Models

<table>
<thead>
<tr>
<th>IBM Model 1</th>
<th>lexical translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Model 2</td>
<td>adds absolute reordering model</td>
</tr>
<tr>
<td>IBM Model 3</td>
<td>adds fertility model</td>
</tr>
<tr>
<td>IBM Model 4</td>
<td>relative reordering model</td>
</tr>
<tr>
<td>IBM Model 5</td>
<td>fixes deficiency</td>
</tr>
</tbody>
</table>

- Only IBM Model 1 has global maximum
  - training of a higher IBM model builds on previous model

- Computationally biggest change in Model 3
  - trick to simplify estimation does not work anymore
    → exhaustive count collection becomes computationally too expensive
  - sampling over high probability alignments is used instead
word alignment
Word Alignment

Given a sentence pair, which words correspond to each other?

<table>
<thead>
<tr>
<th>Michael</th>
<th>geht</th>
<th>davon</th>
<th>aus</th>
<th>dass</th>
<th>er</th>
<th>im</th>
<th>haus</th>
<th>bleibt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael</td>
<td>assumes</td>
<td>that</td>
<td>he</td>
<td>will</td>
<td>stay</td>
<td>in</td>
<td>the</td>
<td>house</td>
</tr>
</tbody>
</table>

Philipp Koehn  Machine Translation: IBM Model 1 and the EM Algorithm  13 September 2018
Word Alignment?

Is the English word *does* aligned to the German *wohnt* (verb) or *nicht* (negation) or neither?
How do the idioms \textit{kicked the bucket} and \textit{biss ins grass} match up? Outside this exceptional context, bucket is never a good translation for grass.
Measuring Word Alignment Quality

- Manually align corpus with sure ($S$) and possible ($P$) alignment points ($S \subseteq P$)
- Common metric for evaluation word alignments: Alignment Error Rate (AER)

$$\text{AER}(S, P; A) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

- AER = 0: alignment $A$ matches all sure, any possible alignment points
- However: different applications require different precision/recall trade-offs
symmetrization
Word Alignment with IBM Models

• IBM Models create a **many-to-one** mapping
  – words are aligned using an alignment function
  – a function may return the same value for different input
    (one-to-many mapping)
  – a function can not return multiple values for one input
    (no many-to-one mapping)

• Real word alignments have **many-to-many** mappings
Symmetrization

- Run IBM Model training in both directions
  - two sets of word alignment points

- Intersection: high precision alignment points

- Union: high recall alignment points

- Refinement methods explore the sets between intersection and union
Example

**english to spanish**

<table>
<thead>
<tr>
<th>Maria no daba una bofetada a la bruja verde</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
</tr>
<tr>
<td>did</td>
</tr>
<tr>
<td>not</td>
</tr>
<tr>
<td>slap</td>
</tr>
<tr>
<td>the</td>
</tr>
<tr>
<td>green</td>
</tr>
<tr>
<td>witch</td>
</tr>
</tbody>
</table>

**spanish to english**

<table>
<thead>
<tr>
<th>Maria no daba una bofetada a la bruja verde</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
</tr>
<tr>
<td>did</td>
</tr>
<tr>
<td>not</td>
</tr>
<tr>
<td>slap</td>
</tr>
<tr>
<td>the</td>
</tr>
<tr>
<td>green</td>
</tr>
<tr>
<td>witch</td>
</tr>
</tbody>
</table>

**intersection**

<table>
<thead>
<tr>
<th>Maria no daba una bofetada a la bruja verde</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
</tr>
<tr>
<td>did</td>
</tr>
<tr>
<td>not</td>
</tr>
<tr>
<td>slap</td>
</tr>
<tr>
<td>the</td>
</tr>
<tr>
<td>green</td>
</tr>
<tr>
<td>witch</td>
</tr>
</tbody>
</table>
Growing Heuristics

- Add alignment points from union based on heuristics:
  - directly/diagonally neighboring points
  - finally, add alignments that connect unaligned words in source and/or target
- Popular method: grow-diag-final-and
Phrase-Based Models

Philipp Koehn

18 September 2018
Phrase-Based Model

- Foreign input is segmented in phrases
- Each phrase is translated into English
- Phrases are reordered
Phrase Translation Table

- Main knowledge source: table with phrase translations and their probabilities

- Example: phrase translations for *natuerlich*

| Translation       | Probability $\phi(\tilde{e}|f)$ |
|-------------------|----------------------------------|
| of course         | 0.5                              |
| naturally         | 0.3                              |
| of course ,       | 0.15                             |
| , of course ,     | 0.05                             |
Scoring Phrase Translations

• Phrase pair extraction: collect all phrase pairs from the data

• Phrase pair scoring: assign probabilities to phrase translations

• Score by relative frequency:

\[ \phi(f|e) = \frac{\text{count}(e, f)}{\sum_{f_i} \text{count}(e, f_i)} \]
• Phrase translations for den Vorschlag learned from the Europarl corpus:

| English             | $\phi(\tilde{e}|f)$ | English             | $\phi(\tilde{e}|f)$ |
|---------------------|----------------------|---------------------|----------------------|
| the proposal        | 0.6227               | the suggestions     | 0.0114               |
| 's proposal         | 0.1068               | the proposed        | 0.0114               |
| a proposal          | 0.0341               | the motion          | 0.0091               |
| the idea            | 0.0250               | the idea of         | 0.0091               |
| this proposal       | 0.0227               | the proposal        | 0.0068               |
| proposal            | 0.0205               | its proposal        | 0.0068               |
| of the proposal     | 0.0159               | it                  | 0.0068               |
| the proposals       | 0.0159               | ...                 | ...                  |

- lexical variation (proposal vs suggestions)
- morphological variation (proposal vs proposals)
- included function words (the, a, ...)
- noise (it)
extract phrase pair consistent with word alignment:

assumes that / geht davon aus, dass
All words of the phrase pair have to align to each other.
Phrase Pair Extraction

Smallest phrase pairs:

michael — michael
assumes — geht davon aus / geht davon aus ,
that — dass / , dass
he — er
will stay — bleibt
in the — im
house — haus

unaligned words (here: German comma) lead to multiple translations
michael assumes — michael geht davon aus / michael geht davon aus,
assumes that — geht davon aus, dass ; assumes that he — geht davon aus, dass er
that he — dass er / , dass er ; in the house — im haus
michael assumes that — michael geht davon aus , dass
michael assumes that he — michael geht davon aus , dass er
michael assumes that he will stay in the house — michael geht davon aus , dass er im haus bleibt
assumes that he will stay in the house — geht davon aus , dass er im haus bleibt
that he will stay in the house — dass er im haus bleibt ; dass er im haus bleibt ,
he will stay in the house — er im haus bleibt ; will stay in the house — im haus bleibt
More Feature Functions

- Bidirectional alignment probabilities: $\phi(\bar{e}|\bar{f})$ and $\phi(\bar{f}|\bar{e})$

- Rare phrase pairs have unreliable phrase translation probability estimates → lexical weighting with word translation probabilities

\[
\text{lex}(\bar{e}|\bar{f}, a) = \prod_{i=1}^{\text{length}(\bar{e})} \frac{1}{|\{j|(i, j) \in a\}|} \sum_{\forall(i, j)\in a} w(e_i|f_j)
\]

<table>
<thead>
<tr>
<th></th>
<th>geht</th>
<th>nicht</th>
<th>davon</th>
<th>aus</th>
<th>NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>does</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>assume</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Distance-Based Reordering

Scoring function: $d(x) = \alpha^{|x|}$ — exponential with distance
Decoding

Philipp Koehn

20 September 2018
Many translation options to choose from

- in Europarl phrase table: 2727 matching phrase pairs for this sentence
- by pruning to the top 20 per phrase, 202 translation options remain
The machine translation decoder does not know the right answer
- picking the right translation options
- arranging them in the right order

→ Search problem solved by heuristic beam search
consult phrase translation table for all input phrases
Decoding: Start with Initial Hypothesis

initial hypothesis: no input words covered, no output produced
Decoding: Hypothesis Expansion

pick any translation option, create new hypothesis
Decoding: Hypothesis Expansion

create hypotheses for all other translation options
Decoding: Hypothesis Expansion

also create hypotheses from created partial hypothesis
Decoding: Find Best Path

Backtrack from highest scoring complete hypothesis
dynamic programming
• The suggested process creates exponential number of hypothesis

• Machine translation decoding is NP-complete

• Reduction of search space:
  – recombination (risk-free)
  – pruning (risky)
Recombination

- Two hypothesis paths lead to two matching hypotheses
  - same foreign words translated
  - same English words in the output

- Worse hypothesis is dropped
pruning
• Hypothesis expansion in a stack decoder
  – translation option is applied to hypothesis
  – new hypothesis is dropped into a stack further down
Stack Decoding Algorithm

1: place empty hypothesis into stack 0
2: for all stacks 0...n − 1 do
3:   for all hypotheses in stack do
4:     for all translation options do
5:       if applicable then
6:         create new hypothesis
7:         place in stack
8:         recombine with existing hypothesis if possible
9:       prune stack if too big
10:   end if
11: end for
12: end for
13: end for
Pruning

• Pruning strategies
  – histogram pruning: keep at most $k$ hypotheses in each stack
  – stack pruning: keep hypothesis with score $\alpha \times$ best score ($\alpha < 1$)

• Computational time complexity of decoding with histogram pruning

  $O(\text{max stack size} \times \text{translation options} \times \text{sentence length})$

• Number of translation options is linear with sentence length, hence:

  $O(\text{max stack size} \times \text{sentence length}^2)$

• Quadratic complexity
future cost estimation
Translating the Easy Part First?

the tourism initiative addresses this for the first time

both hypotheses translate 3 words
worse hypothesis has better score
Estimating Future Cost

• Future cost estimate: how expensive is translation of rest of sentence?

• Optimistic: choose cheapest translation options

• Cost for each translation option
  – translation model: cost known
  – language model: output words known, but not context
    → estimate without context
  – reordering model: unknown, ignored for future cost estimation
Cost Estimates from Translation Options

the tourism initiative addresses this for the first time

-1.0 -2.0 -1.5 -2.4 -1.4 -1.0 -1.0 -1.9 -1.6

-4.0 -2.5 -2.2 -1.3 -2.4 -2.7

-2.3

-2.3

-2.3

cost of cheapest translation options for each input span (log-probabilities)
Cost Estimates for all Spans

- Compute cost estimate for all contiguous spans by combining cheapest options

<table>
<thead>
<tr>
<th>first word</th>
<th>future cost estimate for $n$ words (from first)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>-1.0</td>
</tr>
<tr>
<td>tourism</td>
<td>-2.0</td>
</tr>
<tr>
<td>initiative</td>
<td>-1.5</td>
</tr>
<tr>
<td>addresses</td>
<td>-2.4</td>
</tr>
<tr>
<td>this</td>
<td>-1.4</td>
</tr>
<tr>
<td>for</td>
<td>-1.0</td>
</tr>
<tr>
<td>the</td>
<td>-1.0</td>
</tr>
<tr>
<td>first</td>
<td>-1.9</td>
</tr>
<tr>
<td>time</td>
<td>-1.6</td>
</tr>
</tbody>
</table>

- Function words cheaper (the: -1.0) than content words (tourism: -2.0)
- Common phrases cheaper (for the first time: -2.3) than unusual ones (tourism initiative addresses: -5.9)
Combining Score and Future Cost

- Hypothesis score and future cost estimate are combined for pruning
  - left hypothesis starts with hard part: **the tourism initiative**
    score: -5.88, future cost: -6.1 → total cost -11.98
  - middle hypothesis starts with easiest part: **the first time**
    score: -4.11, future cost: -9.3 → total cost -13.41
  - right hypothesis picks easy parts: **this for ... time**
    score: -4.86, future cost: -9.1 → total cost -13.96
A* Search

- Uses admissible future cost heuristic: never overestimates cost
- Translation agenda: create hypothesis with lowest score + heuristic cost
- Done, when complete hypothesis created