Human Language Technology:
Applications to Information Access

Lesson 6: Translation Models

November 3, 2016

EPFL Doctoral Course EE-724
Andrei Popescu-Belis
Idiap Research Institute, Martigny

Generative modeling for statistical MT *(reminder)*

- Noisy channel model
  - generation of French sentence *f* considered as a transmission of English sentence *e* into French
  - goal: given *f*, what is the most likely *e*?
    - this will actually produce a translation *e* of *f*
- Principle (using Bayes' theorem)
  - learn English language model: $P(e)$
  - learn (reverse) translation model: $P(f|e)$
  - decode source sentence: find* most likely *e* given *f*

$$\operatorname*{argmax}_{e \in English} P(e \mid f) = \operatorname*{argmax}_{e \in English} \left(P(f \mid e) \cdot P(e) \right)$$

*find: easier said than done!

Translation modeling for MT

- Main question: how do we model $P(e|f)$?
  - given a foreign (French) sentence f (which we don't understand) and an English translation candidate e,
  - how do we compute (estimate) the probability that e is really a translation of f?

- Plan of the lesson
  - IBM Model 1
    - expectation-maximization (EM) algorithm
    - quick overview of IBM Models 2-5
  - phrase-based translation models
  - sentence and word alignment algorithms

Learning a translation model

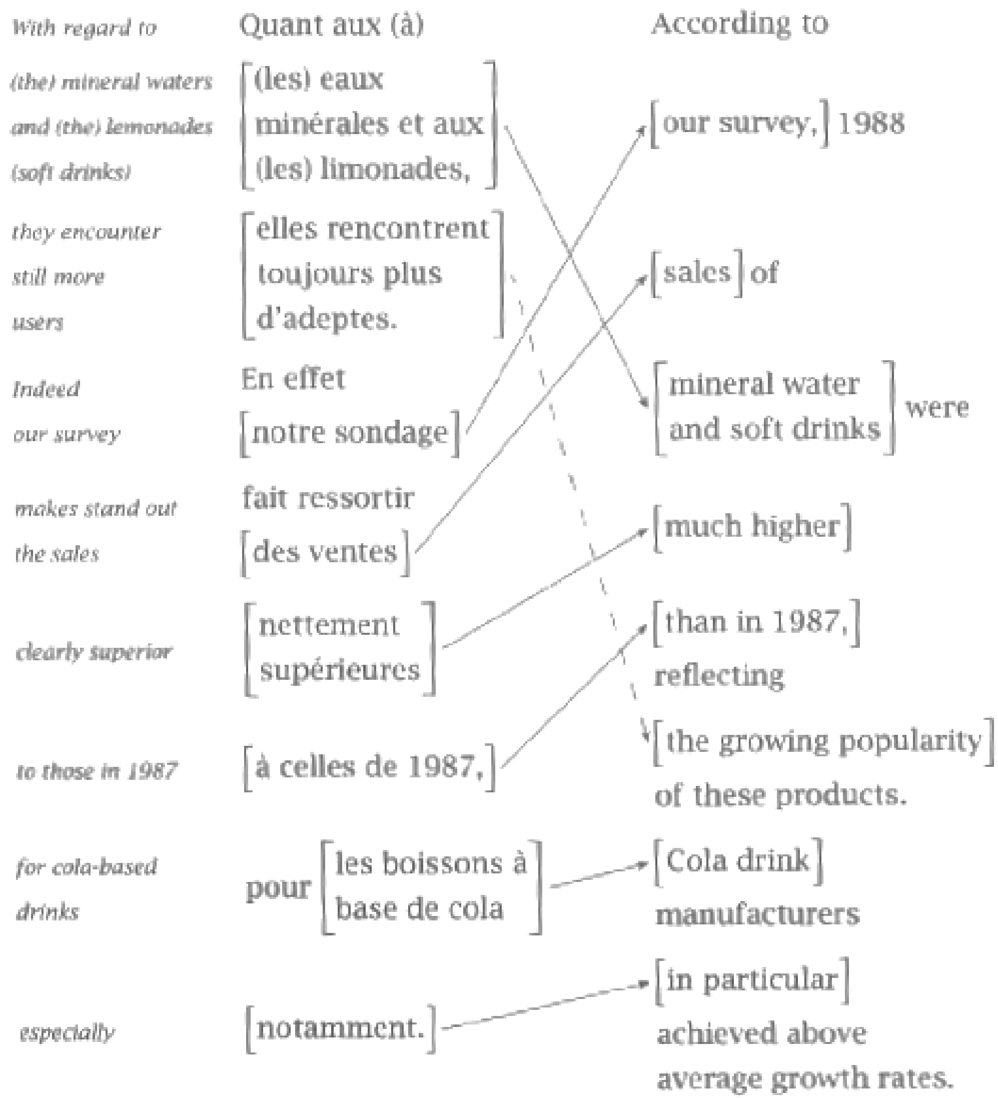
- Estimating parameters of generative model from data
  - be able to estimate $P(e|f)$ for any e, f
    - *$P(e|f)$ or $P(f|e)$ are equivalent because the method is the same*
- Data = parallel corpus: pairs of source and target (human-translated) sentences

- IBM Models 1 and 2
  - IBM Model 1: alignments, EM algorithm, implementation
  - IBM Models 2-5: incremental complexification
- Evaluation of TMs: perplexity

Alignment function: *a*

- Word mapping from a French sentence to an English one
  - DEFINITION: $a(i)$ is the position of the French word that is translated into the English word which is in position i
- Example of a correct alignment
  - French: il_1 a_2 gravi_3 la_4 montagne_5 enneigée_6 →
    English: he_1 climbed_2 the_3 snowy_4 mountain_5
    Alignment: $a(1)=1$, $a(2)=3$, $a(3)=4$, $a(4)=6$, $a(5)=5$
- Remarks
  - not injective; not surjective; needs NULL token (0) in French for English words that are not translations of a French word
- Another example
  - French: NULL_0 je_1 ne_2 veux_3 pas_4 me_5 taire_6 →
    English: I_1 do_2 not_3 want_4 to_5 shut_6 up_7
    Alignment: $a(1)=1$, $a(2)=0$, $a(3)=2$, $a(4)=3$, $a(5)=0$, $a(6)=6$, $a(7)=6$

Example of difficult word (and even sentence) alignments (Jurafsky & Martin 1999, p. 473)



Learning to estimate $P(e|f)$

- Intuitive idea
  - estimate $P(e|f)$ using the probabilities that a word in e gets translated into a word in f
  - these probabilities cannot be computed directly
    - unless we have word-aligned training data (but we don’t, for now)
- Better idea
  - introduce the alignment functions as latent variables to be estimated as well
  - word translations and alignments are related:

$P(e|f) = \sum_a P(e, a|f)$ but also $P(a|e, f) = P(e, a|f) \,/\, P(e|f)$

(applying the chain rule)

Use of the EM algorithm

(Dempster, Laird, Rubin 1977)

- Expectation maximization
  - iterative method to estimate model parameters together with latent variables which have reciprocal dependencies
  - applicable when
    - equations for parameters and variables cannot be solved directly
    - derivative of likelihood function (of parameters) cannot be obtained
- Iterate the E and M steps
  - **E**xpectation: using current estimate of parameters, compute a likelihood function over latent variables (and parameters)
  - **M**aximization: re-compute that parameters so that they maximize the likelihood that was found
- Initialization: e.g. with uniform probabilities for parameters

EM applied to IBM Model 1

- Reminder:

  $P(e|f) = \sum_a P(e, a|f)$ and $P(a|e, f) = P(e, a|f) / P(e|f)$

- We can make $P(e, a|f)$ more explicit
  - assume it depends only on word translation probabilities $P(e_j|f_i)$, noted $t(.|.)$ → "Model 1"
  - assume these are independent
  - note E and F lengths in words of sentences e and f
  - then: $P(e, a|f) = (\varepsilon \prod_{j=1..E} t(e_j|f_{a(j)})) / (F + 1)^E$
    - ε is a normalization factor

Making $P(e|f)$ more explicit (E step)

$P(e|f) = \sum_a P(e, a|f) =$

$= \sum_{a(1)=0..F} \dots \sum_{a(E)=0..F} P(e, a|f)$

$= \sum_{a(1)=0..F} \dots \sum_{a(E)=0..F} (\varepsilon/(F + 1)^E)\prod_{j=1..E} t(e_j|f_{a(j)})$

$= (\varepsilon/(F + 1)^E) \sum_{a(1)=0..F} \dots \sum_{a(E)=0..F} \prod_{j=1..E} t(e_j|f_{a(j)})$

$= (\varepsilon/(F + 1)^E) \prod_{j=1..E} \sum_{i=0..F} t(e_j|f_i)$

Also: $P(a|e, f) = \prod_{j=1..E} (\ t(e_j|f_{a(j)}) \ / \sum_{i=0..F} t(e_j|f_i)\ )$

EM applied to IBM Model 1 (again)

- Better use the word-specific $t(.|.)$ values for EM rather than the full-sentence probability $P(e|f)$
  - they can be estimated directly using counts
  - they suffice to compute $P(e|f)$ given the above formulae
  - much fewer than $P(e|f)$ values when training on large corpora
- Making the EM steps more explicit

1. Start with uniform $t(e_j|f_i)$ values
   a. Expectation step: compute all $P(a|e, f)$ using all current $t(.|.)$ values
   b. Maximization step (of likelihood): update the $t(.|.)$ values (improve estimates) using counts and $P(a|e, f)$ values
2. Iterate (1) and (2) - with a guarantee of convergence!

Estimating $t(.|.)$ values using *counts*

- New notation: say *we* and *wf* are English and French *words*
- Define *count*(*we*, *wf*, *e*, *f*) as the number of times *wf* was translated into *we* according to all alignments of *e* and *f* weighed by their probability → **M** step:

  $t(we|wf) = (\sum_{(e,f)} count(we, wf, e, f)) \,/\, (\sum_{we} \sum_{(e,f)} count(we, wf, e, f))$

  (the denominator is the normalization factor)

- Collecting counts (δ(…, …) is Kronecker's function) → **E** step:

  $count(we, wf, e, f) = \sum_{\mathrm{a}} P(a|e, f) \sum_{j=1..E} \delta(we, e_j)\, \delta(wf, f_{a(j)})$

  which are expressed in terms of $t(.|.)$ using last line of slide 14:

  $= (t(we|wf) \sum_{j=1..E} \delta(we, e_j) \sum_{i=1..F} \delta(wf, f_i)) \,/\, (\sum_{i=0..F} t(we|f_i))$

A simple implementation
[from Koehn 2010, page 91]

```
initialize t(we|wf) uniformly
while <not converged>
    for all we, wf do
        count(we, wf)=0
        total(wf)=0
    end for
    for_1 all e, f do
        for_2 all we ∈ e do
            subtotal(we) = 0
            for_3 all wf ∈ f do
                subtotal(we) += t(we|wf)
            end for_3
        end for_2
        for_4 all we ∈ e do
            for_5 all wf ∈ f do
                count(we, wf) += t(we|wf)/
                        subtotal(we)
                total(wf) += t(we|wf)/
                        subtotal(we)
            end for_5
        end for_4
    end for_1
    for_6 all wf do
        for_7 all we do
            t(we|wf) = count(we|wf)/
                total(wf)
        end for_7
    end for_6
end while
```

When do we stop?

- Quality of a translation model: perplexity
  - if *s* are sentences from a given corpus

    $\log_2 PP = - \sum_s \log_2 P(e_s | f_s)$

- Iterate until *PP* stops decreasing (<converge>)

- IBM Model 1
  - EM guarantees that Model 1 will converge towards a global minimum of perplexity

IBM Model 2

- In Model 1, alignment probabilities were factored out from the formula for $P(e, a|f)$ and from EM

  We had: $P(e, a|f) = (\varepsilon / (F + 1)^E) \prod_{j=1..E} t(e_j|f_{a(j)}))$

  (the big cat → le gros chat) and

  (the big cat → chat gros le) have the same probability!

- Alignment probability distribution $P_a(i|j, E, F)$
  - probability that it is the French word in position i that corresponds to the English word in position j

Therefore, $P(e, a|f) = \varepsilon' \prod_{j=1..E} t(e_j|f_{a(j)})\, P_a(a(j)|j, E, F)$

IBM Model 2 (continued)

- Equations are transformed similarly to Model 1

  $P(e|f) = \varepsilon \prod_{j=1..E} \sum_{i=0..F} t(e_j|f_i)\, P_a(i|j, E, F)$

- Computation of (fractional) counts for word translations

  $count(we, wf, e, f) = \sum_{j=1..E} \sum_{i=1..F} \mathbf{(} t(we|wf)\, P_a(i|j, E, F)\, \delta(we, e_j)\, \delta(wf, f_i)$
  $/ (\sum_{k=0..F} t(\mathrm{we}|f_k)\, P_a(k|j, E, F))\mathbf{)}$

- Computation of counts for alignments

  $count_a\ (i, j, e, f) = t(e_j|f_j)\, P_a(i|j, E, F) / (\sum_{k=0..F} t(e_j|f_k)\, P_a(k|j, E, F))$

- Training algorithm similar to Model 1 + start with initial values of $t(e_j|f_j)$ obtained by some iterations of Model 1

IBM Models 1-3

- IBM Model 1: lexical translation
- IBM Model 2: added absolute alignment model
- IBM Model 3: added fertility and absolute distortion models
  - probability that one English word generates several French ones
  - insertion of NULL token with a fixed probability after each word
  - probability of distortion instead of alignment $P_d(j|i, E, F)$
  - exhaustive count collection no longer possible → sample the alignment space to find optimum by hill climbing, from Model 2

IBM Models 4 and 5

- IBM Model 4: adds relative distortion model
  - introduces notion of "cept" (sort of phrase)
  - models distortion based on the position of a word in a cept (e.g. initial or not), possibly also on word class (POS-based or empirically)

- IBM Model 5: deal with a deficiency
  - deficiency = alignments can have "superposed" words
  - avoid spreading probability over such alignments
  - less commonly used than Models 1-4

Using IBM Models for word alignment: for instance in the GIZA++ tool

- Results of IBM Models after EM algorithm = probabilities for lexical translation and alignment

- Can be used to determine the most probable word alignment for each sentence pair (= the "Viterbi alignment")
  - Model 1, for each word e_i select the most likely word f_j (using $t(e_i|f_j)$)
  - Model 2, same but maximize $t(e_i|f_j)\ P_a(j|i, E, F)$
  - Models 3-5, no closed form expression: start with Model 2, then use heuristics

- Many other methods exist for *word alignment*
  - generative: train HMMs on linking probabilities, then use Viterbi decoding or another dynamic programming method
  - discriminative: structured prediction, feature functions, etc.
  - still, for phrase-based translation, IBM Models 1-4 perform well

Improving word alignments: towards phrase-based translation models

- For a given translation direction, the IBM models can find one-to-one alignments, multiple-to-one, one-to-zero, but *never one-to-multiple*
  - still, for a correct alignment, we might need both
  - {*Paul*} {*was waiting*} {*inside*} $\leftrightarrow$ {*Paul*} {*attendait*} {*à l ' intérieur*}
- Solution: symmetrization, by running the algorithm in both directions
  - consider the intersection of the two sets of alignment points, or their union, or enrich intersection with some points of the union, etc.
- IBM Models are no longer used as translation models (to estimate $P(e|f)$) but only to produce (1) probabilities of word translation, and (2) word alignments that will help learning phrase-based TMs
  - lower IBM models are used as steps to learn higher ones (1 → 4)

Phrase-based translation models

- The goal remains the same
  - given a foreign (French) sentence *f*, look for the English sentence *e* which maximizes $P(e|f)$, with
    $\text{argmax}_e\, P(e|f) = \text{argmax}_e\, P_{TM}(f|e)\, P_{LM}(e)$
- But take a different approach to compute P_{TM} : consider each sentence *e* and *f* as made of phrases

  $e = \{\underline{e_1}, \ldots, \underline{e_i}, \ldots, \underline{e_M}\}$ and $f = \{\underline{f_1}, \ldots, \underline{f_j}, \ldots, \underline{f_N}\}$
  - phrases are non-empty ordered sets of contiguous words
  - phrases cover entirely the sentence

Note on the word `phrase'
- originally, a linguistic notion (noun phrase, verb phrase)
- here, just a set of words, no linguistic motivation
- in French: `groupe de mots', not `phrase'

Phrase-based translation probability

$P_{\mathrm{TM}}(f|e) = \Pi_{i=1..M} P(\underline{f}_i|\underline{e}_i) \cdot d(\textsc{start}(\underline{f}_i) - \textsc{end}(\underline{f}_{i-1}) - 1)$

$P(\underline{f}_i|\underline{e}_i)$ is the probability that $\underline{e}_i$ is translated into $\underline{f}_i$

d is a "distance-based reordering model"

START($\underline{f}_i$) is the position of the first word of phrase $\underline{f}_i$
END($\underline{f}_i$) is the position of the last word of phrase $\underline{f}_i$

- e.g,, if phrases $\underline{e}_{i\text{-}1}$ and $\underline{e}_i$ are translated in sequence by phrases $\underline{f}_{i-1}$ and $\underline{f}_i$, then START($\underline{f}_i$) = END($\underline{f}_{i-1}$)+1, and we have $d(0)$
- a simple and efficient function: $d(x) = \alpha^{|x|}$ (adjust α if needed)

How do we estimate $P(\underline{f}|\underline{e})$
i.e. the probabilities of phrase translations?

1. Consider aligned sentence pairs

   (can be computed, e.g. with the Gale and Church algorithm)

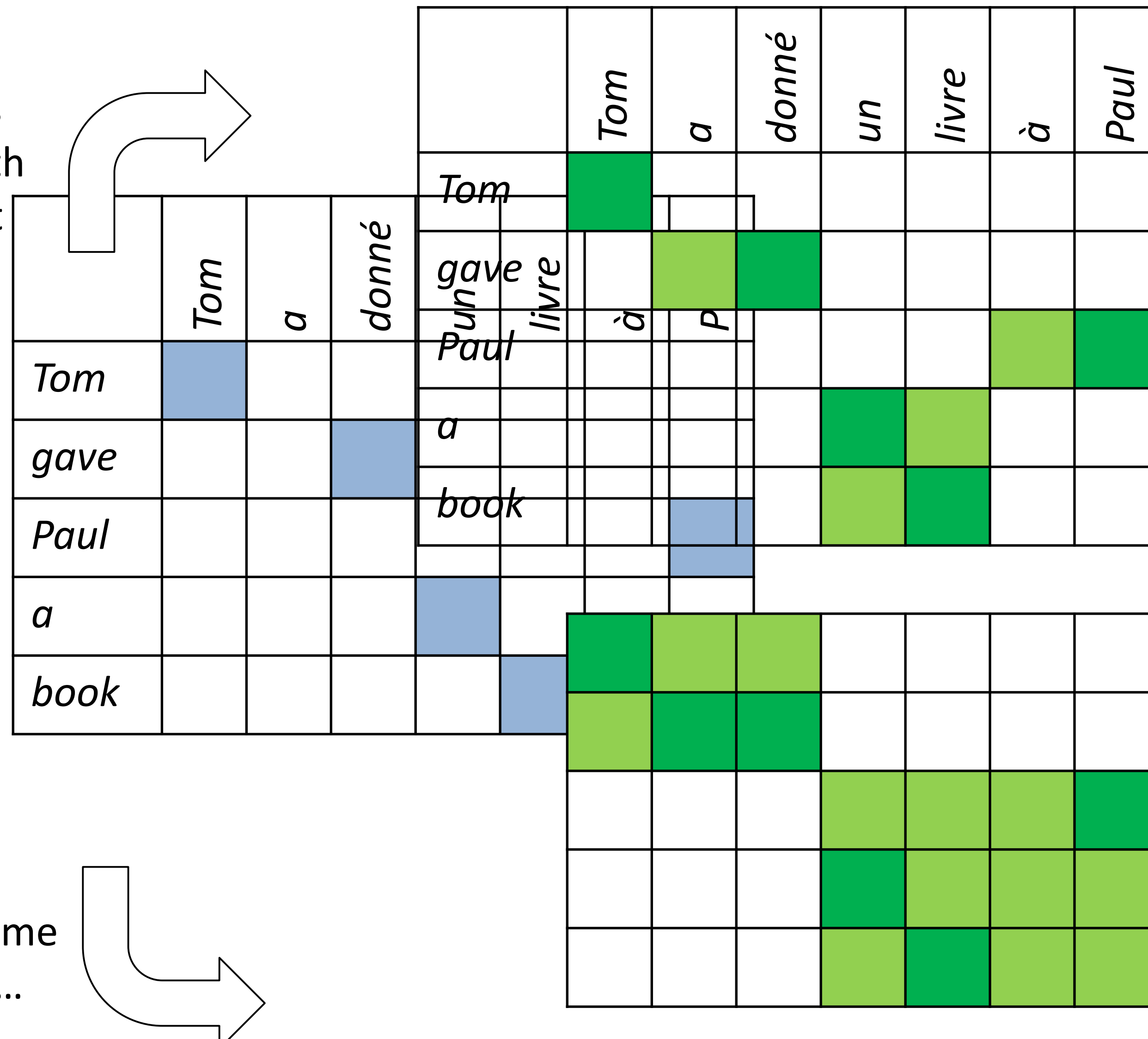
2. Perform word alignment for each pair
   – e.g. with the IBM Models

3. For each sentence pair, extract all phrase pairs that are "consistent" with the word alignment
   – possibly with some filtering, e.g. by length

4. Estimate *P* from counts of phrase pairs

Phrases consistent with alignment points

Some phrases consistent with the alignment



And some others…

Formal definitions

- Definition of "consistent with an alignment"
  - a phrase pair ($\underline{f}$, $\underline{e}$) is consistent with an alignment *A* *iff*
    - all words f_i from $\underline{f}$ that have alignment points have them with words in $\underline{e}$ (not outside $\underline{e}$): $\forall e_i \in \underline{e}, (e_i, f_j) \in A \Rightarrow f_j \in \underline{f}$
    - and vice-versa
    - and the pair includes at least one alignment point

- Definition of counts
  - extract all phrase pairs from the corpus
  - how many times each pair was extracted? count($\underline{e}$, $\underline{f}$)
  - then estimate: $P(\underline{f}|\underline{e}) = \text{count}(\underline{e}, \underline{f}) / \sum_{\underline{f}_i} \text{count}(\underline{e}, \underline{f}_i)$

Remarks

- Number of extracted phrases: quadratic in the number of words, for each sentence pair
  - approximation: $|e| \approx |f|$ and "correct" alignments
    - note: unaligned words generate many more phrases
  - advantage: a lot of phrases to choose from
  - disadvantage: large memory/disk requirements
  - solution: remove long phrases and/or those seen only once

- Comparison of phrase-based and IBM models
  - phrase-based have simpler formulation
  - but they require word alignments
  - so IBM models (or others) are still needed to find the Viterbi alignment for each sentence pair

Towards log-linear "translation models"

- Translate f = find e which maximizes the product of three terms
  - probabilities of inverse phrase translations $P_{\text{tm}}(\underline{f}_i | \underline{e}_i)$
  - reordering model for each phrase $d(\text{START}(\underline{f}_i) - \text{END}(\underline{f}_{i-1}) - 1)$
  - language model for each word $P_{\text{lm}}(e_k | e_1,\ldots, e_{k-1})$
- Terms can be weighted: no longer Bayesian, but more efficient
  - more components can be added + weights can be tuned
- So, now we maximize

$$\prod_{i=1..M} (P_{\text{tm}}(\underline{f}_i | \underline{e}_i)^{\lambda_{\text{tm}}} \cdot d(\text{START}(\underline{f}_i) - \text{END}(\underline{f}_{i-1}) - 1)^{\lambda_{\text{rm}}}) \cdot \prod_{k=1..|e|} P_{\text{lm}}(e_k | e_1,\ldots, e_{k-1})^{\lambda_{\text{lm}}}$$

which can be expressed as $\exp(\sum \lambda_i h_i(e))$ using $h(\ldots) = \log P(\ldots)$

- Translate f = find e (vector of features) that maximizes a weighted sum of feature functions (trained separately)

Extensions to the model

- Additional useful factors – determined empirically
  1. Direct translation probabilities (use both $e|f$ and $f|e$)
  2. Lexical weighting of phrase pairs
     - re-estimate likelihood of a pair based on the translation probabilities of the words that compose it
     - because rare phrases might get high $P_{\text{tm}}(\underline{f}_i|\underline{e}_i)$ scores
  3. Word penalty: multiply by ω for each word
     - $\omega < 1$ favors shorter translations, $\omega > 1$ favors longer ones
     - tune the value of ω
  4. Phrase penalty: ρ factor, similar to ω
- Reordering model can be improved
- Phrase-based models can be trained directly using EM
  - but results are not better than the word-based approach

Conclusions

- Many methods for translation modeling
- We presented two principal approaches
  - IBM Models and Phrase-based
- We showed the importance of word alignment

- Many variants and extensions exist
- More complex models: syntax-based = hierarchical
  - must learn tree-based translation models

- Missing elements:
  - LANGUAGE MODELING, or how to estimate $P(e)$
  - DECODING, or how to search for e

References

- Philipp Koehn, *Statistical Machine Translation*, Cambridge University Press, 2010
  - chapters 4 and 5
- Jörg Tiedemann, *Bitext Alignment*, Morgan & Claypool, 2011

- P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer, "The mathematics of statistical machine translation: parameter estimation", *Computational Linguistics*, 19(2), p. 263-311, 1993
  - introduced IBM Models 1-5
- Franz Josef Och and Hermann Ney, "A Systematic Comparison of Various Statistical Alignment Models", *Computational Linguistics*, 29(1), p. 19-51, 2003
  - review of past state-of-the-art algorithms for word alignment

- Dempster, A.P. and Laird, N.M. and Rubin, D.B., "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society Series B*, p. 1-38, 1977.
  - introduced the EM algorithm

Practical work

- Install the Moses MT system, build a phrase-based translation model
  - Sections 1, 2, 4 (up to 4.2 included) of ‘TP-MT-instructions’
  - optionally: Section 3 to verify that Moses works
- Goal: train Moses on a domain or language pair of your own, examine the translation models (size and “perceived quality”): **4.2**