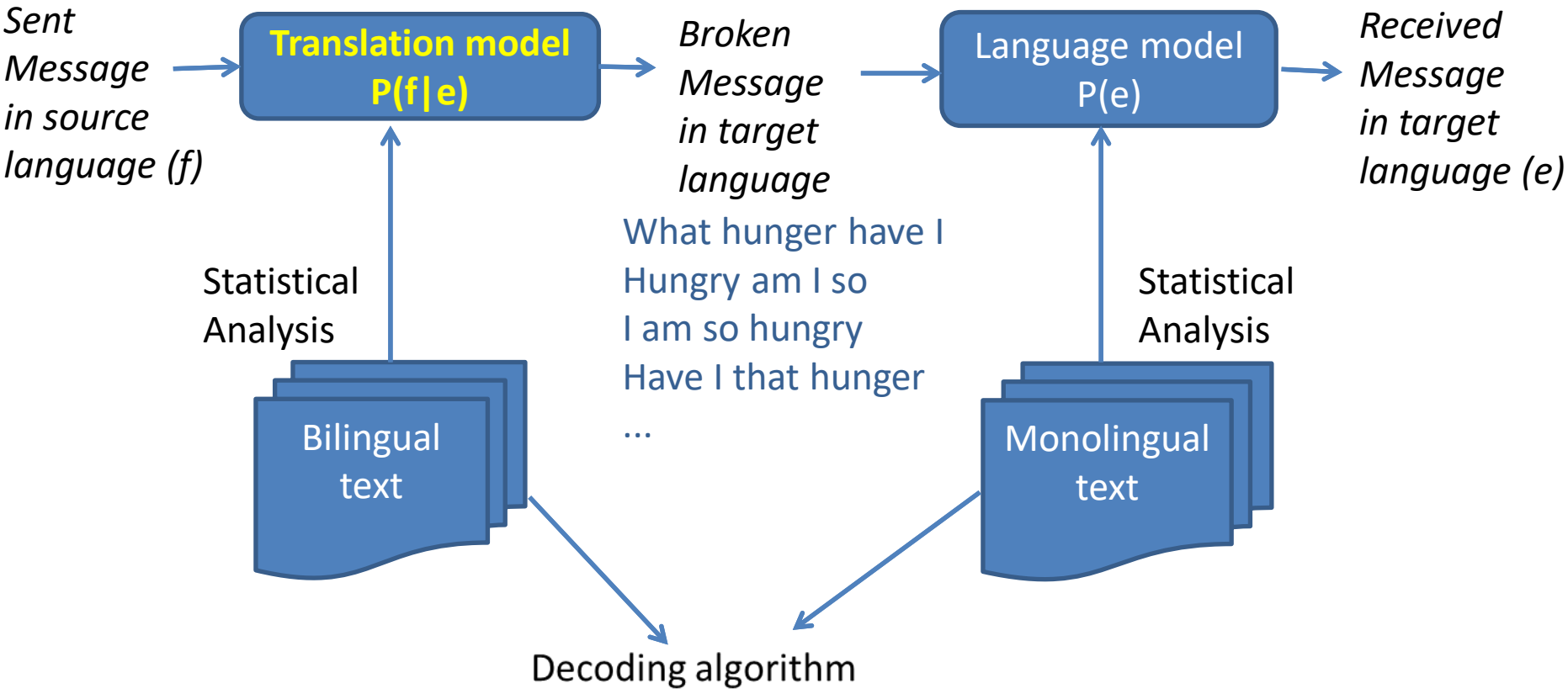
Machine Translation
Course 4

Diana Trandabăț

Academic year: 2022-2023

Noisy Channel Model



$$\hat{e} = argmax_e\, P(e|f) = argmax_e \frac{P(f|e) * P(e)}{P(f)} = argmax_e P(f|e) * P(e)$$

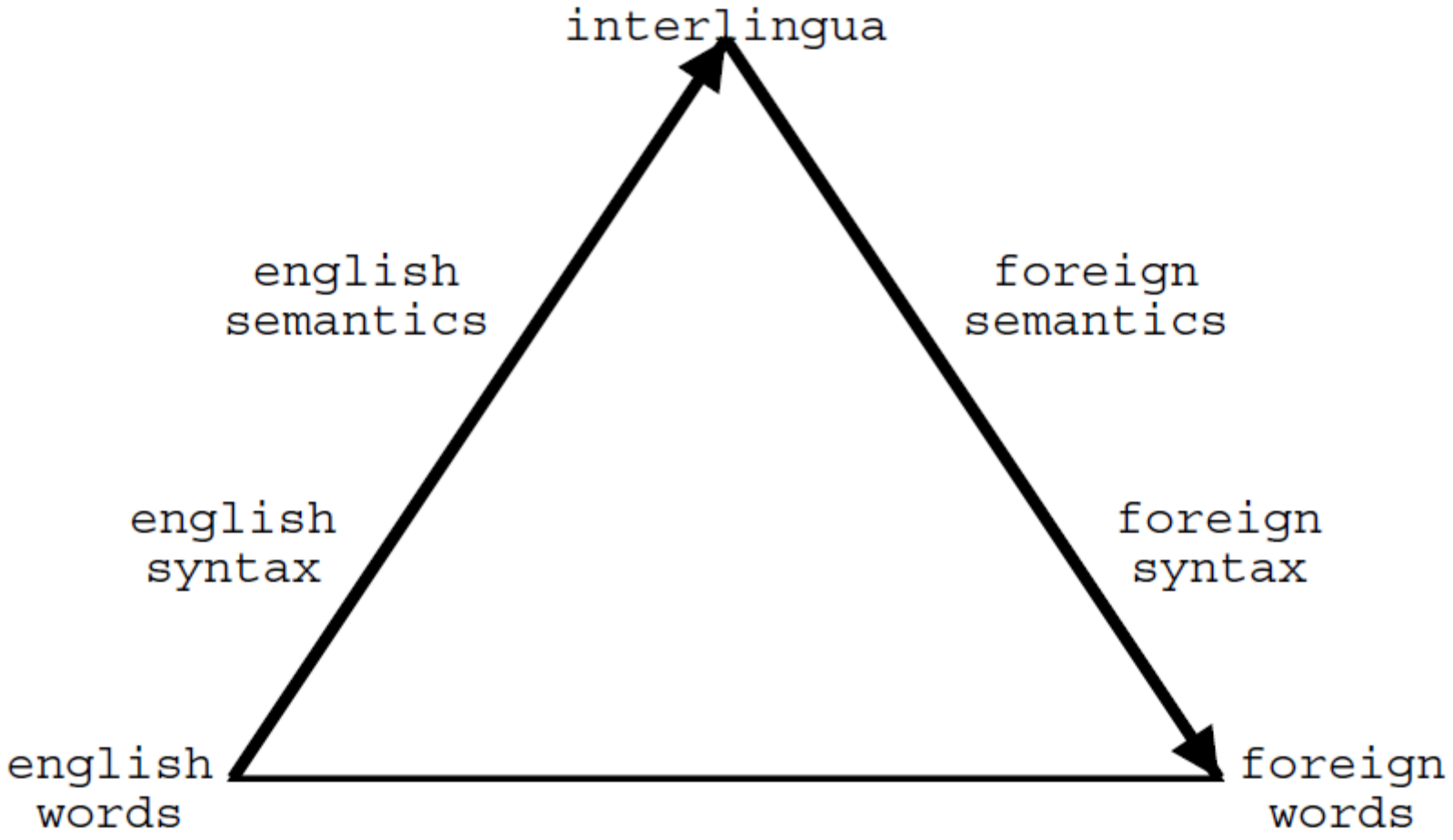
Ce foame am

I am so hungry

Translation model

- Machine Translation pyramid
- Statistical modeling and IBM Models
- EM algorithm
- Word alignment
- Flaws of word-based translation
- Phrase-based translation
- Syntax-based translation

The Machine Translation Pyramid



Sentence Alignment

- If document D_e is translation of document D_f how do we find the translation for each sentence?
- The *n*-th sentence in D_e is not necessarily the translation of the *n*-th sentence in document D_f
- In addition to 1:1 alignments, there are also 1:0, 0:1, 1:n, and n:1 alignments
- Approximately 90% of the sentence alignments are 1:1

Sentence Alignment (c'ntd)

- There are several sentence alignment algorithms:
  - Align (Gale & Church): Aligns sentences based on their character length (shorter sentences tend to have shorter translations then longer sentences). Works astonishingly well
  - Char-align: (Church): Aligns based on shared character sequences. Works fine for similar languages or technical domains
  - K-Vec (Fung & Church): Induces a translation lexicon from the parallel texts based on the distribution of foreign-English word pairs.

Word-Level Alignments

- Given a parallel sentence pair we can link (align) words or phrases that are translations of each other:

Word level alignment

- Question: How many possible alignments are there for a given sentence in source language *e* and its translation in target language *f*, where |e| = l and |f|=m?
- Answer
  - Each foreign word can align with any one of |e|= l words, or it can remain unaligned
  - Each foreign word has (l + 1) choices for an alignment, and there are |f| = m foreign words
  - So, there are $(l+1)^m$ alignments for a given *e* and *f*

Word level alignment

- Question: If all alignments are equally likely, what is the probability of any one alignment, given *e*
- Answer:
  - $P(A|e) = p(|f|=m) * 1/(l+1)$
  - If we assume that $p(|f|= m)$ is uniform over all possible values of $|f|$, then we can let $p(|f|= m) = C$
- $P(A|e) = C /(l+1)^m$

Computing Translation Probabilities

- Given a parallel corpus we can estimate P(e|f)
- The maximum likelihood estimation of P(e|f) is: freq(e,f)/freq(f)
- Way too specific to get any reasonable frequencies! Vast majority of unseen data will have zero counts!
- P(e | f) could be re-defined as:

$$P(e \mid f) = \prod_{f_j} \max_{e_i} P(e_i \mid f_j)$$

- Problem: The English words maximizing P(e|f) might not result in a readable sentence

Computing Translation Probabilities (c'tnd)

- We can account for adequacy: each foreign word translates into its most likely English word
- We cannot guarantee that this will result in a fluent English sentence
- Solution: transform P(e | f) with Bayes' rule:

$P(e \mid f) = P(f \mid e)\, P(e) / P(f)$

- P(f | e) accounts for adequacy
- P(e) accounts for fluency

Accurate vs. Fluent

- Often impossible to have a true translation; one that is:
  - **Faithful** to the source language, and
  - **Fluent** in the target language
  - Ex:
    - Japanese: *"fukaku hansei shite orimasu"*
    - Fluent translation: *"we apologize"*
    - Faithful translation: *"we are deeply reflecting (on our past behaviour, and what we did wrong, and how to avoid the problem next time)"*

- So need to compromise between faithfulness & fluency

IBM Models 1–5

- Model 1: Bag of words
  - Unique local maxima
  - Efficient EM algorithm (Model 1–2)
- Model 2: General alignment: $a(e_{pos} \mid f_{pos}, e_{length}, f_{length})$
- Model 3: fertility: n(k | e)
  - No full EM, count only neighbors (Model 3–5)
  - Deficient (Model 3–4)
- Model 4: Relative distortion, word classes
- Model 5: Extra variables to avoid deficiency

IBM Model 1

- Given an English sentence $e_1 \ldots e_l$ and a foreign sentence $f_1 \ldots f_m$
- We want to find the 'best' alignment *a*, where *a* is a set pairs of the form *{(i , j), . . . , (i', j')}*,

  0 <= i , i' <= l and *1 <= j , j' <= m*
- Note that if *(i , j), (i', j)* are in *a*, then *i* equals *i'*, i.e. no many-to-one alignments are allowed
- Note we add a spurious NULL word to the English sentence at position 0
- In total there are $(l + 1)^m$ different alignments
- Allowing for many-to-many alignments results in $(2^l)^m$ possible alignments

IBM Model 1

- Simplest of the IBM models
- Does not consider word order (bag-of-words approach)
- Does not model one-to-many alignments
- Computationally inexpensive
- Useful for parameter estimations that are passed on to more elaborate models

IBM Model 1

- Translation probability in terms of alignments:

where:

$$P(f \mid e) = \sum_{a \in A} P(f, a \mid e)$$

$$P(f, a \mid e) = P(a \mid e) \cdot P(f \mid a, e)$$

$$= \frac{1}{(l+1)^m} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

and:

$$P(f \mid e) = \sum_{a \in A} \frac{1}{(l+1)^m} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

IBM Model 1

- We want to find the most likely alignment:

$$\arg\max_{a \in A} \frac{1}{(l+1)^m} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

- Since P(a | e) is the same for all a:

$$\arg\max_{a \in A} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

- Problem: We still have to enumerate all alignments

IBM Model 1

- Since $P(f_j \mid e_i)$ is independent from $P(f_{j'} \mid e_{i'})$ we can find the maximum alignment by looking at the individual translation probabilities only
- Let $\underset{a \in A}{\arg\max} = (a_1, \ldots, a_m)$, then for each a_j:

$$a_j = \underset{0 \le i \le l}{\arg\max} P(f_j \mid e_i)$$

- The best alignment can be computed in a quadratic number of steps: (l+1 x m)

Computing Model 1 Parameters

- How to compute translation probabilities for model 1 from a parallel corpus?
- Step 1: Determine candidates. For each English word *e* collect all foreign words *f* that co-occur at least once with *e*
- Step 2: Initialize P(f | e) uniformly, i.e. P(f | e) = 1/(no of co-occurring foreign words in the entire corpus)

Computing Model 1 Parameters

- Step 3: Iteratively refine translation probabilities:

```
1   for n iterations
2       set tc to zero
3       for each sentence pair (e,f) of lengths (l,m)
4           for j=1 to m
5               total=0;
6               for i=1 to l
7                   total += P(f_j | e_i);
8               for i=1 to l
9                   tc(f_j | e_i) += P(f_j | e_i)/total;
10      for each word e
11          total=0;
12          for each word f s.t. tc(f | e) is defined
13              total += tc(f | e);
14          for each word f s.t. tc(f | e) is defined
15              P(f | e) = tc(f | e)/total;
```

IBM Model 1 Example

- Parallel ‘corpus’:

  the dog :: le chien

  the cat :: le chat

- Step 1+2 (collect candidates and initialize uniformly):

  P(le | the) = P(chien | the) = P(chat | the) = 1/3

  P(le | dog) = P(chien | dog) = 1/2

  P(le | cat) = P(chat | cat) = 1/2

  P(le | NULL) = P(chien | NULL) = P(chat | NULL) = 1/3

IBM Model 1 Example

- Step 3: Iterate
- NULL the dog :: le chien
  - j=1

    total = P(le | NULL) + P(le | the) + P(le | dog) = 1/3+1/3+1/2 = 1.16

    tc(le | NULL) += P(le | NULL)/total= 0 += .333/1.16 = 0.28

    tc(le | the) += P(le | the)/1 = 0 += .333/1.16 = 0.28

    tc(le | dog) += P(le | dog)/1 = 0 += .5/1.16 = 0.43
  - j=2

    total = P(chien | NULL)+P(chien | the)+P(chien | dog)=1/3+1/3+1/2=1.16

    tc(chien | NULL) += P(chien | NULL)/1 = 0 += .333/1.16 = 0.28

    tc(chien | the) += P(chien | the)/1 = 0 += .333/1.16 = 0.28

    tc(chien | dog) += P(chien | dog)/1 = 0 += .5/1.16 = 0.43

IBM Model 1 Example

- NULL the cat :: le chat
  - j=1

    total = P(le | NULL)+P(le | the)+P(le | cat)=1/3+1/3+1/2=1.16

    tc(le | NULL) += P(le | NULL)/1 = 0.28 += .333/1.16 = 0.56

    tc(le | the) += P(le | the)/1 = 0.28 += .333/1.16 = 0.56

    tc(le | cat) += P(le | cat)/1 = 0 +=.5/1.16 = 0.43
  - j=2

    total = P(chat | NULL)+P(chat | the)+P(chat | cat)=1/3+1/3+1/2=1.16

    tc(chat | NULL) += P(chat | NULL)/1 = 0 += .333/1.16 = 0.28

    tc(chat | the) += P(chat | the)/1 = 0 += .333/1.16 = 0.28

    tc(chat | cat) += P(chat | cat)/1 = 0 += .5/1 = 0.43

IBM Model 1 Example

- Re-compute translation probabilities
  - total(the) = tc(le | the) + tc(chien | the) + tc(chat | the)
    = 0.56 + 0.28 + 0.28= 1.12

    P(le | the) = tc(le | the)/total(the)
    = 0.56 / 1.12 = 0.5

    P(chien | the) = tc(chien | the)/total(the)
    = 0.28/1.12 = 0.25

    P(chat | the) = tc(chat | the)/total(the)
    = 0.28/1.12 = 0.25
  - total(dog) = tc(le | dog) + tc(chien | dog) = 0.43+0.43=0.86

    P(le | dog) = tc(le | dog)/total(dog)
    = 0.43 / 0.86 = 0.5

    P(chien | dog) = tc(chien | dog)/total(dog)
    = 0.43 / 0.86 = 0.5

IBM Model 1 Example

- total(cat) = tc(le | cat) + tc(chat | cat) = 0.43+0.43=0.86

  P(le | cat) = tc(le | cat)/total(cat)

  = 0.43 / 0.86 = 0.5

  P(chat | cat) = tc(chat | cat)/total(cat)

  = 0.43 / 0.86 = 0.5

- total(NULL) = tc(le | NULL) + tc(chien | NULL) + tc(chat| NULL) = 0.56+0.28+0.28 = 0.12

  P(le | NULL) = tc(le | NULL)/total(NULL)

  = 0.56 / 1,12= 0.5

  P(chien | NULL) = tc(chien | NULL)/total(NULL)

  = 0.28 / 1.12= 0.25

  P(chat| NULL) = tc(chat | NULL)/total(NULL)

  = 0.28/ 1.12 = 0.25

IBM Model 1 Example

- Iteration 2:
- NULL the dog :: le chien
  - j=1

    total = P(le | NULL)+P(le | the)+P(le | dog)= 1.5

    = 0.5 + 0.5 + 0.5 = 1.5

    tc(le | NULL) += P(le | NULL)/1.5 = 0 += .5/1.5 = 0.333

    tc(le | the) += P(le | the)/1 = 0 += .5/1.5 = 0.333

    tc(le | dog) += P(le | dog)/1 = 0 += .5/1.5 = 0.333
  - j=2

    total = P(chien | NULL)+P(chien | the)+P(chien | dog)=1

    = 0.25 + 0.25 + 0.5 = 1

    tc(chien | NULL) += P(chien | NULL)/1 = 0 += .25/1 = 0.25

    tc(chien | the) += P(chien | the)/1 = 0 += .25/1 = 0.25

    tc(chien | dog) += P(chien | dog)/1 = 0 += .5/1 = 0.5

IBM Model 1 Example

- NULL the cat :: le chat
  - j=1 (le)

    total = P(le | NULL)+P(le | the)+P(le | cat)= 1.5

    = 0.5 + 0.5 + 0.5 = 1.5

    tc(le | NULL) += P(le | NULL)/1 = 0.333 += .5/1.5 = 0.66

    tc(le | the) += P(le | the)/1 = 0.333 += .5/1.5 = 0.66

    tc(le | cat) += P(le | cat)/1 = 0 += .5/1.5 = 0.33
  - j=2

    total = P(chat | NULL)+P(chat | the)+P(chat | cat)=1

    = 0.25 + 0.25 + 0.5 = 1

    tc(chat | NULL) += P(chat | NULL)/1 = 0 += .25/1 = 0.25

    tc(chat | the) += P(chat | the)/1 = 0 += .25/1 = 0.25

    tc(chat | cat) += P(chat | cat)/1 = 0 += .5/1 = 0.5

IBM Model 1 Example

- Re-compute translations (iteration 2):
  - total(the) = tc(le | the) + tc(chien | the) + tc(chat | the)
    = 0.66 + 0.25 + 0.25 = 1.16

    P(le | the) = tc(le | the)/total(the)
    = .66 / 1.16 = 0.56

    P(chien | the) = tc(chien | the)/total(the)
    = 0.25/1.16 = 0.188

    P(chat | the) = tc(chat | the)/total(the)
    = 0.25/1.16 = 0.188
  - total(dog) = tc(le | dog) + tc(chien | dog)
    = 0.333 + 0.5 = 0.833

    P(le | dog) = tc(le | dog)/total(dog)
    = 0.333 / 0.833 = 0.4

    P(chien | dog) = tc(chien | dog)/total(dog)
    = 0.5 / 0.833 = 0.6

  …….

IBM Model 1Example

- After 5 iterations:

P(le | NULL) = 0.755608028335301
P(chien | NULL) = 0.122195985832349
P(chat | NULL) = 0.122195985832349
P(le | the) = 0.755608028335301
P(chien | the) = 0.122195985832349
P(chat | the) = 0.122195985832349
P(le | dog) = 0.161943319838057
P(chien | dog) = 0.838056680161943
P(le | cat) = 0.161943319838057
P(chat | cat) = 0.838056680161943

IBM Model 1 Recap

- IBM Model 1 allows for an efficient computation of translation probabilities
- No notion of fertility, i.e., it's possible that the same English word is the best translation for all foreign words
- No positional information, i.e., depending on the language pair, there might be a tendency that words occurring at the beginning of the English sentence are more likely to align to words at the beginning of the foreign sentence

IBM Model 3

- IBM Model 3 offers two additional features compared to IBM Model 1:
  - How likely is an English word e to align to k foreign words (fertility)?
  - Positional information (distortion), how likely is a word in position i to align to a word in position j?

IBM Model 3: Fertility

- The best Model 1 alignment could be that a single English word aligns to all foreign words
- This is clearly not desirable and we want to constrain the number of words an English word can align to
- Fertility models a probability distribution that word e aligns to k words: n(k,e)
- Consequence: translation probabilities cannot be computed independently of each other anymore
- IBM Model 3 has to work with full alignments, note there are up to $(l+1)^m$ different alignments

IBM Model 1 + Model 3

- Iterating over all possible alignments is computationally infeasible
- Solution: Compute the best alignment with Model 1 and change some of the alignments to generate a set of likely alignments (pegging)
- Model 3 takes this restricted set of alignments as input

IBM model 2

- Model parameters:
  - $T(f_j \mid e_{aj})$ translation probability of foreign word f_j given English word e_{aj} that generated it
  - $d(i|j,l,m)$ distortion probability, or probability that f_j is aligned to e_i , given l and m

IBM Model 3: Distortion

- The distortion factor determines how likely it is that an English word in position i aligns to a foreign word in position j, given the lengths of both sentences:

$$d(j \mid i, l, m)$$

- Note, positions are absolute positions

Deficiency

- Problem with IBM Model 3: It assigns probability mass to impossible strings
  - Well formed string: “This is possible”
  - Ill-formed but possible string: “This possible is”
  - Impossible string: 
- Impossible strings are due to distortion values that generate different words at the same position
- Impossible strings can still be filtered out in later stages of the translation process

Limitations of IBM Models

- Only 1-to-N word mapping
- Handling fertility-zero words (difficult for decoding)
- Almost no syntactic information
  - Word classes
  - Relative distortion
- Long-distance word movement
- Fluency of the output depends entirely on the English language model

Decoding

- How to translate new sentences?
- A decoder uses the parameters learned on a parallel corpus
  - Translation probabilities
  - Fertilities
  - Distortions
- In combination with a language model the decoder generates the most likely translation
- Standard algorithms can be used to explore the search space (A*, greedy searching, …)
- Similar to the traveling salesman problem

“Silence is the language of god, everything else is poor translation.”

Rumi