statistical machine translation
Today

• Overview of IBM-2, IBM-3, and phrase-based methods.

• Decoding for SMT.

• Evaluation of MT systems.
Practical note on programming IBM-1

• If you were to code the EM algorithm for IBM-1, you would **not** initialize $\theta = P(f|e)$ uniformly over the entire vocabulary.
  • Don’t make a $V_F \times V_E$ table with $P(f|e) = 1/\|V_E\|$

• This structure would be too large.
  • Probabilities would be too small.
  • It would take too much work to update.

• Rather, initialize a hash table over **possible** alignments, $\mathcal{M}$. For every English word $e$, only consider French words $f$ in sentences **aligned** with English sentences containing $e$.
  • e.g., structure $P.e.f := P(f|e) = 1/\|\mathcal{M}\|$
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 re-ordering model</td>
</tr>
<tr>
<td>IBM Model 3</td>
<td>adds fertility model</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Only IBM Model 1 training reaches a *global maximum*
- Training of each IBM model *extends* the *next lowest* model.

- Higher models become computationally *expensive.*
IBM-2

• Unlike IBM Model-1, the placement of a word in, say, Spanish in IBM Model-2 depends on where its equivalent word was in English.
• IBM-2 captures the intuition that translations should lie roughly “along the diagonal”.

<table>
<thead>
<tr>
<th></th>
<th>Buenos</th>
<th>dias</th>
<th>,</th>
<th>me</th>
<th>gusta</th>
<th>papas</th>
<th>frías</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>day</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>,</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>like</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>cold</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>potatoes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
IBM-2

- IBM Model 2 builds on Model 1 by adding a re-ordering model defined by distortion parameters regardless of actual words.

\[ D(i|j, \mathcal{L}_E, \mathcal{L}_F) = \text{the probability that the } i^{th} \text{ English slot is aligned to the } j^{th} \text{ French slot, given sentence lengths } \mathcal{L}_E \text{ and } \mathcal{L}_F. \]

- In IBM Model 2:

\[ P(a|E, \mathcal{L}_E, \mathcal{L}_F) = \prod_{j=1}^{\mathcal{L}_F} D(a_j|j, \mathcal{L}_E, \mathcal{L}_F) \]

- Recall that in IBM Model 1,

\[ P(a|E, \mathcal{L}_E, \mathcal{L}_F) = \frac{P(\mathcal{L}_F)}{(\mathcal{L}_E + 1)^{\mathcal{L}_F}} \]
IBM-2 – Probability of alignment

• \( E = \text{And the program has been implemented} \)
• \( F = \text{Le programme a été mis en application} \)
• \( L_E = 6 \)
• \( L_F = 7 \)
• \( a = \{2,3,4,5,6,6,6\} \) (i.e., \( f_1 \leftarrow e_2, f_2 \leftarrow e_3, \ldots \))

\[
P(a|E, L_E, L_F) = D(2|1,6,7) \times D(3|2,6,7) \times D(4|3,6,7) \times D(5|4,6,7) \times D(6|5,6,7) \times D(6|6,6,7) \times D(6|7,6,7)
\]
IBM-2: generation

To **generate** a French sentence $F$ from English $E$,

1. **Pick an alignment** with probability

$$\prod_{j=1}^{L_F} D(a_j|j, L_E, L_F)$$

3. **Sample** French words with probability

$$P(F|a, E) = \prod_{j=1}^{L_F} P(f_j|e_{a_j})$$

So,

$$P(F, a|E) = P(a|E)P(F|a, E) = \prod_{j=1}^{L_F} D(a_j|j, L_E, L_F)P(f_j|e_{a_j})$$

This is the same $P(f|e)$ as in IBM-1.
IBM-2: training

• We use EM, as before with IBM-1 except that we need to take the distortion into account when computing the probability of an alignment.

• We also need to learn the distortion function.

• Aren’t you glad that you don’t need to know how to compute EM for IBM-2?
**IBM-3**

- **IBM Model 3** extends Model 2 by adding a **fertility model** that describes how many **French** words each **English** word can produce.
- In the example below, *implemented* appears to be **more fertile** than *program*.

<table>
<thead>
<tr>
<th>e₀</th>
<th>e₁</th>
<th>e₂</th>
<th>e₃</th>
<th>e₄</th>
<th>e₅</th>
<th>e₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>Canada</td>
<td>‘s</td>
<td>program</td>
<td>has</td>
<td>been</td>
<td>implemented</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f₁</th>
<th>f₂</th>
<th>f₃</th>
<th>f₄</th>
<th>f₅</th>
<th>f₆</th>
<th>f₇</th>
<th>f₈</th>
<th>f₉</th>
</tr>
</thead>
<tbody>
<tr>
<td>Le</td>
<td>programme</td>
<td>du</td>
<td>Canada</td>
<td>a</td>
<td>été</td>
<td>mis</td>
<td>en</td>
<td>application</td>
</tr>
</tbody>
</table>
IBM-3: The generation model

• First, we **replicate** each word according to a new hidden parameter, $N(n|e)$, which is the **probability** that word $e$ produces $n$ words.
• We then **re-align** (with distortion) and **translate** as we did in IBM-2.
# 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 re-ordering model</td>
</tr>
<tr>
<td>IBM Model 3</td>
<td>adds fertility model</td>
</tr>
</tbody>
</table>
No more words

I want to see phrases
Phrase-based statistical MT

- **Phrase-based** statistical MT involves segmenting sentences into contiguous blocks or segments.
  - Each phrase is probabilistically translated. e.g., $P(\text{zu Hause} | \text{at home})$
  - Each phrase is probabilistically re-ordered.

```
The green witch is at home this week
```
```
Diese Woche ist die grüne Hexe zu Hause
```
Phrase-based statistical MT

• Phrase-based SMT allows many-to-many word mappings.

• Larger context allows for some disambiguation that is not possible in word-based alignment.
  • E.g.,
    
    \[
    P(\text{intéret} \mid \text{interest}) \\
    \text{vs.} \\
    P(\text{taux d’intérêt} \mid \text{interest rate}) \\
    > P(\text{taux d’intérêt} \mid \text{interest in})
    \]

    No context 😞

    A tiny amount of context 😊
Learning phrase-translations

- Typically, we use alignment templates (Och et al., 1999).
- Start with a word-alignment, then build phrases.

Maria no dió una bofetada a la bruja verde

Mary did not slap the green witch

This word-alignment is produced by a model like IBM-3
Learning phrase-translations

- A phrase alignment **must** contain **all** word alignments for each of its rows and columns.
- Collect **all** phrase alignments that are **consistent** with the **word alignment**, e.g.

<table>
<thead>
<tr>
<th></th>
<th>Maria</th>
<th>no</th>
<th>dió</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>did</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slap</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Maria</th>
<th>no</th>
<th>dió</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>did</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>slap</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Maria</th>
<th>no</th>
<th>dió</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>did</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slap</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Consistent  
Inconsistent  
Inconsistent
Learning phrase-translations

• Given word-alignments (produced automatically or otherwise), we do not need to do EM training. E.g.,

\[
P(f_1f_2 | e_1e_2e_3) = \frac{\text{Count}(f_1f_2,e_1e_2e_3)}{\text{Count}(e_1e_2e_3)}
\]
Phrase-based translation in practice

What is the legal drinking age in Quebec?

Quel est l'âge légal pour boire au Québec?
No more models

I want to see decoding
Decoding

- **Decoding** is the act of translating a ‘foreign’ language into your native language.
  - Decoding is an NP-complete problem (Knight, 1999).

- IBM Models often decoded with **stack decoding or A* search**.

  - Introduces **greedy decoding** – start with a solution and incrementally try to **improve** it.
First stage of greedy method

- For each French word $f_j$, pick the English word $e^*$ such that

$$e^* = \arg\max_e P(f_j | e)$$

- This gives an initial alignment, e.g.,

<table>
<thead>
<tr>
<th>Bien</th>
<th>entendu</th>
<th>,</th>
<th>il</th>
<th>parole</th>
<th>d’</th>
<th>une</th>
<th>belle</th>
<th>victoire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Well</td>
<td>heard</td>
<td>,</td>
<td>it</td>
<td>talking</td>
<td>Ø</td>
<td>a</td>
<td>beautiful</td>
<td>victory</td>
</tr>
</tbody>
</table>

(Better: *quite naturally, he talks about a great victory*)
Some transformations

- **Change\( (j, e) \)**: sets translation of \( f_j \) to \( e \)
  - Usually we only consider English words \( e \) that are in the top \( N \) ranked translations for \( f_j \).

- **Change2\( (j_1, e_1, j_2, e_2) \)**: sets translation of \( f_{j_1} \) to \( e_1 \) and translation of \( f_{j_2} \) to \( e_2 \)
  - Like performing two \textit{Change} transformations in sequence, but \textit{without} evaluating the intermediate string.

- **ChangeAndInsert\( (j, e_1, e_2) \)**: sets translation of \( f_j \) to \( e_1 \) and inserts \( e_2 \) at its most likely position.
Some more transformations

- **RemoveInfertile(i)**: Removes $e_i$ if $e_i$ is aligned with no French words.

- **SwapSeg(i₁, i₂, j₁, j₂)**: Swaps segment $e_{i₁:i₂}$ with segment $e_{j₁:j₂}$ such that segments do not overlap.

- **JoinWords(i₁, i₂)**: Removes $e_{i₁}$ and aligns all French words that were aligned to $e_{i₁}$ to $e_{i₂}$. 
Iterating greedily

• We have an initial pair \((E^{(0)}, a^{(0)})\).

• Use local **transformations** to map \((E, a)\) to new pairs, \((E', a')\).

• At each iteration, \(k\), take the highest probability pair from all possible transformations
  • i.e., if \(\mathcal{R}(E^{(k)}, a^{(k)})\) is the set of all \((E, a)\) ‘reachable’ from \((E^{(k)}, a^{(k)})\), then at each iteration:

\[
(E^{(k+1)}, a^{(k+1)}) = \arg\max_{(E,a) \in \mathcal{R}(E^{(k)},a^{(k)})} P(E)P(F, a | E)
\]
Example of greedy search

<table>
<thead>
<tr>
<th>Bien</th>
<th>entendu</th>
<th>,</th>
<th>il</th>
<th>parole</th>
<th>d’</th>
<th>une</th>
<th>belle</th>
<th>victoire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Well</td>
<td>heard</td>
<td>,</td>
<td>it</td>
<td>talking</td>
<td>Ø</td>
<td>a</td>
<td>beautiful</td>
<td>victory</td>
</tr>
</tbody>
</table>

\[ \text{Change}_2(5, \text{talks}, 8, \text{great}) \]
Example of greedy search

<table>
<thead>
<tr>
<th>Bien</th>
<th>intendu</th>
<th>,</th>
<th>il</th>
<th>parle</th>
<th>d’</th>
<th>une</th>
<th>belle</th>
<th>victoire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Well</td>
<td>heard</td>
<td>,</td>
<td>it</td>
<td>talks</td>
<td>Ø</td>
<td>a</td>
<td>great</td>
<td>victory</td>
</tr>
</tbody>
</table>

\[ \text{Change2}(2, \text{understood}, 6, \text{about}) \]
Example of greedy search

<table>
<thead>
<tr>
<th>Bien</th>
<th>intendu</th>
<th>,</th>
<th>il</th>
<th>parole</th>
<th>d’</th>
<th>une</th>
<th>belle</th>
<th>victoire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Well</td>
<td>understood</td>
<td>,</td>
<td>it</td>
<td>talks</td>
<td>about</td>
<td>a</td>
<td>great</td>
<td>victory</td>
</tr>
</tbody>
</table>

Change(4, he)
Example of greedy search

<table>
<thead>
<tr>
<th>Bien</th>
<th>intendu</th>
<th>,</th>
<th>il</th>
<th>parole</th>
<th>d’</th>
<th>une</th>
<th>belle</th>
<th>victoire</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Well understood</strong></td>
<td>,</td>
<td>he</td>
<td>talks</td>
<td>about</td>
<td>a</td>
<td>great</td>
<td>victory</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{Change2}(1, \text{quite}, 2, \text{naturally})
\]

<table>
<thead>
<tr>
<th>Bien</th>
<th>intendu</th>
<th>,</th>
<th>il</th>
<th>parole</th>
<th>d’</th>
<th>une</th>
<th>belle</th>
<th>victoire</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quite naturally</strong></td>
<td>,</td>
<td>he</td>
<td>talks</td>
<td>about</td>
<td>a</td>
<td>great</td>
<td>victory</td>
<td></td>
</tr>
</tbody>
</table>
Greedy transformations

- At each iteration, we try *each possible* transformation.

- For each possible transformation, we evaluate

\[ P(E)P(F, a | E) \]

- We choose the transformation that gives the highest probability, and iterate until some stopping condition.
No more decoding

I want to see evaluation
Evaluation of MT systems

<table>
<thead>
<tr>
<th>Human</th>
<th>According to the data provided today by the Ministry of Foreign Trade and Economic Cooperation, as of November this year, China has actually utilized 46.959B US dollars of foreign capital, including 40.007B US dollars of direct investment from foreign businessmen.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM4</td>
<td>The Ministry of Foreign Trade and Economic Cooperation, including foreign direct investment 40.007B US dollars today provide data include that year to November China actually using foreign 46.959B US dollars and</td>
</tr>
<tr>
<td>Yamada/Knight</td>
<td>Today’s available data of the Ministry of Foreign Trade and Economic Cooperation shows that China’s actual utilization of November this year will include 40.007B US dollars for the foreign direct investment among 46.959B US dollars in foreign capital.</td>
</tr>
</tbody>
</table>

How can we objectively compare the quality of two translations?
Automatic evaluation

• We want an **automatic** and effective method to **objectively** rank competing translations.
  • **Word Error Rate (WER)** measures the number of erroneous word **insertions**, **deletions**, **substitutions** in a translation.
  • E.g.,
    - **Reference**: *how to recognize speech*
    - **Translation**: *how understand a speech*

• **Problem**: There are many possible valid translations. (There’s no need for an exact match)
Challenges of evaluation

• **Human judges:** expensive, slow, non-reproducible (different judges – different biases).

• Multiple valid translations, e.g.:
  - **Source:** *Il s’agit d’un guide qui assure que l’armée sera toujours fidèle au Parti*
  - **T1:** *It is a guide to action that ensures that the military will forever heed Party commands*
  - **T2:** *It is the guiding principle which guarantees the military forces always being under command of the Party*
BLEU evaluation

• BLEU (BiLingual Evaluation Understudy) is an automatic and popular method for evaluating MT.
  • It uses multiple human reference translations, and looks for local matches, allowing for phrase movement.

• Candidate: n. a translation produced by a machine.

• There are a few parts to a BLEU score...
Example of BLEU evaluation

- **Reference 1**: It is a guide to action that ensures that the military will forever heed Party commands
- **Reference 2**: It is the guiding principle which guarantees the military forces always being under command of the Party
- **Reference 3**: It is the practical guide for the army always to heed the directions of the party

- **Candidate 1**: It is a guide to action which ensures that the military always obeys the commands of the party
- **Candidate 2**: It is to insure the troops forever hearing the activity guidebook that party direct
BLEU: Unigram precision

- The **unigram precision** of a candidate is
  \[
  \frac{C}{N}
  \]
  where \(N\) is the number of words in the **candidate** and \(C\) is the number of words in the **candidate** which are in **at least one reference**.

- e.g., **Candidate 1**: *It is a guide to action which ensures that the military always obeys the commands of the party*
  - **Unigram precision** = \(\frac{17}{18}\)
  - *(obeys appears in none of the three references)*.
BLEU: Modified unigram precision

• Reference 1: *The lunatic is on the grass*
• Reference 2: *There is a lunatic upon the grass*
• Candidate: *The the the the the the the the*
  • Unigram precision $= \frac{7}{7} = 1$

• Capped unigram precision:
  A candidate word type $w$ can only be correct a maximum of $\text{cap}(w)$ times.
  • e.g., with $\text{cap}(\text{the}) = 2$, the above gives
  $$ p_1 = \frac{2}{7} $$
BLEU: Generalizing to $N$-grams

• Generalizes to higher-order $N$-grams.
  
  • Reference 1: It is a guide to action that ensures that the military will forever heed Party commands
  
  • Reference 2: It is the guiding principle which guarantees the military forces always being under command of the Party
  
  • Reference 3: It is the practical guide for the army always to heed the directions of the party

• Candidate 1: It is a guide to action which ensures that the military always obeys the commands of the party

• Candidate 2: It is to insure the troops forever hearing the activity guidebook that party direct

Bigram precision, $p_2$

\[ p_2 = \frac{10}{17} \]

\[ p_2 = \frac{1}{13} \]
BLEU: Precision is not enough

• **Reference 1**: It is a guide to action that ensures that the military will forever heed Party commands

• **Reference 2**: It is the guiding principle which guarantees the military forces always being under command of the Party

• **Reference 3**: It is the practical guide for the army always to heed the directions of the party

• **Candidate 1**: of the

Unigram precision, $p_1 = \frac{2}{2} = 1$  
Bigram precision, $p_2 = \frac{1}{1} = 1$
BLEU: Brevity

- **Solution:** Penalize brevity.
- **Step 1:** for each candidate, find the reference most similar in length.
- **Step 2:** $c_i$ is the length of the $i^{th}$ candidate, and $r_i$ is the nearest length among the references,

$$brevity_i = \frac{r_i}{c_i}$$

- **Step 3:** multiply precision by the (0..1) **brevity penalty**:

$$BP = \begin{cases} 
1 & \text{if} \ brevity < 1 \\
\exp\left(1 - \beta \cdot brevity\right) & \text{if} \ brevity \geq 1 
\end{cases}$$

Bigger = too brief
BLEU: Final score

- On slide 39, \( r_1 = 16, r_2 = 17, r_3 = 16, \) and \( c_1 = 18 \) and \( c_2 = 14, \)

\[
\begin{align*}
\text{brevity}_1 &= \frac{17}{18} \quad BP_1 = 1 \\
\text{brevity}_2 &= \frac{16}{14} \quad BP_2 = e^{1 - \left(\frac{\theta}{7}\right)} = 0.8669
\end{align*}
\]

- **Final score** of candidate \( C \):

\[
\text{BLEU} = BP_C \times (p_1 p_2 \ldots p_n)^{1/n}
\]

where \( p_n \) is the \( n \)-gram precision. (You can set \( n \) empirically)
Example: Final BLEU score

- **Reference 1:** I am afraid Dave
- **Reference 2:** I am scared Dave
- **Reference 3:** I have fear David
- **Candidate:** I fear David

- **brevity = \[
\frac{4}{3} \geq 1 \text{ so } BP = e^{1 - \left(\frac{4}{3}\right)}
\]**

- \[p_1 = \frac{1+1+1}{3} = 1\]
- \[p_2 = \frac{1}{2}\]

- **BLEU = BP \left(p_1 p_2\right)^{\frac{1}{2}} = e^{1 - \left(\frac{4}{3}\right)} \left(\frac{1}{2}\right)^{\frac{1}{2}} \approx 0.5067**

Assume \(cap(\cdot) = 2\) for all \(N\)-grams

Also assume BLEU order \(n = 2\)
BLEU: summary

- BLEU is a geometric mean over $n$-gram precisions.
  - These precisions are capped to avoid strange cases.
    - E.g., the translation “the the the the” is not favoured.
  - This geometric mean is weighted so as not to favour unrealistically short translations, e.g., “the”

- Initially, evaluations showed that BLEU predicted human judgements very well, but:
  - People started optimizing MT systems to maximize BLEU. Correlations between BLEU and humans decreased.
Reading


• Useful reading on IBM Model-1: Section 25.5 of the 2nd edition of the Jurafsky & Martin text.
  • 1st edition available at Robarts library.

• Other: Manning & Schütze Sections 13.1.2 (Gale&Church), 13.1.3 (Church), 13.3
Announcements – assignment 2

• We’ve placed an alternative decoder, `decode2.m`, on CDF at `/u/cs401/A2_SMT/code/decode2.m`.
• For Task 5 you can use either decoder, or improve either, or write your own – it’s all equivalent – just be sure to say which decoder you use and submit any changes you make.
Announcements

• Assignment 1 marks/comments will be emailed individually.

• Not-for-marks midterm on Monday 7 March.

• Information on the final is posted on the course webpage.
  • 18 April 9h-12h in EX