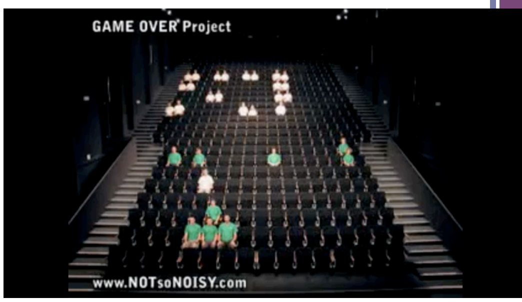


Natural Language Processing 2

CS151
David Kauchak

Admin

- Project reports
- Keep up with the homework
- NLP spring course?

Language translation

Noisy channel model

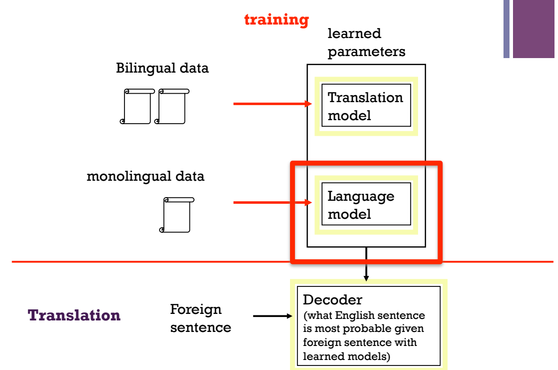
model

$$p(e \mid f) \propto p(f \mid e)\, p(e)$$

translation model — how do foreign sentences get translated to English sentences?

language model — what do English sentences look like?

Statistical MT Overview



Language modeling

Step 1: decompose the probability

P(I think today is a good day to be me) =

P(I) x
P(think | I) x
P(today | I think) x
P(is | I think today) x
P(a | I think today is) x
P(good | I think today is a) x
…

The n-gram Approximation

Assume each word depends only on the previous n-1 words (e.g. trigram: three words total)

P(is | I think today) ≈ P(is | think today)
P(a | I think today is) ≈ P(a | today is)
P(good | I think today is a) ≈ P(good | is a)

Estimating probabilities

- How do we find probabilities? P(is | think today)
- Get real text, and start counting!

$$P(\text{is} \mid \text{think today}) = \frac{\text{count(think today is)}}{\text{count(think today)}}$$

Smoothing

Is this sentence reasonable?

The Singing Toadstools are a new band.

$$P(\text{are} \mid \text{Singing Toadstools}) = \frac{\text{count(Singing Toadtools are)}}{\text{count(Singing Toadstools)}}$$

"singing toadstools are" Search

Advanced search

No results found for **"singing toadstools are"**.

Smoothing: Add One (Laplacian)

- Add one smoothing: $P(c \mid ab) \approx \dfrac{C(abc)+1}{C(ab)+V}$
- Works very badly. DO NOT DO THIS
- Add delta smoothing: $P(c \mid ab) \approx \dfrac{C(abc)+\delta}{C(ab)+\delta V}$
- Still very bad. DO NOT DO THIS

A better idea

Trigram: P(is | think today)

Bigram: P(is | today)

Unigram: P(is)

How are these probabilities related?

Is one more likely to be 0 than another?

Could we combine them somehow?

+ Two general approaches

- Interpolation
  - $p^*(z|\ x,y) = \lambda\, p(z \mid x, y) + \mu\, p(z \mid y) + (1-\lambda-\mu)_2 p(z)$
  - Combine the probabilities with some linear combination
- Backoff

$$P(z \mid xy) = \begin{cases} \frac{C^*(xyz)}{C(xy)} & \text{if } C(xyz) > 0 \\ \alpha(xy)P(z \mid y) & \text{otherwise} \end{cases}$$

  - Combine the probabilities by "backing off" to lower models only when we don't have any information

+ Smoothing: Simple Interpolation

$$P(z \mid xy) \approx \lambda \frac{C(xyz)}{C(xy)} + \mu \frac{C(yz)}{C(y)} + (1-\lambda-\mu)\frac{C(z)}{C(\bullet)}$$

- Trigram is very context specific, very noisy
- Unigram is context-independent, smooth
- Interpolate Trigram, Bigram, Unigram for best combination
- How should we determine λ and μ ?

+ Smoothing: Finding parameter values

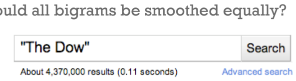
- Split data into training, "heldout", test
- Try lots of different values for λ, μ on heldout data, pick best
- Two approaches for finding these efficiently
  - EM (expectation maximization)
  - "Powell search" – see Numerical Recipes in C

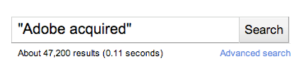
+ Smoothing: Jelinek-Mercer

- Simple interpolation:

$$P_{smooth}(z \mid xy) = \lambda \frac{C(xyz)}{C(xy)} + (1-\lambda)P_{smooth}(z \mid y)$$

- Should all bigrams be smoothed equally?





Smoothing: Jelinek-Mercer

- Simple interpolation:

$$P_{smooth}(z \mid xy) = \lambda \frac{C(xyz)}{C(xy)} + (1-\lambda) P_{smooth}(z \mid y)$$

- Smooth a little after "The Dow", more after "Adobe acquired"

$$P_{smooth}(z \mid xy) = \lambda(C(xy)) \frac{C(xyz)}{C(xy)} + (1-\lambda(C(xy)) P_{smooth}(z \mid y)$$

Smoothing: Jelinek-Mercer continued

$$P_{smooth}(z \mid xy) = \lambda(C(xy)) \frac{C(xyz)}{C(xy)} + (1-\lambda(C(xy)) P_{smooth}(z \mid y)$$

- Bin counts by frequency and assign a λs for each bin
- Find λs by cross-validation on held-out data

Backoff models: absolute discounting

$$P_{absolute}(z \mid xy) = \begin{cases} \frac{C(xyz) - D}{C(xy)} & if\ C(xyz) > 0 \\ \alpha(xy) P_{absolute}(z \mid y) & otherwise \end{cases}$$

- Subtract some absolute number from each of the counts (e.g. 0.75)
  - will have a large effect on low counts
  - will have a small effect on large counts

Kneser-Ney

- Idea: not all counts should be discounted with the same value

P(Francisco | eggplant) vs
P(stew | eggplant)

If we've never seen either, which should be more likely? why?

What would an interpolated/backoff model say?

What is the problem?

+ Kneser-Ney

- Idea: not all counts should be discounted with the same value
- "Francisco" is common, so backoff/interpolated methods say it is likely
  - But it only occurs in context of "San"
- "Stew" is common, and in many contexts

- Weight backoff by number of contexts word occurs in

P(Francisco | eggplant) low
P(stew | eggplant) higher

+ Smoothing

- Lots of other approaches…
  - Good Turing: estimate unseen events based on 1-count events
  - Katz, Witten-Bell,…

- In practice, Kneser-Ney works very well (or minor modifications of it)

+ Other language model ideas?

- Skipping models: rather than just the previous 2 words, condition on the previous word and the 3rd word back, etc.
- Caching models: phrases seen are more likely to be seen again (helps deal with new domains)
- Clustering:
  - some words fall into categories (e.g. Monday, Tuesday, Wednesday…)
  - smooth probabilities with category probabilities
- Domain adaptation:
  - interpolate between a general model and a domain specific model

+ Language model evaluation

- We have two different language models (i.e. two different probability distributions over English)
- How can we determine which is better?
  - Idea 1: use it in our MT system and see which works better
  - Idea 2: should predict actual English sentences with high probability

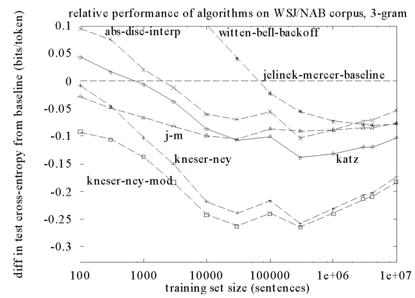
+ Perplexity

- Ask the two models to predict the likelihood of some test data
- The one with the higher probability is better
- Perplexity standardizes this idea by averaging over the probability of all words:

$$\max \sqrt[n]{\prod_{i=1}^{n} P(w_i \mid w_{1..i-1})} \cong \min \frac{\sum_{i=1}^{n} \log p(w_i \mid w_{1...i-1})}{n}$$

- Or… can be seen as the average log of the probability

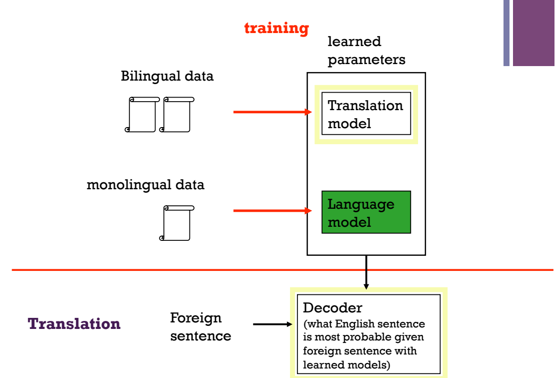
+ Smoothing results



+ Language Modeling Toolkits

- SRI
  - http://www-speech.sri.com/projects/srilm/
- CMU
  - http://www.speech.cs.cmu.edu/SLM_info.html

+ Statistical MT Overview



+ The Classic Translation Model

Word Substitution/Permutation
[IBM Model 3, Brown et al., 1993]

$p(f \mid e)$

Generative story:

Mary did not slap the green witch

How do we get from English to Spanish?

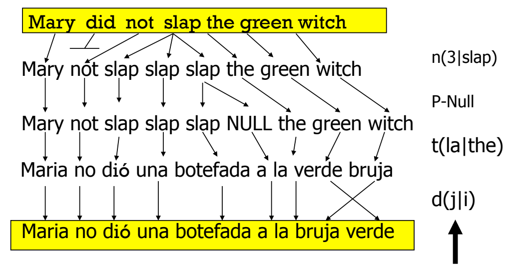
Maria no dió una botefada a la bruja verde

+ The Classic Translation Model

Word Substitution/Permutation
[IBM Model 3, Brown et al., 1993]

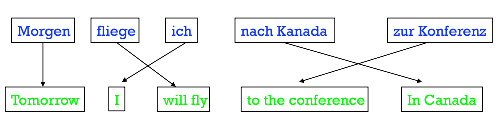
$p(f \mid e)$

Generative story:



Probabilities can be learned from raw bilingual text.

+ Phrase-based translation model

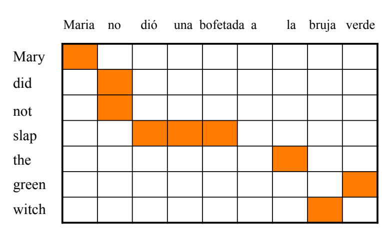


- Phrase-based machine translation:
  - break English into phrases
  - probabilistically translate those phrases, $p(f_{phrase} \mid e_{phrase})$
  - with some probability, phrases can be reordered

How can we learn these?

+ How to Learn the Phrase Translation Table?

- Start with word alignment, build phrases from that

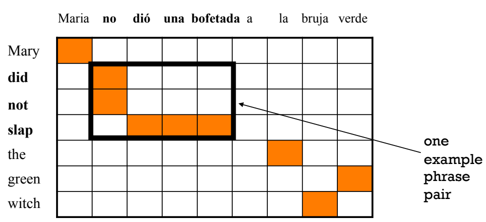


This word-to-word alignment is a by-product of training a translation model like IBM-Model-3.

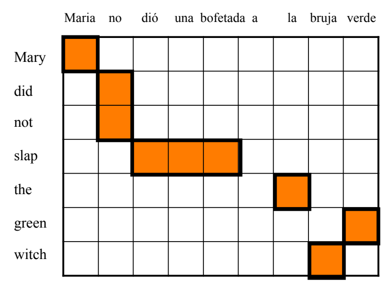
This is the best (or "Viterbi") alignment.

+ How to Learn the Phrase Translation Table?

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

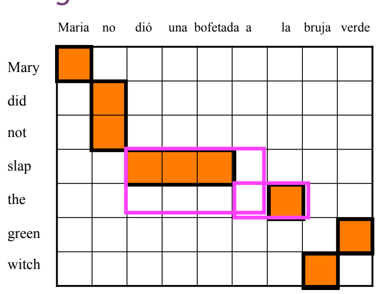


+ Word Alignment Induced Phrases



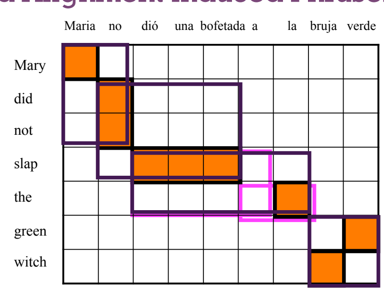
(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)

+ Word Alignment Induced Phrases



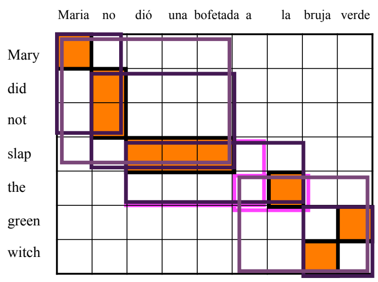
(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)
(a la, the) (dió una bofetada a, slap the)

+ Word Alignment Induced Phrases



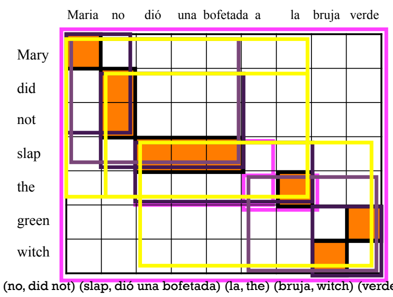
(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)
(a la, the) (dió una bofetada a, slap the)
(Maria no, Mary did not) (no dió una bofetada, did not slap), (dió una bofetada a la, slap the)
(bruja verde, green witch)

+ Word Alignment Induced Phrases



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

+ Word Alignment Induced Phrases

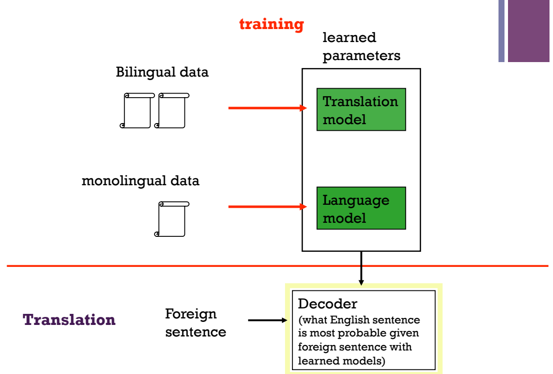


(Maria, Mary) (no, did not) (slap, dió una bofetada) (la, the) (bruja, witch) (verde, green)
(a la, the) (dió una bofetada a, slap the)
(Maria no, Mary did not) (no dió una bofetada, did not slap), (dió una bofetada a la, slap the)
(bruja verde, green witch) (Maria no dió una bofetada, Mary did not slap)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

+ Learning phrase probabilities

- Extract all phrase pairs from our bilingual corpus
- Estimate $p(f_{phrase} | e_{phrase})$ using MLE
- Prune the rules
  - remove rules based on low-frequency events
  - remove rules with very low probabilities

+ Statistical MT Overview



+ Translation (aka decoding)

Translation — Foreign sentence → Decoder (what English sentence is most probable given foreign sentence with learned models)

$$\arg_e \max p(e \mid f) \propto p(f \mid e)\, p(e)$$

phrase-based translation model — n-gram language model

Ideas?

+ Decoding

- Of all conceivable English word strings, find the one maximizing P(e) x P(f | e)
- Decoding is an NP-complete problem (for many translation models)
  - (Knight, 1999)
- AI to the rescue: pose as a search problem

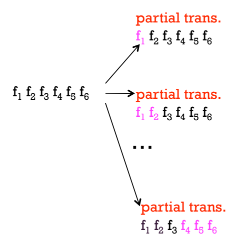
+ Search

$\arg_e \max p(f \mid e)\, p(e)$

$f_1\ f_2\ f_3\ f_4\ f_5\ f_6$

+ Search

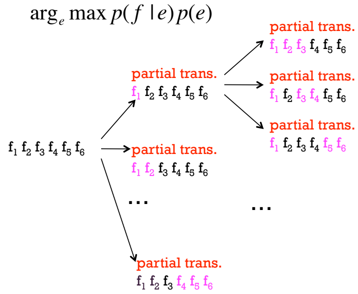
$\arg_e \max p(f \mid e)\, p(e)$



Key: partial translations represent ***inorder*** translations on the English side

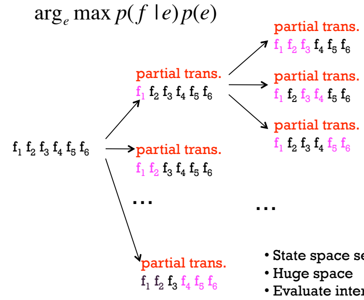
Search

$\arg_e \max p(f \mid e)p(e)$



Search

$\arg_e \max p(f \mid e)p(e)$



- State space search problem
- Huge space
- Evaluate intermediate hypotheses using models and prune partial hypotheses (Beam search)

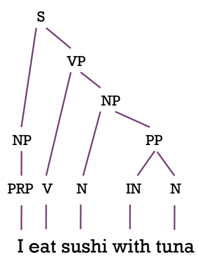
Phrased-based approach

- Works very well!
- Still more or less state of the art
- Google translate currently still uses this approach
  - much larger n-gram language model (5-grams >)
- but… we can do better

Some things are hard to model with phrases…

- Word order within constituents:
  - English NPs: art adj n *the big boy*
  - Hebrew NPs: art n art adj *ha yeled ha gadol*
  - Spanish NPs: art n adj *el niño grande*
- Constituent structure:
  - English is SVO: Subj Verb Obj *I saw the man*
  - Modern Arabic is VSO: Verb Subj Obj
- Different verb syntax:
  - Verb complexes in English vs. in German
    I can eat the apple Ich kann den apfel essen

Syntax!



All of the examples before were motivated by the syntax of the languages

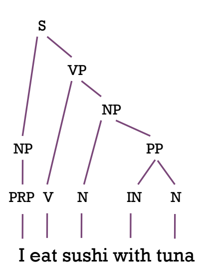
Parsing

- Parsing is the field of NLP interested in automatically determining the syntactic structure of a sentence
- parsing can be thought of as determining what sentences are "valid" English sentences
- As a by product, we often can get the structure
- What types of grammars have you seen in CS?

Context free grammars

- Specify a language (i.e. set of valid sentences)
- If we look at the derivation it forms a tree

S -> NP VP
NP -> PRP
NP -> N PP
VP -> V NP
VP -> V NP PP
PP -> IN N
PRP -> I
V -> eat
N -> sushi
N -> tuna
IN -> with



Parsing

- Given a CFG and a sentence, determine the possible parse tree(s)

S -> NP VP
NP -> PRP
NP -> N PP
VP -> V NP
VP -> V NP PP
PP -> IN N
PRP -> I
V -> eat
N -> sushi
N -> tuna
IN -> with

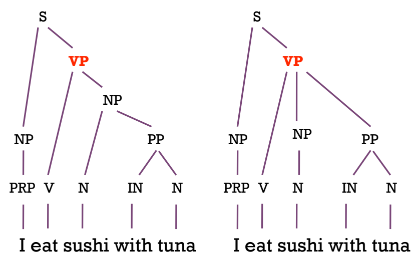
I eat sushi with tuna

What parse trees are possible for this sentence?

How can you algorithmically determine this?

What if the grammar is much larger?

+ Parsing ambiguity



S -> NP VP
NP -> PRP
NP -> N PP
VP -> V NP
VP -> V NP PP
PP -> IN N
PRP -> I
V -> eat
N -> sushi
N -> tuna
IN -> with

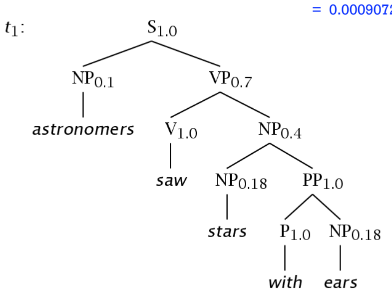
How can we decide between these?

+ A Simple PCFG

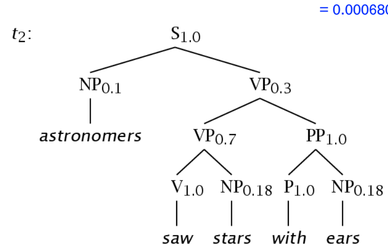
Probabilities!

S	→	NP VP	1.0	NP	→	NP PP	0.4
VP	→	V NP	0.7	NP	→	*astronomers*	0.1
VP	→	VP PP	0.3	NP	→	*ears*	0.18
PP	→	P NP	1.0	NP	→	*saw*	0.04
P	→	*with*	1.0	NP	→	*stars*	0.18
V	→	*saw*	1.0	NP	→	*telescope*	0.1

$P(t_1) = 1.0 * 0.1 * 0.7 * 1.0 * 0.4 * 0.18 * 1.0 * 1.0 * 0.18 = 0.0009072$



$P(t_2) = 1.0 * 0.1 * 0.3 * 0.7 * 1.0 * 0.18 * 1.0 * 1.0 * 0.18 = 0.0006804$



+
Parsing

- Top-down parsing
  - ends up doing a lot of repeated work
  - doesn't take into account the words in the sentence until the end!
- Bottom-up parsing
  - constrain based on the words
  - avoids repeated work (dynamic programming)
  - CKY parser

---

score[i][j] has all of the partial parses spanning words i to j-1

build tree from the bottom up

	cats	scratch	walls	with	claws
0	score[0][1]	score[0][2]	score[0][3]	score[0][4]	score[0][5]
1		score[1][2]	score[1][3]	score[1][4]	score[1][5]
2			score[2][3]	score[2][4]	score[2][5]
3				score[3][4]	score[3][5]
4					score[4][5]
5					

+

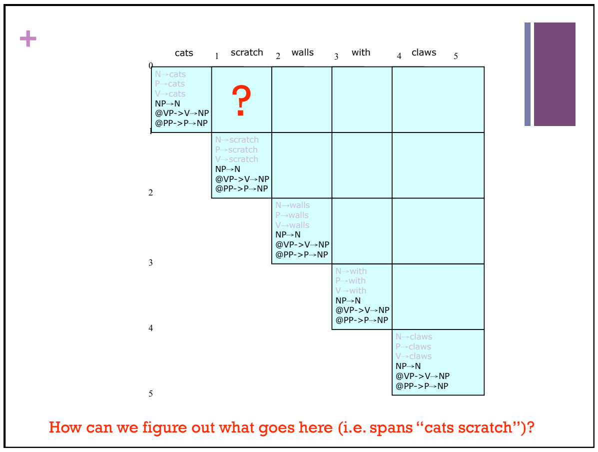
	cats	scratch	walls	with	claws
0	N→cats P→cats V→cats				
1		N→scratch P→scratch V→scratch			
2			N→walls P→walls V→walls		
3				N→with P→with V→with	
4					N→claws P→claws V→claws
5					

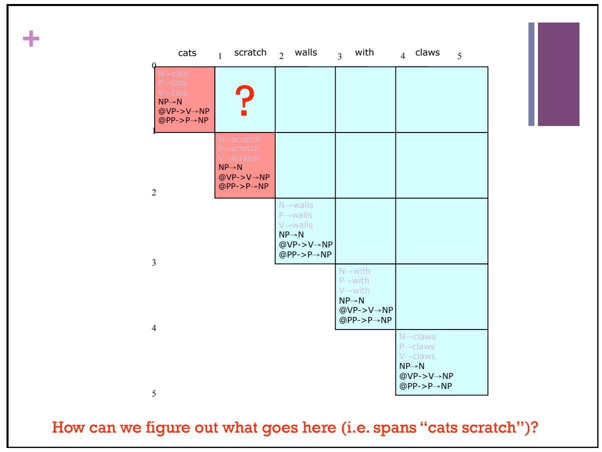
Start at the bottom level

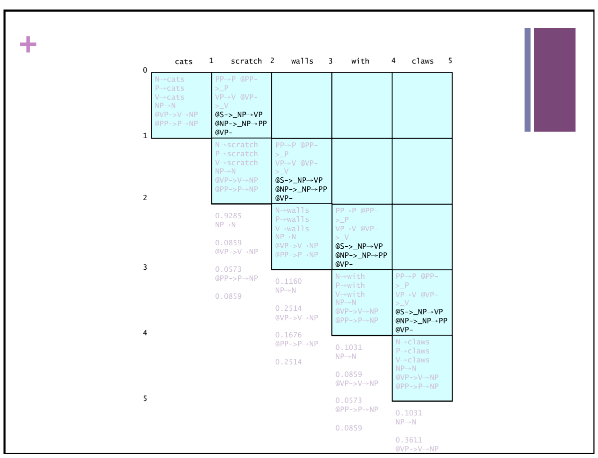
+

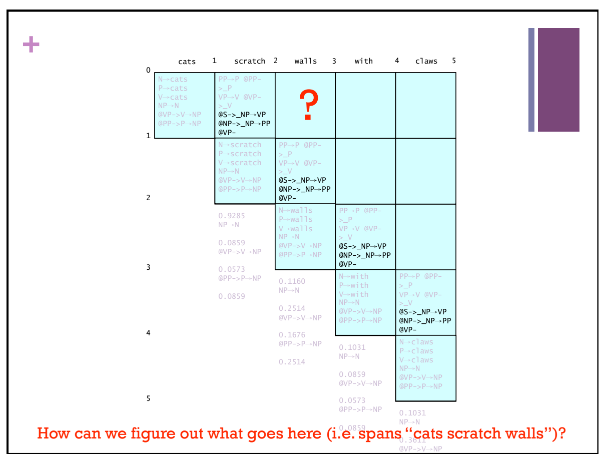
	cats	scratch	walls	with	claws
0	N→cats P→cats V→cats NP→N @VP->V→NP @PP->P→NP				
1		N→scratch P→scratch V→scratch NP→N @VP->V→NP @PP->P→NP			
2			N→walls P→walls V→walls NP→N @VP->V→NP @PP->P→NP		
3				N→with P→with V→with NP→N @VP->V→NP @PP->P→NP	
4					N→claws P→claws V→claws NP→N @VP->V→NP @PP->P→NP
5					

Be careful about unary rules (i.e. rules with 1 element on the RHS)

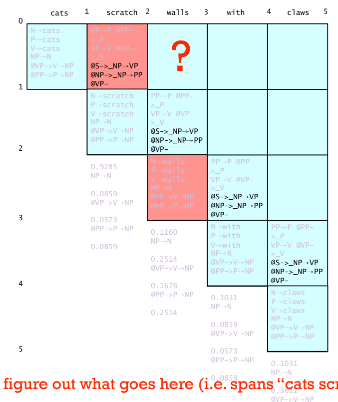

cats
scratch
walls
with
claws
?
NP→N
@VP->V→NP
@PP->P→NP
How can we figure out what goes here (i.e. spans "cats scratch")?


cats
scratch
walls
with
claws
?
NP→N
@VP->V→NP
@PP->P→NP
How can we figure out what goes here (i.e. spans "cats scratch")?


cats
scratch
walls
with
claws
@S->_NP→VP
@NP->_NP→PP
@VP-

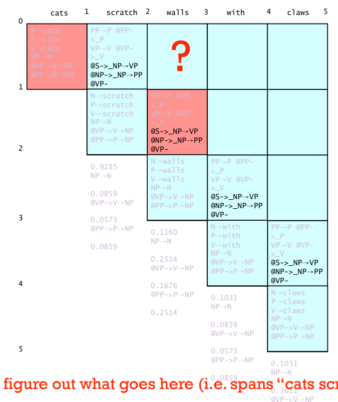

cats
scratch
walls
with
claws
?
@S->_NP→VP
@NP->_NP→PP
@VP-
How can we figure out what goes here (i.e. spans "cats scratch walls")?

+



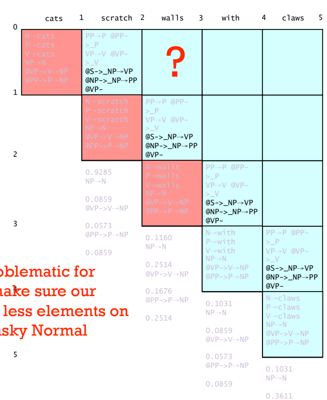
How can we figure out what goes here (i.e. spans "cats scratch walls")?

+



How can we figure out what goes here (i.e. spans "cats scratch walls")?

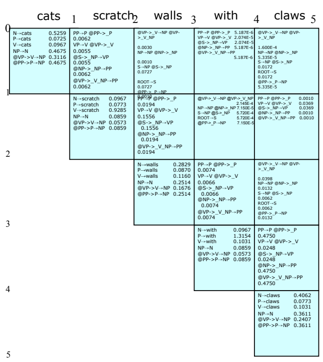
+



This can be problematic for larger spans: make sure our rules have 2 or less elements on the RHS (Chomsky Normal Form)

+

....



Parsing

- Works well in practice
  - many publicly available
  - Stanford, CMU, Berkeley all have good ones available
  - Many with online interfaces:
    - http://nlp.stanford.edu:8080/parser/
- How do we obtain the grammars?
  - Tree banks: contain examples sentences with human annotated parse trees
  - learn rules and probabilities from these tree banks
- Improvements
  - Lexicalize the rules
    - S -> NP VP becomes S(eat) -> NP(I) VP(eat)
  - Machine learning approaches

Other types of parsing

- Part of speech tagging
  - I/PRP eat/VBP sushi/NN with/IN tuna/NN ./.
- Dependency parsing
  - nsubj(eat-2, I-1) – who's doing the eating?
  - dobj(eat-2, sushi-3) – what are they eating?
  - prep_with(sushi-3, tuna-5) – what is the sushi with?

How do people do it?

The horse raced past the barn fell.

The horse *that was* raced past the barn fell.

+ How do people do it?

The old man the boat.

The old *people* man the boat.

+ How do people do it?

The man whistling tunes pianos

The man *who is* whistling tunes pianos

+ How do people do it?

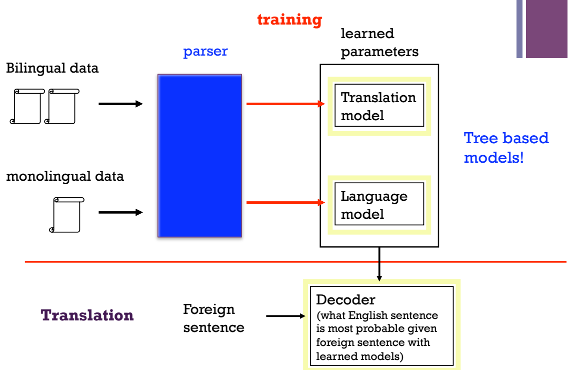
The government plans to raise taxes were defeated.

The plans *of the government* to raise taxes were defeated.

+ Garden path effect

- People tend to parse this a sentence as they read it
  - not bottoms-up
- For garden path sentences, the initial parse is incorrect
  - have to go back and reparse the sentence

+ Statistical MT Overview



+ Tons of other NLP problems

- Information extraction
  - find entities in text
  - find how they relate to each other
- Information retrieval (search)
- Speech recognition
- Speech generation
- Dialogue/conversational agents
  - http://www.chatbots.org/us
  - http://www.geekcavecreations.com/pChat/
- Summarization
- Text simplification/compression

+ Tons of other NLP problems

- Language recognition (play with Google translate)
- Word sense disambiguation
- Text understanding
- Question answering
- Text coherence
  - Coreference resolution
- Text segmentation
- Document classification
- Document clustering
- …