Statistical Machine Translation
Lecture 3
Word Alignment and Phrase Models

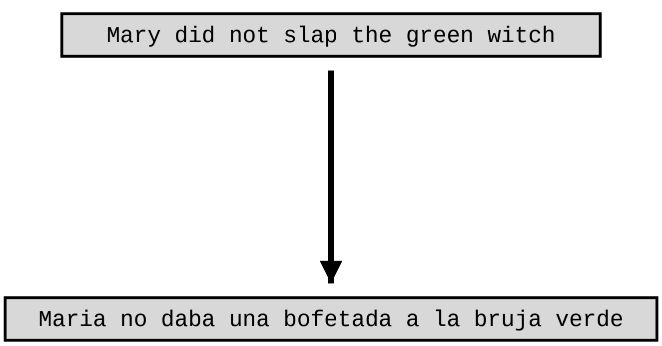
Philipp Koehn
pkoehn@inf.ed.ac.uk

School of Informatics
University of Edinburgh

Overview

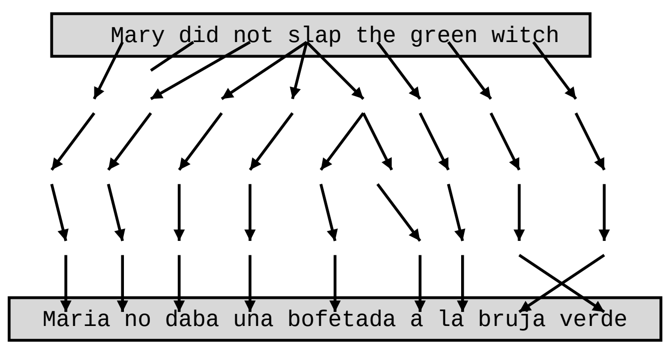
- Statistical modeling
- EM algorithm
- Improved word alignment
- Phrase-based SMT

Statistical Modeling



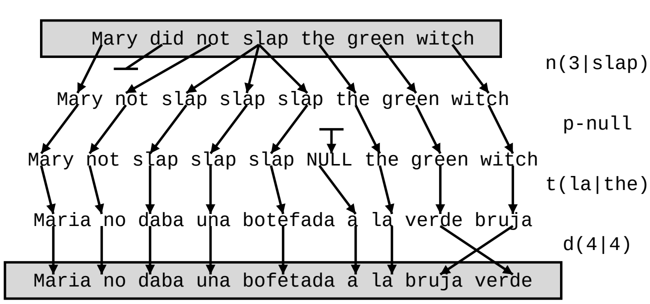
- Learn $P(f|e)$ from a parallel corpus
- Not sufficient data to estimate $P(f|e)$ directly

Statistical Modeling (2)



- Break the process into smaller steps

Statistical Modeling (3)



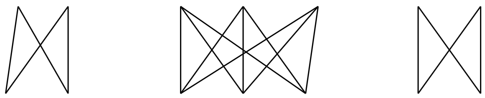
- Probabilities for smaller steps can be learned

Statistical Modeling (4)

- Generate a story how an English string e gets to be a foreign string f
  - choices in story are decided by reference to parameters
  - e.g., $p(\text{bruja}|\text{witch})$
- Formula for $P(f|e)$ in terms of parameters
  - usually long and hairy, but mechanical to extract from the story
- Training to obtain parameter estimates from possibly incomplete data
  - off-the-shelf EM

Parallel Corpora

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



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

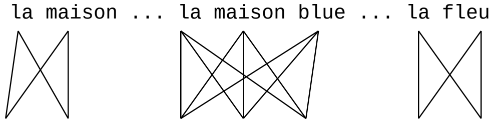
- Incomplete data
  - English and foreign words, but no connections between them
- Chicken and egg problem
  - if we had the connections, we could estimate the parameters of our generative story
  - if we had the parameters, we could estimate the connections

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
- EM in a nutshell
  - initialize model parameters (e.g. uniform)
  - assign probabilities to the missing data
  - estimate model parameters from completed data
  - iterate

EM Algorithm (2)

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

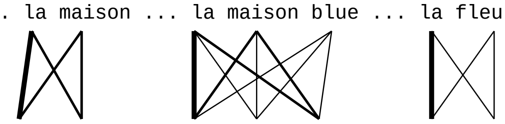


... the house ... the blue house ... the flower ...

- Initial step: all connections equally likely
- Model learns that, e.g., la is often connected with the

EM Algorithm (3)

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

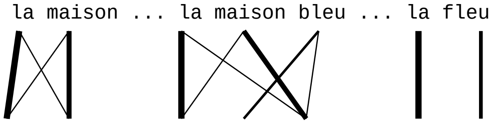


... the house ... the blue house ... the flower ...

- After one iteration
- Connections, e.g., between la and the are more likely

EM Algorithm (4)

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



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

- After another iteration
- It becomes apparent that connections, e.g., between fleur and flower are more likely (pigeon hole principle)

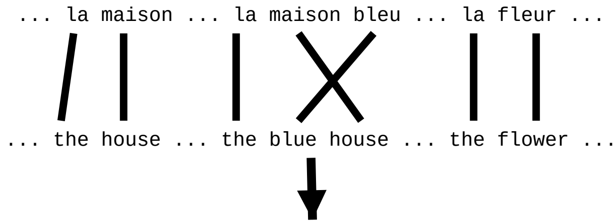
EM Algorithm (5)

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

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

- Convergence
- Inherent hidden structure revealed by EM

EM Algorithm (6)




```
p(la|the) = 0.453
p(le|the) = 0.334
p(maison|house) = 0.876
p(bleu|blue) = 0.563
...
```

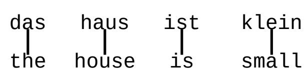
- Parameter estimation from the connected corpus

IBM Model 1

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

- What is going on?
  - foreign sentence **f** = $f_1 ... f_m$
  - English sentence **e** = $e_1 ... e_l$
  - each English word e_j is generated by a English word $f_{a(j)}$, as defined by the alignment function a, with the probabilty t
  - the normalization factor ϵ is required to turn the formula into a proper probability function

One example



das

e	$t(e\|f)$
the	0.7
that	0.15
which	0.075
who	0.05
this	0.025

Haus

e	$t(e\|f)$
house	0.8
building	0.16
home	0.02
household	0.015
shell	0.005

ist

e	$t(e\|f)$
is	0.8
's	0.16
?	0.02
?	0.015
?	0.005

klein

e	$t(e\|f)$
small	0.4
little	0.4
short	0.1
minor	0.06
petty	0.04

$$\begin{aligned} p(e,a|f) &= \frac{\epsilon}{4^3} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \\ &\quad \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein}) \\ &= \frac{\epsilon}{4^3} \times 0.7 \times 0.8 \times 0.8 \times 0.4 \\ &= 0.0256\epsilon \end{aligned}$$

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

IBM Model 1 and EM

- We need to be able to compute:
  - Expectation-Step: probability of alignments
  - Maximization-Step: count collection

IBM Model 1 and EM: Expectation Step

- We need to compute $p(\mathbf{a}|\mathbf{e},\mathbf{f})$
- Applying the chain rule:

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

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

IBM Model 1 and EM: Expectation Step

- We need to compute $p(\mathbf{e}|\mathbf{f})$

$$
\begin{aligned}
p(\mathbf{e}|\mathbf{f}) &= \sum_{\mathbf{a}} p(\mathbf{e},\mathbf{a}|\mathbf{f}) \\
&= \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} p(\mathbf{e},\mathbf{a}|\mathbf{f}) \\
&= \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(e_j|f_{a(j)}) \\
&= \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(e_j|f_{a(j)}) \\
&= \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(e_j|f_i)
\end{aligned}
$$

- Note the trick in the last line
  - removes the need for an exponential number of products
  - $\rightarrow$ this makes IBM Model 1 estimation tractable

IBM Model 1 and EM: Expectation Step

- Combine what we have:

$$
\begin{aligned}
p(\mathbf{a}|\mathbf{e},\mathbf{f}) &= p(\mathbf{e},\mathbf{a}|\mathbf{f})/p(\mathbf{e}|\mathbf{f}) \\
&= \frac{\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m} t(e_j|f_{a(j)})}{\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l} t(e_j|f_i)} \\
&= \frac{\prod_{j=1}^{m} t(e_j|f_{a(j)})}{\prod_{j=1}^{m}\sum_{i=0}^{l} t(e_j|f_i)} \\
&= \prod_{j=1}^{m} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l} t(e_j|f_i)}
\end{aligned}
$$

IBM Model 1 and EM: Maximization Step

- Now we have to collect counts
- Evidence from a sentence pair **e,f** that word e is a translation of word f:

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

- With the same simplication as before:

$$c(e|f;\mathbf{e},\mathbf{f}) = \frac{t(e|f)}{\sum_l t(e|f_{a(j)})} \sum_{j=1}^{m} \delta(e,e_j) \sum_{i=0}^{l} \delta(f,f_i)$$

IBM Model 1 and EM: Maximization Step

- After collecting these counts over a corpus, we can estimate the model:

$$t(e|f;\mathbf{e},\mathbf{f}) = \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f}))}{\sum_f \sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f}))}$$

IBM Model 1 and EM: Pseudocode

```
initialize t(e|f) uniformly
do
  set count(e|f) to 0 for all e,f
  set total(f) to 0 for all f
  for all sentence pairs (e_s,f_s)
    for all unique words e in e_s
      n_e = count of e in e_s
      total_s = 0
      for all unique words f in f_s
        total_s += t(e|f) * n_e
      for all unique words f in f_s
        n_f = count of f in f_s
        count(e|f) += t(e|f) * n_e * n_f / total_s
        total(f)   += t(e|f) * n_e * n_f / total_s
  for all f in domain( total(.) )
    for all e in domain( count(.|f) )
      t(e|f) = count(e|f) / total(f)
until convergence
```


Higher IBM Models

IBM Model 1	lexical translation
IBM Model 2	adds absolute reordering model
IBM Model 3	adds fertility model
IBM Model 4	relative reordering model
IBM Model 5	fixes deficiency

- Compuationally biggest change in Model 3
  - trick to simplify estimation does not work anymore
  - $\rightarrow$ exhaustive count collection becomes computationally too expensive
  - sampling over high probability alignments is used instead

Flaws of Word-Based MT

- Multiple English words for one German word

one-to-many problem: `Zeitmangel` → `lack of time`

German:	Zeitmangel	erschwert	das	Problem	.
Gloss:	LACK OF TIME	MAKES MORE DIFFICULT	THE	PROBLEM	.
Correct translation:	Lack of time makes the problem more difficult.				
MT output:	Time makes the problem .				

- Phrasal translation

non-compositional phrase: `erübrigt sich` → `there is no point in`

German:	Eine	Diskussion	erübrigt	sich	demnach
Gloss:	A	DISCUSSION	IS MADE UNNECESSARY	ITSELF	THEREFORE
Correct translation:	Therefore, there is no point in a discussion.				
MT output:	A debate turned therefore .				

Flaws of Word-Based MT (2)

- Syntactic transformations

reordering, genitive NP: `der Sache` → `for this matter`

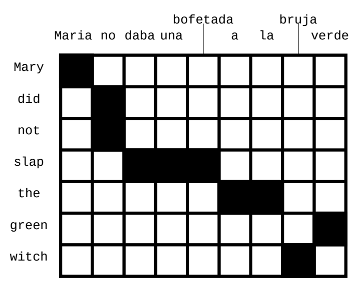
German:	Das	ist	der	Sache	nicht	angemessen	.
Gloss:	THAT	IS	THE	MATTER	NOT	APPROPRIATE	.
Correct translation:	That is not appropriate for this matter .						
MT output:	That is the thing is not appropriate .						

object/subject reordering

German:	Den	Vorschlag	lehnt	die	Kommission	ab	.
Gloss:	THE	PROPOSAL	REJECTS	THE	COMMISSION	OFF	.
Correct translation:	The commission rejects the proposal .						
MT output:	The proposal rejects the commission .						

Word Alignment

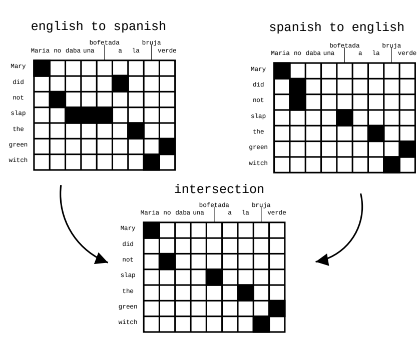
- Notion of word alignments valuable
- Trained humans can achieve high agreement
- Shared task at NAACL 2003 and ACL 2005 workshops



Word Alignment with IBM Models

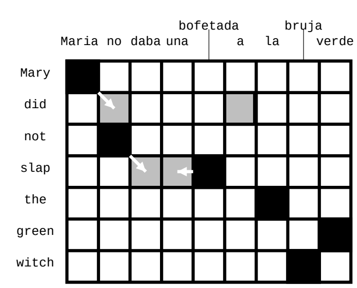
- IBM Models create a many-to-one mapping
  - words are aligned using an alignment function
  - a function may return the same value for different input (one-to-many mapping)
  - a function can not return multiple values for one input (no many-to-one mapping)
- But we need many-to-many mappings

Improved Word Alignments



- Intersection of GIZA++ bidirectional alignments

Improved Word Alignments (2)



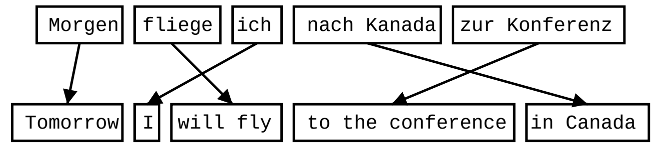
- Grow additional alignment points [Och and Ney, CompLing2003]

Growing Heuristic

```
GROW-DIAG-FINAL(e2f,f2e):
  neighboring = ((-1,0),(0,-1),(1,0),(0,1),(-1,-1),(-1,1),(1,-1),(1,1))
  alignment = intersect(e2f,f2e);
  GROW-DIAG(); FINAL(e2f); FINAL(f2e);

GROW-DIAG():
  iterate until no new points added
    for english word e = 0 ... en
      for foreign word f = 0 ... fn
        if ( e aligned with f )
          for each neighboring point ( e-new, f-new ):
            if ( ( e-new not aligned and f-new not aligned ) and
                 ( e-new, f-new ) in union( e2f, f2e ) )
              add alignment point ( e-new, f-new )
FINAL(a):
  for english word e-new = 0 ... en
    for foreign word f-new = 0 ... fn
      if ( ( e-new not aligned or f-new not aligned ) and
           ( e-new, f-new ) in alignment a )
        add alignment point ( e-new, f-new )
```


Phrase-Based Translation



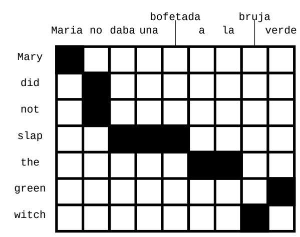
- Foreign input is segmented in phrases
  - any sequence of words, not necessarily linguistically motivated
- Each phrase is translated into English
- Phrases are reordered
- See [Koehn et al., NAACL2003] as introduction

Advantages of Phrase-Based Translation

- Many-to-many translation can handle non-compositional phrases
- Use of local context in translation
- The more data, the longer phrases can be learned

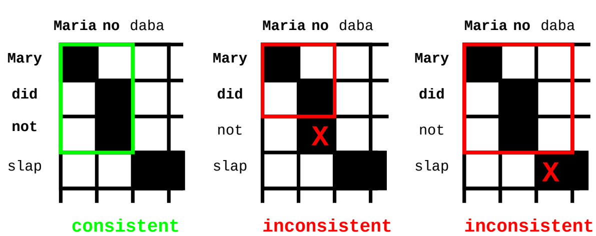
How to Learn the Phrase Translation Table?

- Start with the word alignment:



- Collect all phrase pairs that are consistent with the word alignment

Consistent with Word Alignment

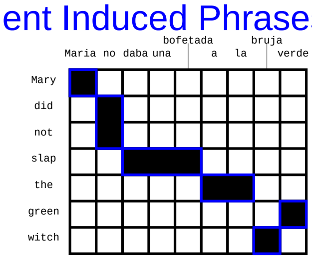


- Consistent with the word alignment :=

  phrase alignment has to contain all alignment points for all covered words

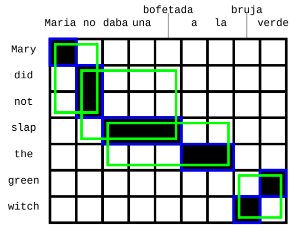
$$(\bar{e}, \bar{f}) \in BP \Leftrightarrow \quad \forall e_i \in \bar{e} : (e_i, f_j) \in A \rightarrow f_j \in \bar{f}$$
$$\text{AND} \quad \forall f_j \in \bar{f} : (e_i, f_j) \in A \rightarrow e_i \in \bar{e}$$

Word Alignment Induced Phrases



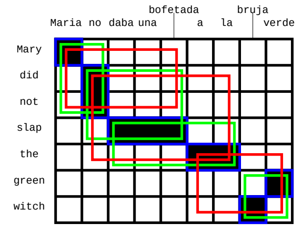
(Maria, Mary), (no, did not), (slap, daba una bofetada), (a la, the), (bruja, witch), (verde, green)

Word Alignment Induced Phrases (2)



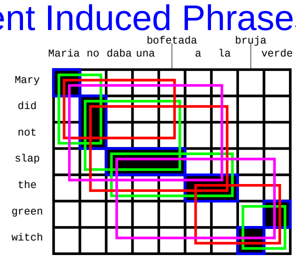
(Maria, Mary), (no, did not), (slap, daba una bofetada), (a la, the), (bruja, witch), (verde, green), (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch)

Word Alignment Induced Phrases (3)



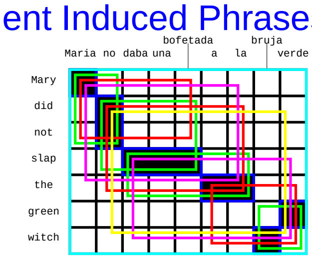
(Maria, Mary), (no, did not), (slap, daba una bofetada), (a la, the), (bruja, witch), (verde, green), (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch),
(Maria no daba una bofetada, Mary did not slap),
(no daba una bofetada a la, did not slap the), (a la bruja verde, the green witch)

Word Alignment Induced Phrases (4)



(Maria, Mary), (no, did not), (slap, daba una bofetada), (a la, the), (bruja, witch), (verde, green), (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch),
(Maria no daba una bofetada, Mary did not slap),
(no daba una bofetada a la, did not slap the), (a la bruja verde, the green witch),
(Maria no daba una bofetada a la, Mary did not slap the),
(daba una bofetada a la bruja verde, slap the green witch)

Word Alignment Induced Phrases (5)



(Maria, Mary), (no, did not), (slap, daba una bofetada), (a la, the), (bruja, witch), (verde, green), (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch),
(Maria no daba una bofetada, Mary did not slap),
(no daba una bofetada a la, did not slap the), (a la bruja verde, the green witch),
(Maria no daba una bofetada a la, Mary did not slap the),
(daba una bofetada a la bruja verde, slap the green witch),
(no daba una bofetada a la bruja verde, did not slap the green witch),
(Maria no daba una bofetada a la bruja verde, Mary did not slap the green witch)

Probability Distribution of Phrase Pairs

- We need a probability distribution $\phi(\bar{f}|\bar{e})$ over the collected phrase pairs

⇒ Possible choices
  - relative frequency of collected phrases: $\phi(\bar{f}|\bar{e}) = \frac{\text{count}(\bar{f},\bar{e})}{\sum_{\bar{f}} \text{count}(\bar{f},\bar{e})}$
  - or, conversely $\phi(\bar{e}|\bar{f})$
  - use lexical translation probabilities

Reordering

- Monotone translation
  - do not allow any reordering
  - → worse translations
  - however: limiting reordering to maximum movement helps
- Distance-based reordering cost
  - moving a foreign phrase over n words: cost ω^n
- Lexicalized reordering model
  - p(monotone|e,f)
  - p(swap|e,f)
  - p(-3|e,f)

Log-Linear Models

- IBM Models provided mathematical justification for factoring components together

  $p_{LM} \times p_{TM} \times p_D$
- These may be weighted

  $p_{LM}^{\lambda_{LM}} \times p_{TM}^{\lambda_{TM}} \times p_D^{\lambda_D}$
- Many components p_i with weights λ_i

  $\Rightarrow \prod_i p_i^{\lambda_i} = exp(\sum_i \lambda_i log(p_i))$

  $\Rightarrow log \prod_i p_i^{\lambda_i} = \sum_i \lambda_i log(p_i)$

Set Feature Weights

- Contribution of components p_i determined by weight λ_i
- Methods
  - manual setting of weights: try a few, take best
  - automate this process
- Learn weights
  - set aside a development corpus
  - set the weights, so that optimal translation performance on this development corpus is achieved
  - requires automatic scoring method (e.g., BLEU)

Additional Features

- Word count
  - add fixed factor for each generated word
  - if output is too short $\rightarrow$ add benefit for each word
- Phrase count
  - add fixed factor for each phrase
  - balances use of longer or shorter phrases