Inference for Machine Translation

Micha Elsner

March 28, 2013
Noisy channel model:

- To translate *German* into *English*
- Applied Bayes’ thm: problem decomposes into:
  - *English* language model $P(E)$
  - *English-to-German* translation model $P(F|E)$

\[
\max_E P(E|F) = \max_E P(E)P(F|E)
\]
IBM model 1 specifies $P(F|E)$:

- Based on alignment: each foreign word has a single English source word
- Alignments represented by hidden variables $A_{1:m}$
- French words independent of one another...
- Given alignment, also independent of all but one English word
- Actual translations stored in probabilistic dictionary $P_{tr}$

$$P(F_i|E_{1:n}, a_i) = P_{tr}(F_i|E_{a_i})$$
Today’s lecture

We’ve used EM to learn $P_{tr}$...

- Still have to do inference (compute $\max_E$ expression)

Will also look at some improvements to the basic IBM 1 model
Decoding

MT people call the inference problem “decoding”

▶ Just to confuse you?

We’ve discussed estimation, but how do we solve this max?

\[
\max_{E} P(E|F) = \max_{E} P(E)P(F|E)
\]
Some dumb approximations

Charniak’s, designed for pedagogical purposes only:

The very dumb decoder
- Drop the language model entirely
- Translate every French word once, in order

\[
\max_E P(E|F) = \max_E \left( \prod_i P_{tr}(E_i|F_i) \right)
\]
Very dumb decoding

Je sais que tu es un chat noir

- *Je*: I
- *sais*: know
- *que*: that
- *tu*: you
- *es*: role (p=.36, are: p=.23)
- *un*: a (p=.73, an: p=.08)
- *chat*: cat
- *noir*: says (p=.42, black=.27)

I know that you role a cat says
Issues

- VDD: I know that you are a cat black
- vs: I know you are a black cat
- Each word is translated (*que* can’t be omitted)
- No extra words can be inserted
- No reordering
  - *cat black* instead of *black cat*
- No phonological effects
  - *a apple* instead of *an apple*
- Similarly, wrong prepositions and particles for verbs
Dumb decoding

Our effort to do without the LM is not going so well...

The dumb decoder

- Keep the language model
- Translate every French word once, in order
- Pick English words one at a time

\[
\max_E P(E|F) = \max_{E_1} P(E_1) P_{tr}(E_1|F_1) \max_{E_2} P(E_2|E_1) P_{tr}(E_2|F_2) \ldots
\]

This might begin to help us pick correct particles or prepositions:

- Ex: German *an* can mean *at, on, by*
- *stehen am Fenster* “to stand by the window”
- *beginnen am Anfang* “to start at the beginning”
- English (trigram) LM could tell the difference
At each point, we need to maximize:

\[ P(E_i|E_{i-1})P(E_i|F_i) \]

- Could compute this product for every word
- However, can do this somewhat faster...

Since these are less than 1,

\[ P(E_i|E_{i-1})P(E_i|F_i) < P(E_i|F_i) \]

- Loop over \( E_i \) in order of \( P(E_i|F_i) \)
- Keep track of best product \( P^* = P(E_i|E_{i-1})P(E_i|F_i) \)
- As soon as \( P(E_i|F_i) < P^* \), can stop
Better modeling, better decoding

- Our decoder can’t do reordering
- But our model is too terrible for this to matter much
- All alignments uniform: no good reordering strategy
- So left-to-right constraint in decoding actually helps us!

Next section: introduce some models that can do reordering...
- Then return to decoding them
Modeling improvements

IBM model 1 makes some really simplistic assumptions

▶ We’ll start off by improving the alignment model
▶ Leaving phrase-based and syntactic models till next time
Why improve alignments?

Our translation system should be able to do some reordering:

- *un chat noir* “a black cat”
- *Dir kann ich das buch geben* “I can give you the book”

But it shouldn’t do too much reordering:

- *un chat noir et un chien gris* “a black cat and a gray dog”
- IBM 1: uniform alignments
- Equally good: “a gray cat and a black dog”
  - The LM can’t rescue us here
- Need a model that can distinguish plausible from implausible reorderings
First attempt: IBM model 2

IBM 1: uniform alignments:

\[
P(A_1:m|E_1:n) = \prod_i \frac{1}{n}
\]

IBM 2: alignments independent of one another but depend on position in French sentence

- Called distortion

\[
P(A_1:m|E_1:n) = \prod_i P_{dis}(A_i = j|i, m, n)
\]

In languages like English and French (both SVO) we expect:

- \(P_{dis}(A_2 = 2|i = 2, m, n)\) pretty high
- \(P_{dis}(A_2 = 3|i = 2, m, n)\) still high
  - 3rd French word corresponds to 2nd English: *le chat noir*
- \(P_{dis}(A_2 = 6|i = 2, m, n)\) pretty low
  - 3rd French word translates 6th English: *le chat gris*
- Can learn that last German word (verb) often comes from middle of English sentence
Learning IBM model 2

We need to incorporate $P_{\text{dis}}$ distortion distribution

Before:

$$P(F_i|E_{1:n}) = \sum_{A_i \in [1...n]} P_{tr}(F_i|E_{A_i}) \frac{1}{n}$$

And therefore (derived by applying Bayes’ thm), the marginals we need for partial counts are:

$$P(A_i = j|F_i, E_{1:n}) = \frac{P_{tr}(F_i|E_j) \frac{1}{n}}{\sum_{A_i \in [1...n]} P_{tr}(F_i|E_{1:n}, A_i) \frac{1}{n}}$$

Now, can no longer divide out the $\frac{1}{n}$

$$P(A_i = j|F_i, E_{1:n}) = \frac{P_{tr}(F_i|E_j)P_{\text{dis}}(j|i, m, n)}{\sum_{A_i \in [1...n]} P_{tr}(F_i|E_{1:n}, A_i)P_{\text{dis}}(A_i|i, m, n)}$$
As usual, EM benefits from a good initial starting point:

- Learn $P_{tr}$ values with IBM 1
- Values for IBM 2 should be similar; initialize $P_{tr}$ from model 1, $P_{dis}$ uniform
- Then start running EM for IBM 2
HMM-based IBM alignments (Model 2.5)

Charniak 3.7
Phrases tend to move together:

- *Das schwarze Buch habe ich gelesen* “I read the black book”
- IBM 2: alignments are independent of one another
  - Coincidentally moved “das”, “schwarze” and “Buch” all to the end
- But this isn’t really a coincidence!

Instead, model alignments as a sequence:

\[
P(A_{1:m}|n) = \prod_{i=0}^{n} P_{\text{trans}}(A_{i+1} = j|A_i, n)
\]

So transition from \(A_1 = 3\) (das/the) to \(A_2 = 4\) (schwarze/black) is high probability
Probability of the whole sentence

\[ P(F_{1:m}, A_{1:m} | E_{1:n}) = \prod_{i=0}^{n} P_{\text{trans}}(A_{i+1} = j | A_i, n) P_{\text{tr}}(F_i | E_{A_i=j}) \]

Can compute marginals using forward-backward

- Number of states changes for each sentence
- = Number of English words
So far, we’ve seen a few strategies for complex inference problems:

- Dynamic programming
  - HMMs, bottom-up parsers...
- ILP/declarative optimization
  - Compression
- Left-to-right search based on heuristics
  - Left-to-right parsers

There are MT decoders from each of these paradigms... mostly left-to-right with dynamic programming augmentations
Recall the left-to-right parser

Our analysis has the following parts:

The *partial tree*:

```
    S
   /   |
  NP   VP
     /   |
    DT  NN ... 
     |
    The ... 
```

The *stack* records the parts in the tree we have yet to fill:

\[ [\text{NN VP}] \]

- The top category on the stack (\textit{NN}) is our *goal*
- The *probability* of the parse and string, up to the current word:
  - Currently .9
Parsing the next word

Our goal category is $NN$:

```
S
  NP  VP
    DT  NN ...
  The ...
```

Next word is $cat$

- Apply $NN \rightarrow cat$, $P=.5$
- Pop $NN$ from stack
  - New goal is $VP$
- Update prob to $0.9 \times 0.5 = 0.45$
Parsing the next word

Our goal category is *NN*:

```
S
  /   \
NP   VP
  / \\  /
DT NN ...
```

Next word is *cat*

- Apply *NN* → *cat*, P=.5
- Pop *NN* from stack
  - New goal is *VP*
- Update prob to \( .9 \times .5 = .45 \)
Parsing the third word

Goal category is $VP$:

$S$

$NP$ $VP$

$DT$ $NN$ $\ldots$

The cat $\ldots$

Next word is *sat*

- Two possible rules: $VP \rightarrow V PP$
- And: $VP \rightarrow V$
- Which to use?
Multiple analyses

Just like in HMM, we can be in multiple states at once:

Analysis 1 (no PP)

Stack: []
Prob: $0.45 \times 0.4 = 0.18$

Analysis 2 (PP)

Stack: [PP] (goal $PP$)
Prob: $0.45 \times 0.6 = 0.27$
Continuing the analysis

We now do twice as much work (we have to parse the next word starting from both analyses)

Analysis 1 (no PP)

Stack: []
Next word is on: no goal category, can’t match: FAIL

Analysis 2 (PP)

Stack: [NP] (goal NP)
Basic left-to-right decoder

(This isn’t quite Charniak’s decoder from 2.5.2; it’s changed to emphasize similarity with the left-to-right parser)

Our analysis has the following parts:

The *partial aligned sentence*:

- I(=Je) know(=sais) (=que) you(=tu)...

The *remaining French words*:

- es un chat noir

The *probability* of the partial aligned sentence:

- eg .00049

And our *heuristic*: a guess as to how much it will cost to produce the remaining French words:

- By the time we’re done, we anticipate a probability .00049 × .000051
Translating another word

For each hypothesis with 8 French words remaining:
  ▶ For each choice of French word to produce next...
    ▶ For each possible English translation
      ▶ Generate a new hypothesis with 7 French words remaining
      ▶ Calculate its heuristic probability
      ▶ If it’s too small, discard it

Then use the 7-word hypotheses with highest heuristic probability to generate 6-word hypotheses, etc
  ▶ Once one or more translations of the entire sentence are done, return the best

The important part is coming up with a good heuristic...
More advanced decoders

From Pharoah decoder manual (Philipp Koehn)

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

27
More advanced decoders

Using dynamic programming ideas in L2R decoding:
More advanced decoders

Heuristic: score each chunk automatically
- Translation score and LM score inside chunk
- Don’t consider ordering or LM scores between chunks
- Heuristic: add up all remaining chunks
The inference problem for MT is difficult
Mostly because of interactions between the LM and translation model
Simple decoders make strong greedy assumptions
In real life, left-to-right search is preferred
Modeling word reordering: IBM model 2
...and IBM model 2.5 (the HMM)
Can sort of deal with certain types of syntactic reorderings
Next time: phrase and syntax-based MT