
Agenda for today (Jan.27)

- Introduction to Machine Translation
 - Data-driven statistical machine translation
 - Translation models
 - Parallel corpora
 - Document-, sentence-, word-alignment
 - Phrase-based translation
 - Alignment algorithms (more Viterbi)
 - Edit distance
 - Substitution matrices and gap penalties
 - Alignment in MT
 - MT decoding algorithm
 - MT evaluation

Machine Translation

- Mapping from a *source* language string to a *target* language string, e.g.,

Spanish source:

Perros pequeños tienen miedo de mi hermanita torpe

English target:

Small dogs fear my clumsy little sister

- The “right way” to do this
 - Map the source language to some semantic *interlingua*, e.g.,
fear(dog([plural],[small]),sister([my,singular],[young,clumsy]))
 - Generate the target string from the interlingual representation
- This isn’t feasible in current state of technology

Current best approaches to MT

- Statistical models are the current best practice
 - e.g., Google translation is data driven
- Basic approach taken from statistical speech recognition
 - Let source string be f and target language be e

$$\begin{aligned}\operatorname{argmax}_e P(e \mid f) &= \operatorname{argmax}_e \frac{P(f \mid e) P(e)}{P(f)} \\ &= \operatorname{argmax}_e P(f \mid e) P(e)\end{aligned}$$

- $P(f \mid e)$ is the translation model
(akin to acoustic model in statistical speech recognition)
- $P(e)$ is the language model)

Translation model

- Given a pair of strings $\langle f, e \rangle$, assigns $P(f | e)$
 - If f looks like a good translation of e , then $P(f | e)$ will be high
 - If f doesn't look like a good translation of e , then $P(f | e)$ will be low
- Where do these pairs of strings $\langle f, e \rangle$ come from?
 - Paying people to translate from multiple languages is expensive
 - Would rather get free resources, even if imperfect (or “noisy”) data
 - Such data is produced independently: parallel corpora

Parallel corpora

- Examples:
 - The Hansards corpus of Canadian Parliament transcripts, by law in both French and English
 - Similar resources for EU official proceedings and documents
 - Software manuals, web pages, other available data
- Document-aligned
- Must be sentence- and word-aligned to derive models

Learning alignment models

- If we only have document-aligned parallel corpora, how do we get to the sentence alignment?
- Simple heuristics based on length of sentences.
- Once we have sentence-aligned parallel corpora, how do we get to the word alignment?
- One answer: align words that often appear together

Example parallel corpus

Small dogs fear my clumsy little sister. Because she is so clumsy, the dogs think she will fall on them. Big dogs do not fear her, just the small ones. They do not fear my little sister because she fears them.

Perros pequeños tienen miedo de mi hermanita torpe. Porque es tan torpe, los perros creen que ella se caerá sobre ellos. Perros grandes no tienen miedo de ella, solo los pequeños. No tienen miedo de mi hermanita porque ella tiene miedo de ellos.

Example sentence alignment

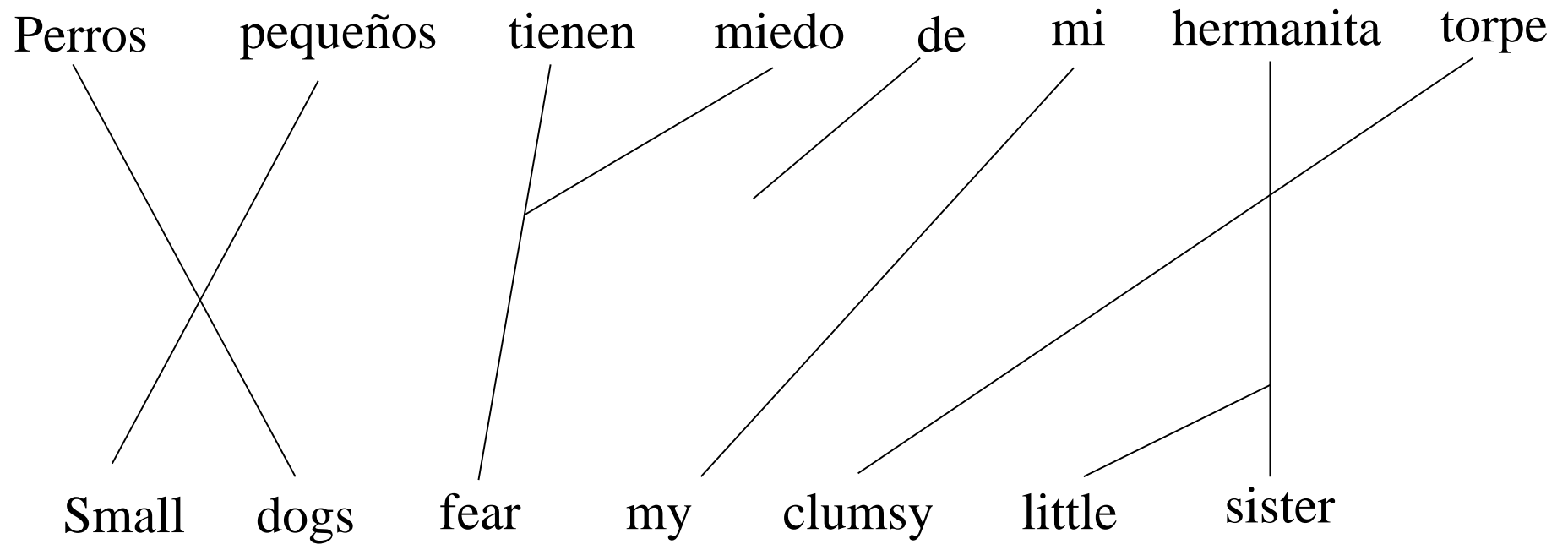
Small dogs fear my clumsy little sister	Perros pequeños tienen miedo de mi hermanita torpe
Because she is so clumsy, the dogs think she will fall on them	Porque es tan torpe, los perros creen que ella se caerá sobre ellos
Big dogs do not fear her, just the small ones	Perros grandes no tienen miedo de ella, solo los pequeños
They do not fear my little sister because she fears them	No tienen miedo de mi hermanita porque ella tiene miedo de ellos

Example word alignment

Perros pequeños tienen miedo de mi hermanita torpe

Small dogs fear my clumsy little sister

Example word alignment



Notation

- Source string: $f = f_1 \dots f_{|f|}$
- Target string: $e = e_1 \dots e_{|e|}$
- Alignment under the assumption of at most one target word per source word: $a = a_1 \dots a_{|f|}$, where $0 \leq a_i \leq |e|$
- $a_i = j$ if f_i aligns with e_j
- $a_i = 0$ if f_i is unaligned with anything in e
- Thus for our example:

$f =$ Perros pequeños tienen miedo de mi hermanita torpe

$e =$ Small dogs fear my clumsy little sister

$a =$ 2 1 3 3 0 4 7 5

Probabilistic modeling

- Given a target string, assign joint probabilities to source strings and alignments: $P(f, a \mid e)$

- The probability of the source string is the sum over all alignments

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

- The best alignment is the one that maximizes the probability

$$\hat{a} = \operatorname{argmax}_a P(f, a \mid e)$$

- Decompose full joint into product of conditionals:

$$P(f, a \mid e) = P(F \mid e) \prod_{i=1}^F P(f_i, a_i \mid e f_1 a_1 \dots f_{i-1} a_{i-1})$$

where $F = |f|$

Dynamic programming in alignments

- Matching two strings corresponds to finding the best alignment between the strings according to some distance metric
- Searching for some minimal distance between two strings
- Linguistic motivation
 - Spell check
 - Morpheme sequence homology across languages
- Biological motivation
 - Sequence similarities imply functional similarities
 - Pairs of proteins related by common ancestry
 - DNA sequence homology across species

Edit distance

- Given two strings, one can ask: how many changes to the first string would it take to yield the second?
- For example, if I typed ‘eammpld’ but meant ‘example’
 - first need to add back the ‘x’: eammpld → exammpld
 - next need to remove the extra ‘m’: exammpld → exampld
 - next need to switch the ‘d’ to an ‘e’: exampld → example
 - One insertion, one deletion and one substitution: 3 edits
- Many other ways to map ‘eammpld’ onto ‘example,’ some more reasonable than others
 - first remove all of the letters in ‘eammpld,’ then insert all of the letters in ‘example’
 - Seven deletions and seven insertions: 14 edits
- Of all possible mappings, which has the LEAST number of edits?

Edit distance dynamic programming algorithm

- Given two strings S_1 and S_2 of length m and n respectively
- Let $F(i, j)$ be the fewest edits mapping $S_1[1, i]$ to $S_2[1, j]$
- Let $F(0, j) = j$ and $F(i, 0) = i$ for all i, j
- Let $M[x, y]$ be the cost of mapping from symbol x to symbol y

$$M[x, y] = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise} \end{cases}$$

- Then

$$F(i, j) = \min \left\{ \begin{array}{l} F(i, j-1) + 1, \\ F(i-1, j) + 1, \\ F(i-1, j-1) + M[S_1(i), S_2(j)] \end{array} \right\}$$

Tabular representation: ‘perambulate’ \rightarrow ‘preamble’

			p	r	e	a	m	b	l	e
	$\begin{smallmatrix} i \\ \downarrow \\ j \end{smallmatrix} \rightarrow$	0	1	2	3	4	5	6	7	8
	0									
p	1									
e	2									
r	3									
a	4									
m	5									
b	6									
u	7									
l	8									
a	9									
t	10									
e	11									

Initialize zero positions

			p	r	e	a	m	b	l	e
	$\begin{matrix} i \\ \downarrow \\ j \rightarrow \end{matrix}$	0	1	2	3	4	5	6	7	8
	0	0	1	2	3	4	5	6	7	8
p	1	1								
e	2	2								
r	3	3								
a	4	4								
m	5	5								
b	6	6								
u	7	7								
l	8	8								
a	9	9								
t	10	10								
e	11	11								

Fill cell, $i = 1, j = 1$

			p	r	e	a	m	b	l	e
	$\begin{matrix} i \\ \downarrow \end{matrix} \begin{matrix} j \rightarrow \end{matrix}$	0	1	2	3	4	5	6	7	8
	0	0	1	2	3	4	5	6	7	8
p	1	1	$\swarrow \downarrow \rightarrow \cdot$							
e	2	2								
r	3	3								
a	4	4								
m	5	5								
b	6	6								
u	7	7								
l	8	8								
a	9	9								
t	10	10								
e	11	11								

$$F(1, 1) = \min \left\{ \begin{array}{l} F(1, 0) + 1, \\ F(0, 1) + 1, \\ F(0, 0) + M[p, p] \end{array} \right\}$$

Fill cell, $i = 2, j = 1$

			p	r	e	a	m	b	l	e
	$\begin{matrix} i \\ \downarrow \end{matrix} \begin{matrix} j \rightarrow \end{matrix}$	0	1	2	3	4	5	6	7	8
	0	0	1	2	3	4	5	6	7	8
p	1	1	0							
e	2	2	$\nearrow \downarrow$							
r	3	3								
a	4	4								
m	5	5								
b	6	6								
u	7	7								
l	8	8								
a	9	9								
t	10	10								
e	11	11								

$$F(2, 1) = \min \left\{ \begin{array}{l} F(2, 0) + 1, \\ F(1, 1) + 1, \\ F(1, 0) + M[e, p] \end{array} \right\}$$

Fill cell, $i = 1, j = 2$

			p	r	e	a	m	b	l	e
	$\begin{matrix} i \\ \downarrow \end{matrix} \begin{matrix} j \rightarrow \end{matrix}$	0	1	2	3	4	5	6	7	8
	0	0	1	2	3	4	5	6	7	8
p	1	1	0	$\searrow \downarrow$						
e	2	2	1							
r	3	3								
a	4	4								
m	5	5								
b	6	6								
u	7	7								
l	8	8								
a	9	9								
t	10	10								
e	11	11								

$$F(1, 2) = \min \left\{ \begin{array}{l} F(1, 1) + 1, \\ F(0, 2) + 1, \\ F(0, 1) + M[p, r] \end{array} \right\}$$

Fill cell, $i = 2, j = 2$

			p	r	e	a	m	b	l	e
	$\begin{matrix} i \\ \downarrow \end{matrix} \begin{matrix} j \rightarrow \end{matrix}$	0	1	2	3	4	5	6	7	8
	0	0	1	2	3	4	5	6	7	8
p	1	1	0	1						
e	2	2	1	$\begin{matrix} \searrow \\ \downarrow \end{matrix}$						
r	3	3								
a	4	4								
m	5	5								
b	6	6								
u	7	7								
l	8	8								
a	9	9								
t	10	10								
e	11	11								

$$F(2, 2) = \min \left\{ \begin{array}{l} F(2, 1) + 1, \\ F(1, 2) + 1, \\ F(1, 1) + M[e, r] \end{array} \right\}$$

Fill cell, $i = 3, j = 3$

			p	r	e	a	m	b	l	e
	$\begin{matrix} i \\ \downarrow \end{matrix} \begin{matrix} j \rightarrow \end{matrix}$	0	1	2	3	4	5	6	7	8
	0	0	1	2	3	4	5	6	7	8
p	1	1	0	1	2					
e	2	2	1	1	1					
r	3	3	2	1	$\begin{matrix} \searrow \\ \rightarrow \end{matrix} \downarrow$					
a	4	4								
m	5	5								
b	6	6								
u	7	7								
l	8	8								
a	9	9								
t	10	10								
e	11	11								

$$F(3, 3) = \min \left\{ \begin{array}{l} F(3, 2) + 1, \\ F(2, 3) + 1, \\ F(2, 2) + M[r, e] \end{array} \right\}$$

Fill cell, $i = 11, j = 8$

			p	r	e	a	m	b	l	e
	$\begin{matrix} i \\ \downarrow \end{matrix} \begin{matrix} j \rightarrow \end{matrix}$	0	1	2	3	4	5	6	7	8
	0	0	1	2	3	4	5	6	7	8
p	1	1	0	1	2	3	4	5	6	7
e	2	2	1	1	1	2	3	4	5	6
r	3	3	2	1	2	2	3	4	5	6
a	4	4	3	2	2	2	3	4	5	6
m	5	5	4	3	3	3	2	3	4	5
b	6	6	5	4	4	4	3	2	3	4
u	7	7	6	5	5	5	4	3	3	4
l	8	8	7	6	6	6	5	4	3	4
a	9	9	8	7	7	6	6	5	4	4
t	10	10	9	8	8	7	7	6	5	5
e	11	11	10	9	8	8	8	7	6	$\searrow \downarrow$

Minimal edit distance: cell $i = 11, j = 8$

			p	r	e	a	m	b	l	e
	$\begin{smallmatrix} i \\ \downarrow \\ j \end{smallmatrix} \rightarrow$	0	1	2	3	4	5	6	7	8
	0	0	1	2	3	4	5	6	7	8
p	1	1	0	1	2	3	4	5	6	7
e	2	2	1	1	1	2	3	4	5	6
r	3	3	2	1	2	2	3	4	5	6
a	4	4	3	2	2	2	3	4	5	6
m	5	5	4	3	3	3	2	3	4	5
b	6	6	5	4	4	4	3	2	3	4
u	7	7	6	5	5	5	4	3	3	4
l	8	8	7	6	6	6	5	4	3	4
a	9	9	8	7	7	6	6	5	4	4
t	10	10	9	8	8	7	7	6	5	5
e	11	11	10	9	8	8	8	7	6	5

Find the optimal alignment

- Now we know that the lowest cost of aligning ‘perambulate’ to ‘preamble’ is 5
- Just knowing this cost might be useful in some cases
- But in general, we want to know *which* edits led to the optimal alignment
- Thus, backtrack to find the path(s) corresponding to the score in bottom-right cell ($i = 11, j = 8$)
 - (Why might we have more than one optimal path?)


















Backtrace

- Can find the path(s) corresponding to final score in $O(n + m)$
- While filling in the matrix, keep a backpointer $B(i, j)$ for each cell such that

$$B(i, j) = \operatorname{argmin} \left\{ \begin{array}{l} F(i, j-1) + 1, \\ F(i-1, j) + 1, \\ F(i-1, j-1) + M[S_1(i), S_2(j)] \end{array} \right\}$$

- On a match/substitution, $B(i, j)$ will point to cell $(i-1, j-1)$
- On an insertion, $B(i, j)$ will point to cell $(i, j-1)$
- On a deletion, $B(i, j)$ will point to cell $(i-1, j)$
- On a tie, $B(i, j)$ may point to multiple cells

Backpointers along optimal path(s)

			p	r	e	a	m	b	l	e
	$\begin{matrix} i \\ \downarrow \end{matrix} j \rightarrow$	0	1	2	3	4	5	6	7	8
	0									
p	1									
e	2									
r	3				 					
a	4									
m	5									
b	6									
u	7									
l	8									
a	9									
t	10									
e	11									

Paths correspond to alignments

- Three different alignments result in edit distance of 5:

1.

p	r	e	a	m	b	-	l	-	-	e
p	e	r	a	m	b	u	l	a	t	e

2.

p	-	r	e	a	m	b	-	l	-	-	e
p	e	r	-	a	m	b	u	l	a	t	e

3.

p	r	e	-	a	m	b	-	l	-	-	e
p	-	e	r	a	m	b	u	l	a	t	e

- Can choose to slightly skew costs to avoid such ambiguities
 - e.g., score substitutions at cost 0.99

Substitution models

- For single-language sequences like spell checking, typically looking for full approximate matches
- For multi-language sequences like MT, might look to match subsequences (e.g., for (orthographic?) similarity across languages)
- Need some way to find “likely” related subsequences, i.e., approximate matches that probably didn’t arise by chance
 - Build “random” model, whereby two sequences are modeled independently
 - Build joint model, whereby two sequences are modeled together
 - Compare likelihoods via log likelihood or log odds ratio
- This is a principled way to capture the fact that particular symbols tend to substitute for each other
 - i.e., are linguistically or semantically related

Substitution likelihood

- Let $q(a)$ be the probability of observing symbol a
- Let $p(ab)$ be the probability that symbols a and b are substituted
- Then, for a given ungapped alignment between S_1 and S_2 , the *odds ratio* between the joint and random models is

$$\text{odds}(S_1, S_2) = \frac{\prod_i p(S_1(i)S_2(i))}{\prod_i q(S_1(i)) \prod_i q(S_2(i))} = \prod_i \frac{p(S_1(i)S_2(i))}{q(S_1(i))q(S_2(i))}$$

- Taking the log, we get

$$\text{log-odds}(S_1, S_2) = \sum_i L[S_1(i), S_2(i)]$$

$$\text{where} \quad L[a, b] = \log p(ab) - \log q(a) - \log q(b)$$

- $L[a, b]$ will be positive for symbols with high probability of substitution

Gap penalties

- Not just substitution to consider – also insertion and deletion
- These are penalized as “gaps” of a certain length g
- Linear gap penalties give the same cost d to every single symbol gap
 - Thus, the penalty for a gap of length g is $\gamma(g) = -gd$
- Also, commonly (in biology), an “affine” gap penalty is used
 - A penalty for starting a gap d
 - Another penalty for continuing an already started gap e
 - Thus, the penalty for a gap of length g is $\gamma(g) = -d - (g - 1)e$
- For affine gap penalties, need to keep track of whether gap is started or not
 - (slightly different dynamic programming...)

Local alignment

- Simple idea: allow resetting alignment at any point
- Get high quality local alignments, rather than global alignments
- Same algorithm, except now:

$$F(i, j) = \max \left\{ \begin{array}{l} 0, \\ F(i, j-1) - d, \\ F(i-1, j) - d, \\ F(i-1, j-1) + M[S_1(i), S_2(j)] \end{array} \right\}$$

- Similar modification for multi-state models
- Note: assumes scores less than zero

Initialize zero positions (Global)

			P	A	W	H	E	A	E
	$\begin{matrix} i \\ \downarrow \\ j \end{matrix} \rightarrow$	0	1	2	3	4	5	6	7
	0	0	-8	-16	-24	-32	-40	-48	-56
H	1	-8							
E	2	-16							
A	3	-24							
G	4	-32							
A	5	-40							
W	6	-48							
G	7	-56							
H	8	-64							
E	9	-72							
E	10	-80							

Initialize zero positions (Local)

			P	A	W	H	E	A	E
	$\begin{matrix} i \\ \downarrow \\ j \end{matrix} \rightarrow$	0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
H	1	0							
E	2	0							
A	3	0							
G	4	0							
A	5	0							
W	6	0							
G	7	0							
H	8	0							
E	9	0							
E	10	0							

Great local match – not in global solutions

			P	A	W	H	E	A	E
	$\begin{smallmatrix} i \\ \downarrow \\ j \rightarrow \end{smallmatrix}$	0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
H	1	0	0	0	0	10	0	0	0
E	2	0	0	0	0	2	16	8	6
A	3	0	0	5	0	0	8	21	13
G	4	0	0						
A	5	0	0						
W	6	0	0						
G	7	0	0						
H	8	0	0						
E	9	0	0						
E	10	0	0						

Other alignment models

- Approximate matching
 - Bounded number of differences
 - Exclusion methods
- Multiple sequences to jointly align
- Better models
 - Hidden Markov Models

Heuristic alignments

- Calculate word similarity in some way, e.g., Dice coefficient

$$\text{dice}(i, j) = \frac{2c(e_i, f_j)}{c(e_i)c(f_j)}$$

where $c(e_i, f_j)$ is the count of parallel sentences containing e_i on the source side and f_j on the target side

- Build matrix of similarities
- Align highly-similar words
- Various strategies to align:
 - Choose $a_j = \text{argmax}_i \{\text{dice}(i, j)\}$
 - Greedily choose best link (globally), then remove row and column from matrix (*competitive linking* algorithm)

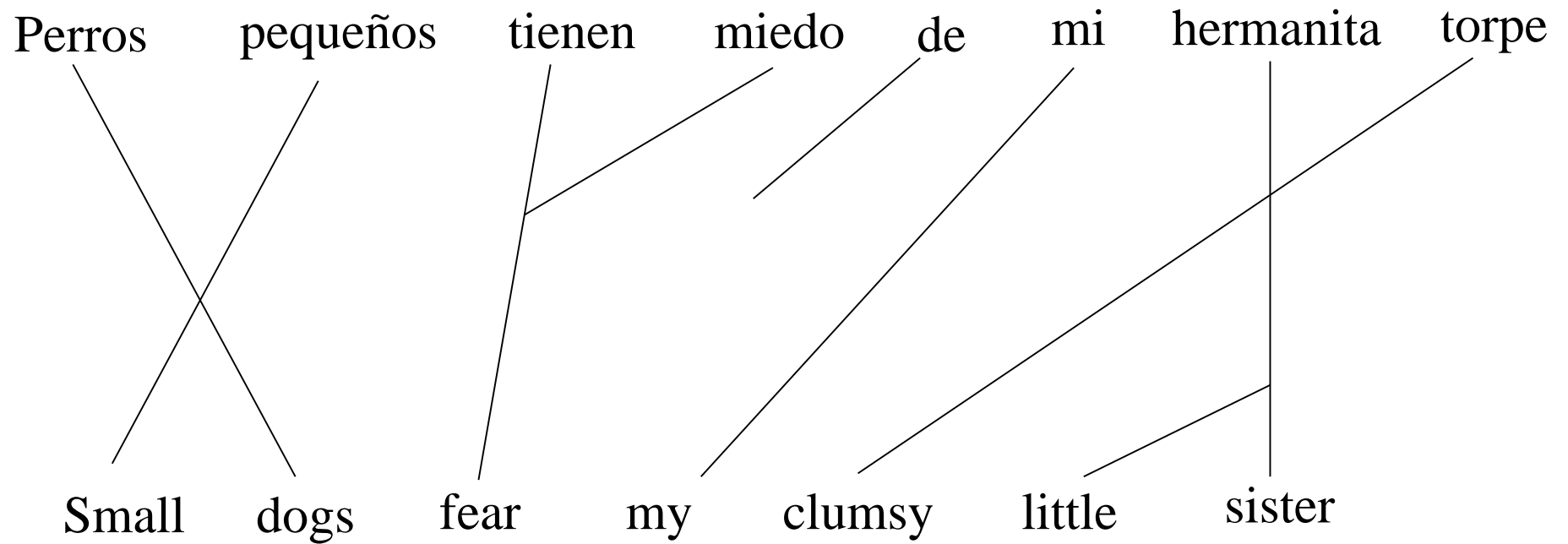
Word-alignment algorithms for MT

- Heuristic
 - Dice
 - Competitive linking
- Statistical
 - IBM models 1-5 [Brown et al. 93]
 - Expectation-Maximization algorithm
 - Another pipeline
 - HMM model [Deng & Byrne 05]
 - GIZA++ software [code.google.com/p/giza-pp/]

Limitations of word-based translation

- One-to-many and many-to-many alignment
 - Some approaches make simplifying assumptions regarding word “fertility”, i.e., number of aligned words
- Crossing alignments
 - Relatively small permutations
 - e.g., post-nominal modifiers (perros pequeños \Rightarrow small dogs)
 - Relatively large permutations
 - e.g., argument ordering (‘in pain young Skywalker is’)

Example word alignment



Phrase-based translation

- Translate sequences of source-language words into (possibly) sequences of target-language words
- Advantages of phrase-based translation
 - Many-to-many translation
 - Allows for more context in translation
- Phrase table
 - Extracted by “growing” word alignments
 - Limited by phrase length
 - Ambiguity in translation look-up

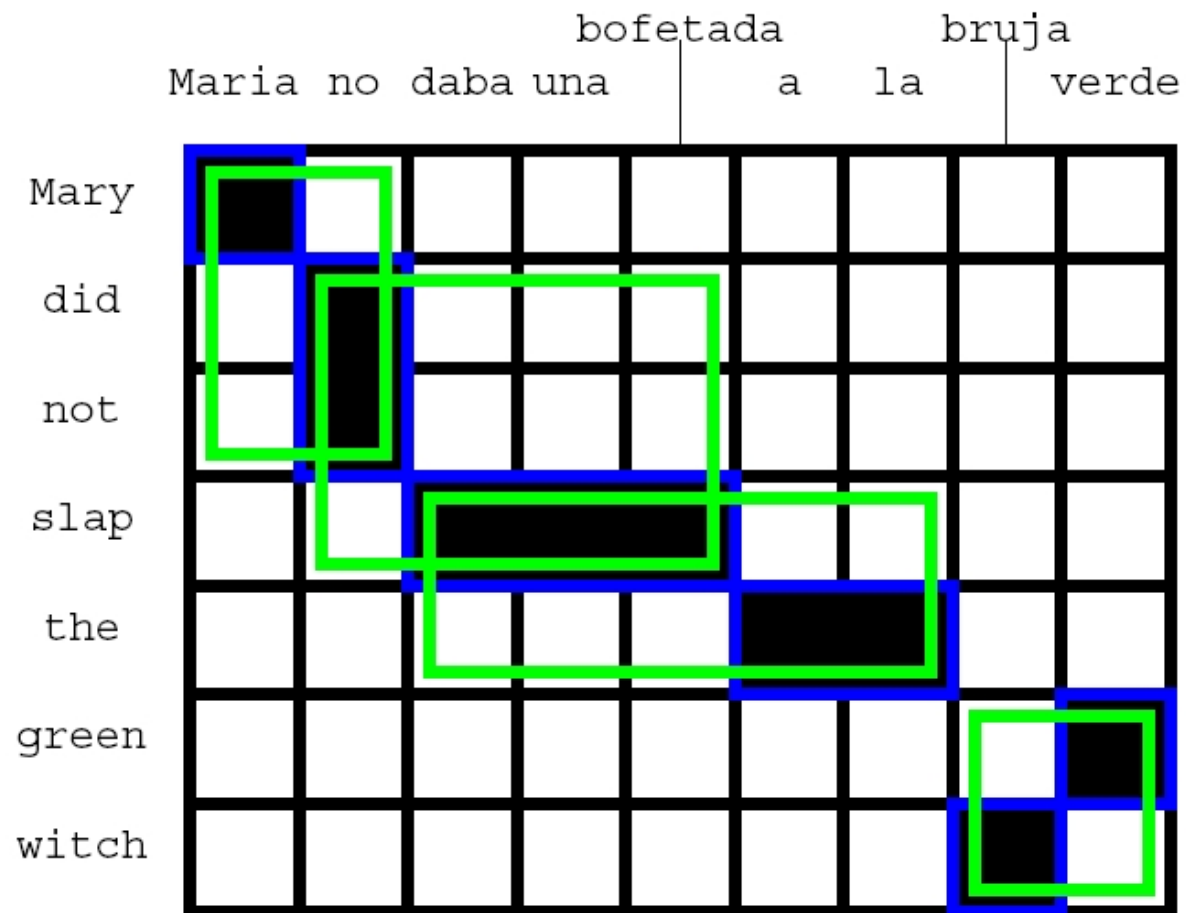
Extracting phrases from word-alignments

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

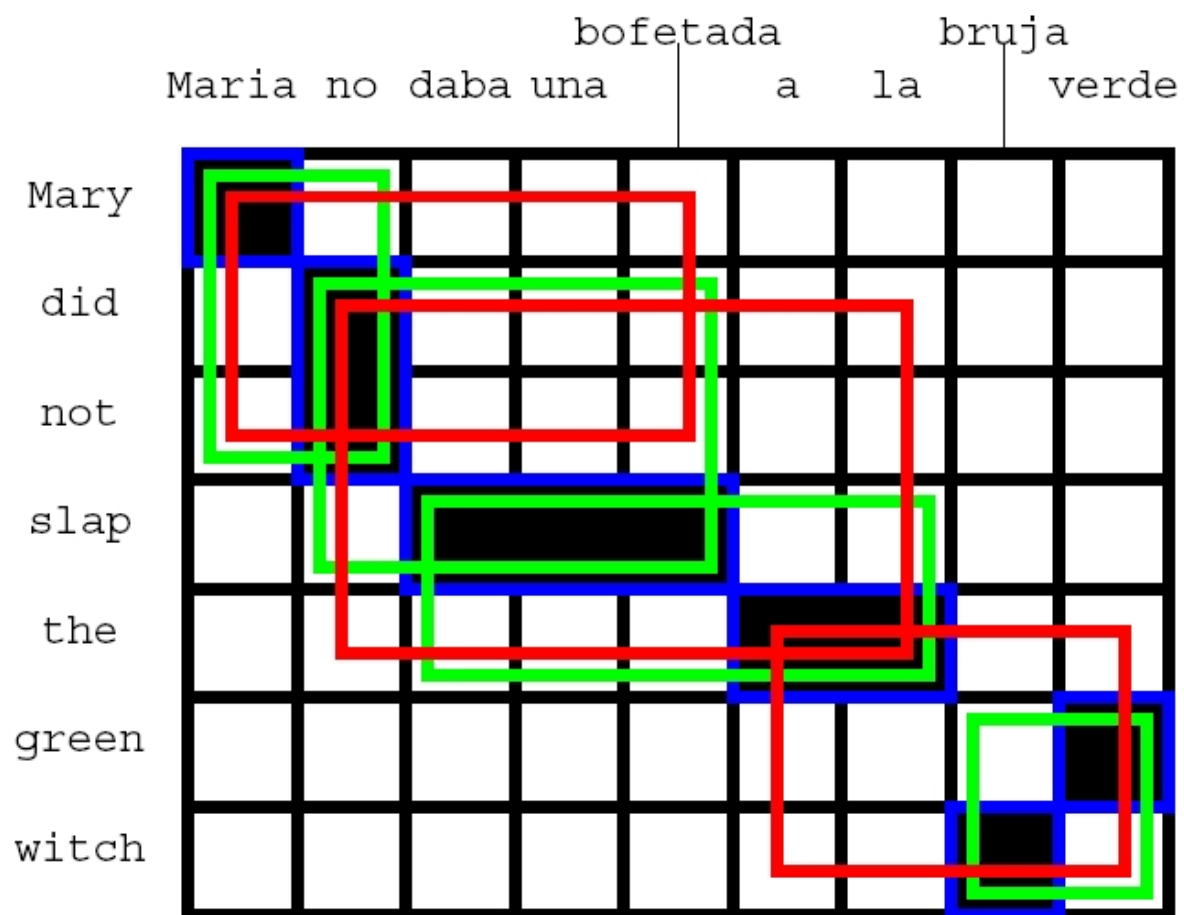
Extracting phrases from word-alignments

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

Extracting phrases from word-alignments



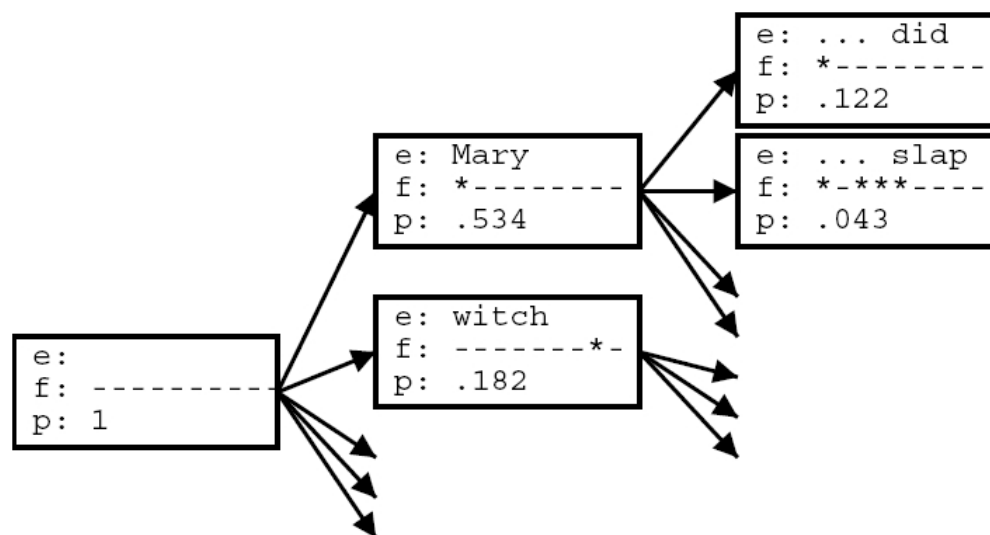
Extracting phrases from word-alignments



Decoding algorithm

- Moses decoder [www.statmt.org/moses/]

- Beam search



- Build English (target language sentence) by hypothesis expansion (left-to-right)
- Ambiguity
- Search space pruning

MT evaluation

- Ideal: human evaluation
 - Adequacy: does the translation correctly capture the information of the source sentence?
 - Fluency: is the translation a “good” sentence of the target language?
 - But: slow and expensive
- Automatic evaluation
 - Intuition: comparing two candidate translations T_1 and T_2
 - To the extent that T_1 overlaps more with a reference (human-produced) translation R , it is “better” than T_2
 - How to measure overlap?
 - Differences in length of translation?
 - Multiple reference translations?

BLEU

- Measure overlap by counting n -grams in candidate that match the reference translation
- More matches \Rightarrow better translation
- Precision metric
- Brevity penalty

$$\log \text{BLEU} = \min\left(1 - \frac{r}{c}, 0\right) + \sum_{n=1}^N w_n \log(p_n)$$

Brief note on text processing

- Tokenization
- Casing

Further topics of exploration

- Translation model
 - More, better, different data
 - Different word-alignment algorithms
 - Length of extracted phrases
- Language model (didn't discuss this lecture...)
 - More, better, different data
 - Size of n -grams
- Add more knowledge to the process
 - Numbers
 - Dates
 - Named entities