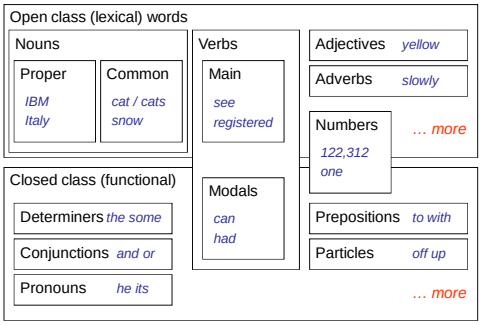
Statistical NLP
Spring 2011

Lecture 6: POS / Phrase MT

Dan Klein – UC Berkeley

Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes



CC	conjunction, coordinating	and both but either or
CD	numeral, cardinal	mid-1890 nine-thirty 0.5 one
DT	determiner	a all an every no that the
EX	existential there	there
FW	foreign word	gemeinschaft hund ich jeux
IN	preposition or conjunction, subordinating	among whether out on by if
JJ	adjective or numeral, ordinal	third ill-mannered regrettable
JJR	adjective, comparative	braver cheaper taller
JJS	adjective, superlative	bravest cheapest tallest
MD	modal auxiliary	can may might will would
NN	noun, common, singular or mass	cabbage thermostat investment subhumanity
NNP	noun, proper, singular	Motown Cougar Yvette Liverpool
NNPS	noun, proper, plural	Americans Materials States
NNS	noun, common, plural	undergraduates bric-a-brac averages
POS	genitive marker	' 's
PRP	pronoun, personal	hers himself it we them
PRP$	pronoun, possessive	her his mine my our ours their thy your
RB	adverb	occasionally maddeningly adventurously
RBR	adverb, comparative	further gloomier heavier less-perfectly
RBS	adverb, superlative	best biggest nearest worst
RP	particle	aboard away back by on open through
TO	"to" as preposition or infinitive marker	to
UH	interjection	huh howdy uh whammo shucks heck
VB	verb, base form	ask bring fire see take
VBD	verb, past tense	pleaded swiped registered saw
VBG	verb, present participle or gerund	stirring focusing approaching erasing
VBN	verb, past participle	dilapidated imitated reunifed unsettled
VBP	verb, present tense, not 3rd person singular	twist appear comprise mold postpone
VBZ	verb, present tense, 3rd person singular	bases reconstructs marks uses
WDT	WH-determiner	that what whatever which whichever
WP	WH-pronoun	that what whatever which who whom
WP$	WH-pronoun, possessive	whose
WRB	Wh-adverb	however whenever where why

Part-of-Speech Ambiguity

- Words can have multiple parts of speech

VBD		VB			
VBN	VBZ	VBP	VBZ		
NNP	NNS	NN	NNS	CD	NN

Fed raises interest rates 0.5 percent

Mrs./NNP Shaefer/NNP never/RB got/VBD **around/RP** to/TO joining/VBG
All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB **around/IN** the/DT corner/NN
Chateau/NNP Petrus/NNP costs/VBZ **around/RB** 250/CD

- Two basic sources of constraint:
  - Grammatical environment
  - Identity of the current word
- Many more possible features:
  - Suffixes, capitalization, name databases (gazetteers), etc…

Why POS Tagging?

- Useful in and of itself (more than you'd think)
  - Text-to-speech: record, lead
  - Lemmatization: saw[v] → see, saw[n] → saw
  - Quick-and-dirty NP-chunk detection: grep {JJ | NN}* {NN | NNS}
- Useful as a pre-processing step for parsing
  - Less tag ambiguity means fewer parses
  - However, some tag choices are better decided by parsers

					IN		
DT	NNP	NN	VBD	VBN	RP	NN	NNS

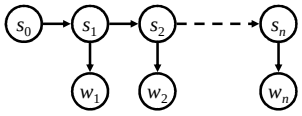
The Georgia branch had taken on loan commitments …

				VDN		
DT	NN	IN	NN	VBD	NNS	VBD

The average of interbank offered rates plummeted …

Classic Solution: HMMs

- We want a model of sequences s and observations w

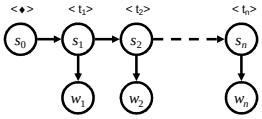


$$P(\mathbf{s}, \mathbf{w}) = \prod_i P(s_i|s_{i-1})P(w_i|s_i)$$

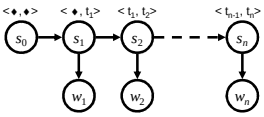
- Assumptions:
  - States are tag n-grams
  - Usually a dedicated start and end state / word
  - Tag/state sequence is generated by a markov model
  - Words are chosen independently, conditioned only on the tag/state
  - These are totally broken assumptions: why?

States

- States encode what is relevant about the past
- Transitions P(s|s') encode well-formed tag sequences
  - In a bigram tagger, states = tags



  - In a trigram tagger, states = tag pairs



Estimating Transitions

- Use standard smoothing methods to estimate transitions:

$$P(t_i \mid t_{i-1}, t_{i-2}) = \lambda_2 \hat{P}(t_i \mid t_{i-1}, t_{i-2}) + \lambda_1 \hat{P}(t_i \mid t_{i-1}) + (1 - \lambda_1 - \lambda_2)\hat{P}(t_i)$$

- Can get a lot fancier (e.g. KN smoothing) or use higher orders, but in this case it doesn't buy much
- One option: encode more into the state, e.g. whether the previous word was capitalized (Brants 00)
- BIG IDEA: The basic approach of state-splitting turns out to be very important in a range of tasks

Estimating Emissions

$$P(\mathbf{s}, \mathbf{w}) = \prod_i P(s_i|s_{i-1})P(w_i|s_i)$$

- Emissions are trickier:
  - Words we've never seen before
  - Words which occur with tags we've never seen them with
  - One option: break out the Good-Turning smoothing
  - Issue: unknown words aren't black boxes:

    343,127.23 11-year Minteria reintroducibly

  - Basic solution: unknown words classes (affixes or shapes)

    D+,D+.D+ D+-x+ Xx+ x+-"ly"

  - [Brants 00] used a suffix trie as its emission model

Disambiguation (Inference)

- Problem: find the most likely (Viterbi) sequence under the model

$$\mathbf{t}^* = \arg\max_{\mathbf{t}} P(\mathbf{t}|\mathbf{w})$$

- Given model parameters, we can score any tag sequence

<◆,◆>	<◆,NNP>	<NNP, VBZ>	<VBZ, NN>	<NN, NNS>	<NNS, CD>	<CD, NN>	<STOP>
	NNP	VBZ	NN	NNS	CD	NN	.
	Fed	raises	interest	rates	0.5	percent	.

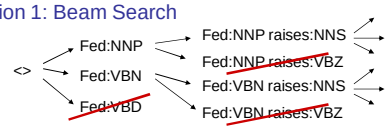
P(NNP|<◆,◆>) P(Fed|NNP) P(VBZ|<NNP,◆>) P(raises|VBZ) P(NN|VBZ,NNP).....

- In principle, we're done – list all possible tag sequences, score each one, pick the best one (the Viterbi state sequence)

NNP VBZ NN NNS CD NN ⇨ logP = -23
NNP NNS NN NNS CD NN ⇨ logP = -29
NNP VBZ VB NNS CD NN ⇨ logP = -27

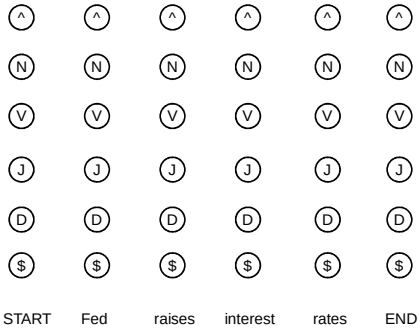
Finding the Best Trajectory

- Too many trajectories (state sequences) to list
- Option 1: Beam Search

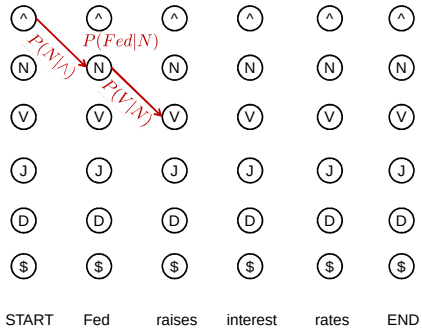


  - A beam is a set of partial hypotheses
  - Start with just the single empty trajectory
  - At each derivation step:
    - Consider all continuations of previous hypotheses
    - Discard most, keep top k, or those within a factor of the best
- Beam search works ok in practice
  - … but sometimes you want the optimal answer
  - … and you need optimal answers to validate your beam search
  - … and there's usually a better option than naïve beams

The State Lattice / Trellis



The State Lattice / Trellis



The Viterbi Algorithm

- Dynamic program for computing

$$\delta_i(s) = \max_{s_0 \ldots s_{i-1} s} P(s_0 \ldots s_{i-1} s, w_1 \ldots w_{i-1})$$

  - The score of a best path up to position i ending in state s

$$\delta_0(s) = \begin{cases} 1 & if\ s = <\bullet,\bullet> \\ 0 & otherwise \end{cases}$$

$$\delta_i(s) = \max_{s'} P(s \mid s') P(w \mid s') \delta_{i-1}(s')$$

  - Also can store a backtrace (but no one does)

$$\psi_i(s) = \arg\max_{s'} P(s \mid s') P(w \mid s') \delta_{i-1}(s')$$

- Memoized solution
- Iterative solution

So How Well Does It Work?

- Choose the most common tag
  - 90.3% with a bad unknown word model
  - 93.7% with a good one
- TnT (Brants, 2000):
  - A carefully smoothed trigram tagger
  - Suffix trees for emissions
  - 96.7% on WSJ text (SOA is ~97.5%)
- Noise in the data
  - Many errors in the training and test corpora

DT NN IN NN VBD NNS VBD
The average of interbank offered rates plummeted …

  - Probably about 2% guaranteed error from noise (on this data)

JJ JJ NN
chief executive officer

NN JJ NN
chief executive officer

JJ NN NN
chief executive officer

NN NN NN
chief executive officer

Overview: Accuracies

- Roadmap of (known / unknown) accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - TnT (HMM++): 96.2% / 86.0%
  - Maxent P(t|w): 93.7% / 82.6%
  - MEMM tagger: 96.9% / 86.9%
  - Cyclic tagger: 97.2% / 89.0%
  - Upper bound: ~98%

Most errors on unknown words

Common Errors

- Common errors [from Toutanova & Manning 00]

	JJ	NN	NNP	NNPS	RB	RP	IN	VB	VBD	VBN	VBP	Total
JJ	0	**177**	**56**	0	**61**	2	5	10	15	**108**	0	488
NN	**244**	0	**103**	0	12	1	1	29	5	6	19	525
NNP	**107**	**106**	0	**132**	5	0	7	5	1	2	0	427
NNPS	1	0	**110**	0	0	0	0	0	0	0	0	142
RB	**72**	21	7	0	0	16	**138**	1	0	0	0	295
RP	0	0	0	0	**39**	0	**65**	0	0	0	0	104
IN	11	0	1	0	**169**	**103**	0	1	0	0	0	323
VB	17	**64**	9	0	2	0	1	0	4	7	**85**	189
VBD	10	5	3	0	0	0	0	3	0	**143**	2	166
VBN	**101**	3	3	0	0	0	0	3	**108**	0	1	221
VBP	5	34	3	1	1	0	2	**49**	6	3	0	104
Total	626	536	348	144	317	122	279	102	140	269	108	3651

NN/JJ NN
official knowledge

VBD RP/IN DT NN
made up the story

RB VBD/VBN NNS
recently sold shares

Corpus-Based MT

Modeling correspondences between languages

Sentence-aligned parallel corpus:

Yo lo haré mañana / I will do it tomorrow
Hasta pronto / See you soon
Hasta pronto / See you around

Machine translation system:

Yo lo haré pronto → Model of translation → I will do it soon / I will do it around / See you tomorrow

Novel Sentence

Phrase-Based Translation Overview

Input: lo haré | rápidamente | .

Translations: I'll do it | quickly | .

quickly | I'll do it | .

The decoder...

tries different segmentations,

translates phrase by phrase,

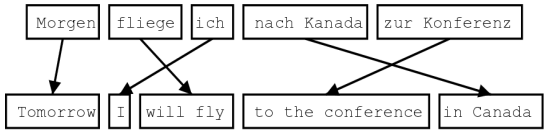
and considers reorderings.

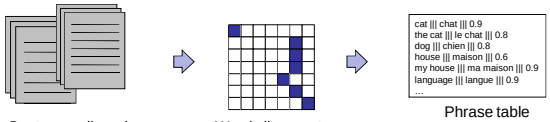
Objective:

$$\arg\max_{\mathbf{e}} [P(\mathbf{f}|\mathbf{e}) \cdot P(\mathbf{e})]$$

$$\arg\max_{\mathbf{e}} \left[\prod_{\langle \bar{e}, \bar{f} \rangle} P(\bar{f}|\bar{e}) \cdot \prod_{i=1}^{|\mathbf{e}|} P(e_i|e_{i-1}, e_{i-2}) \right]$$

Phrase-Based Systems





Sentence-aligned corpus

Word alignments

Phrase table (translation model)

Many slides and examples from Philipp Koehn or John DeNero

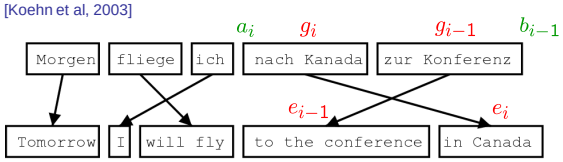
Phrase-Based Decoding

这 | 7人 | 中包括 | 来自 | 法国 | 和 | 俄罗斯 | 的 | 宇航 | 员 | .

Decoder design is important: [Koehn et al. 03]

The Pharaoh "Model"

[Koehn et al, 2003]



$$P(e|g) = P(\{\bar{g}_i\}|g) \prod_i \phi(\bar{e}_i|\bar{g}_i) d(a_i - b_{i-1})$$

Segmentation | Translation | Distortion

The Pharaoh "Model"

$$P(f|e) = P(\{\bar{e}_i\}|e) \prod_i \phi(\bar{f}_i|\bar{e}_i) d(a_i - b_{i-1})$$

$\frac{1}{K}$ | $\frac{count(\bar{f}_i, \bar{e}_i)}{count(\bar{e}_i)}$ | $\alpha^{|a_i - b_{i-1}|}$

Where do we get these counts?

Phrase Weights

How the MT community estimates $P(\bar{f}|\bar{e})$

Parallel training sentences — *provide phrase pair counts.*

Gracias , lo haré de muy buen grado .
Thank you , I shall do so gladly .

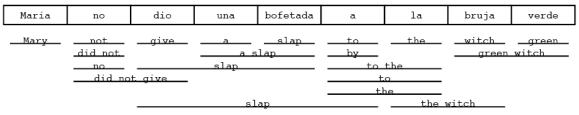
lo haré ⇔ I shall do so
44 times in the corpus

All phrase pairs are counted, — *and counts are normalized.*

Gracias lo haré de muy buen grado .
Thank you I shall do so gladly .

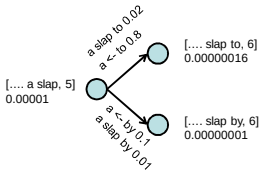
$$P(\bar{f}|\bar{e}) = \frac{\text{count}(\bar{f}, \bar{e})}{\text{count}(\bar{e})}$$

Phrase-Based Decoding



Monotonic Word Translation

- Cost is LM * TM
- It's an HMM?
  - $P(e|e_{-1},e_{-2})$
  - P(f|e)
- State includes
  - Exposed English
  - Position in foreign
- Dynamic program loop?



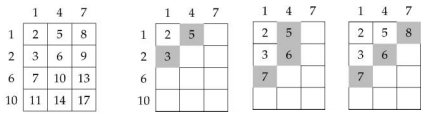

```
for (fPosition in 1…|f|)
  for (eContext in allEContexts)
    for (eOption in translations[fPosition])
      score = scores[fPosition-1][eContext] * LM(eContext) * TM(eOption, fWord[fPosition])
      scores[fPosition][eContext[2]+eOption] =max score
```

Beam Decoding

- For real MT models, this kind of dynamic program is a disaster (why?)
- Standard solution is beam search: for each position, keep track of only the best k hypotheses

```
for (fPosition in 1…|f|)
  for (eContext in bestEContexts[fPosition])
    for (eOption in translations[fPosition])
      score = scores[fPosition-1][eContext] * LM(eContext) * TM(eOption, fWord[fPosition])
      bestEContexts.maybeAdd(eContext[2]+eOption, score)
```

- Still pretty slow… why?
- Useful trick: cube pruning (Chiang 2005)



Example from David Chiang

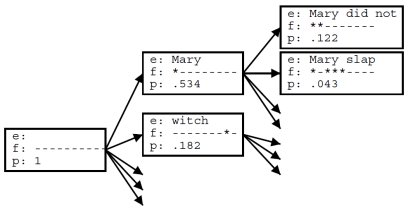
Phrase Translation

- If monotonic, almost an HMM; technically a semi-HMM

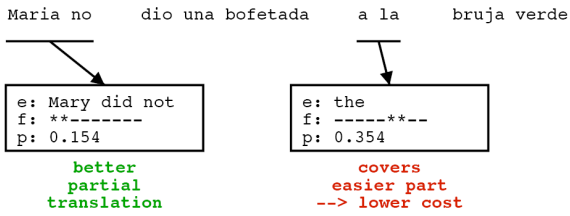
```
for (fPosition in 1…|f|)
  for (lastPosition < fPosition)
    for (eContext in eContexts)
      for (eOption in translations[fPosition])
        … combine hypothesis for (lastPosition ending in eContext) with eOption
```

- If distortion… now what?

Non-Monotonic Phrasal MT

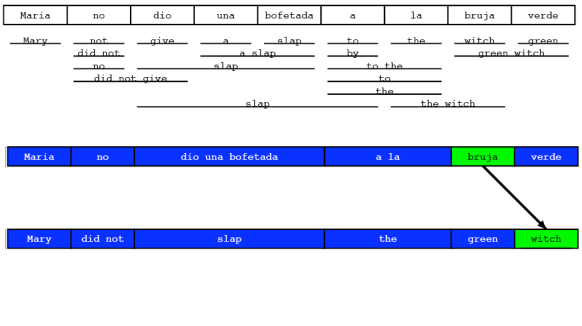


Pruning: Beams + Forward Costs



- Problem: easy partial analyses are cheaper
  - Solution 1: use beams per foreign subset
  - Solution 2: estimate forward costs (A*-like)

The Pharaoh Decoder



Hypotheis Lattices