What is the internet?

- http://www.youtube.com/watch?v=JUs7iG1mNjI

NATURAL LANGUAGE LEARNING: EM

David Kauchak
CS159 – Spring 2011

some slides adapted from Dan Klein

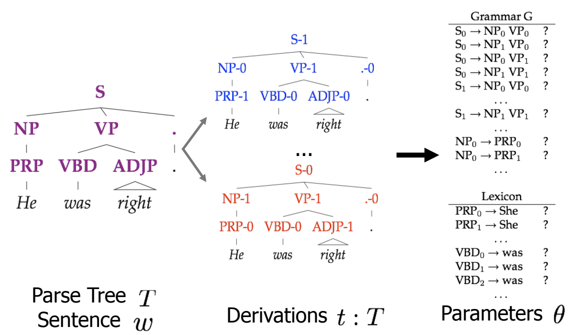
Admin

- My laptop
- Assignment 2 grading
- Assignment 3 out
  - Due Friday at 6pm
  - packages: submit code in package structure
    - in code:
      - nlp/parser/*.java
- Read the book!

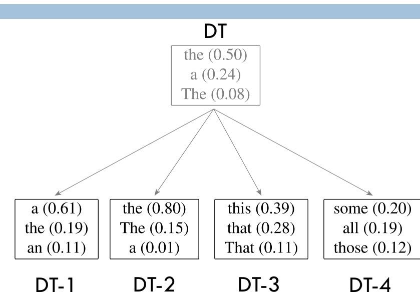
Parsing other languages

- http://nlp.stanford.edu/software/lex-parser.shtml
  - German
  - Chinese
  - Arabic
- Most parsers can be retrained as long as you have a Treebank
  - Czech
  - Korean
    - http://www.cis.upenn.edu/~xtag/koreantag/

Learning good splits: Latent Variable Grammars



Refinement of the DT tag



Learned Splits

- Proper Nouns (NNP):

NNP-14	Oct.	Nov.	Sept.
NNP-12	John	Robert	James
NNP-2	J.	E.	L.
NNP-1	Bush	Noriega	Peters
NNP-15	New	San	Wall
NNP-3	York	Francisco	Street

- Personal pronouns (PRP):

PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him

Learned Splits

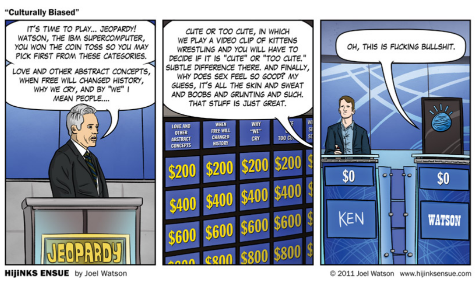
- Relative adverbs (RBR):

RBR-0	further	lower	higher
RBR-1	more	less	More
RBR-2	earlier	Earlier	later

- Cardinal Numbers (CD):

CD-7	one	two	Three
CD-4	1989	1990	1988
CD-11	million	billion	trillion
CD-0	1	50	100
CD-3	1	30	31
CD-9	78	58	34

A step back: data



http://hijinksensue.com/2011/02/15/culturally-biased/

Why do we need computers for dealing with natural text?

We knew the web was big...

7/25/2008 10:12:00 AM

We've known it for a long time: the web is big. The first Google index in 1998 already had 26 million pages, and by 2000 the Google index reached the one billion mark. Over the last eight years, we've seen a lot of big numbers about how much content is really out there. Recently, even our search engineers stopped in awe about just **how** big the web is these days -- when our systems that process links on the web to find new content hit a milestone: 1 trillion (as in 1,000,000,000,000) unique URLs on the web at once!

How do we find all those pages? We start at a set of well-connected initial pages and follow each of their links to new pages. Then we follow the links on those new pages to even more pages and so on, until we have a huge list of links. In fact, we found even more than 1 trillion individual links, but not all of them lead to unique web pages. Many pages have multiple URLs with exactly the same content or URLs that are auto-generated copies of each other. Even after removing those exact duplicates, we saw a trillion unique URLs, and the number of individual web pages out there is growing by several billion pages per day.

Web is just the start…

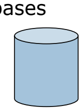
e-mail

247 billion e-mails a day

twitter

27 million tweets a day

corporate databases



Blogs: **126 million** different blogs

http://royal.pingdom.com/2010/01/22/internet-2009-in-numbers/

Corpora examples

- Linguistic Data Consortium
  - http://www.ldc.upenn.edu/Catalog/byType.jsp
- Dictionaries
  - WordNet – 206K English words
  - CELEX2 – 365K German words
- Monolingual text
  - Gigaword corpus
    - 4M documents (mostly news articles)
    - 1.7 trillion words
    - 11GB of data (4GB compressed)

Corpora examples

- Monolingual text continued
  - Enron e-mails
    - 517K e-mails
  - Twitter
  - Chatroom
  - Many non-English resources
- Parallel data
  - ~10M sentences of Chinese-English and Arabic-English
  - Europarl
    - ~1.5M sentences English with 10 different languages

Corpora examples

- Annotated
  - Brown Corpus
    - 1M words with part of speech tag
  - Penn Treebank
    - 1M words with full parse trees annotated
  - Other Treebanks
    - Treebank refers to a corpus annotated with trees (usually syntactic)
    - Chinese: 51K sentences
    - Arabic: 145K words
    - many other languages…
    - BLIPP: 300M words (automatically annotated)

Corpora examples

- Many others…
  - Spam and other text classification
  - Google n-grams
    - 2006 (24GB compressed!)
    - 13M unigrams
    - 300M bigrams
    - ~1B 3,4 and 5-grams
  - Speech
  - Video (with transcripts)

Problem

Labeled

Penn Treebank
1M words with full parse trees annotated

☹

Unlabeled

web

1 trillion web pages

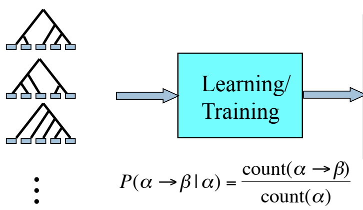
e-mail

247 billion e-mails a day

Learning a grammar

Parsed sentences

Grammar



S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

Parsing other data sources

What if we wanted to parse sentences from the web?

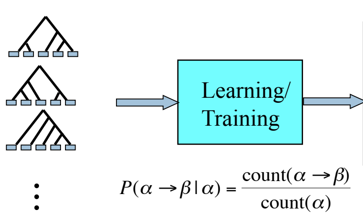
web

1 trillion web pages

Idea 1

Penn Treebank

Penn Grammar



S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

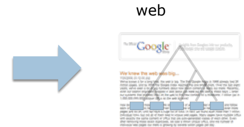
Idea 1

Penn Grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

web



1 trillion web pages

How well will this work?

Parsing other data sources

What if we wanted to parse "sentences" from twitter?

twitter

27 million tweets a day

Idea 1

Penn Grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

twitter

27 million tweets a day

Probably not going to work very well

Ideas?

Idea 2

twitter

Learning/ Training

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

Pseudo-Twitter grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

Idea 2

Pseudo-Twitter grammer

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

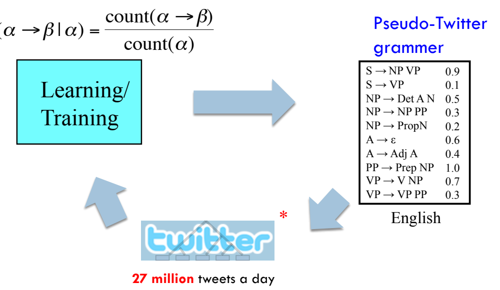
twitter *

27 million tweets a day

Often, this improves the parsing performance

Idea 3

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$



Idea 3: some things to think about

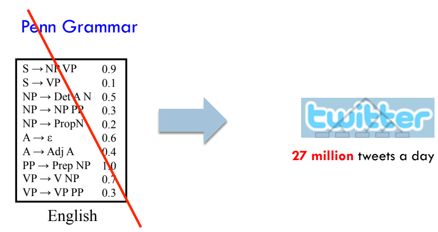
- How many iterations should we do it for?
  - When should we stop?
- Will we always get better?
- What does "get better" mean?

Idea 3: some things to think about

- How many iterations should we do it for?
  - We should keep iterating as long as we improve
- Will we always get better?
  - Not guaranteed for most measures
- What does "get better" mean?
  - Use our friend the development set
  - Does it increase the likelihood of the training data

Idea 4

What if we don't have any parsed data?



Idea 4

Randomly initialized grammar

S → NP VP	?
S → VP	?
NP → Det A N	?
NP → NP PP	?
NP → PropN	?
A → ε	?
A → Adj A	?
PP → Prep NP	?
VP → V NP	?
VP → VP PP	?

English

Pseudo-random

27 million tweets a day

Idea 4

Pseudo-random

Learning/ Training

Pseudo-Twitter grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

Idea 4

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

Learning/ Training

Pseudo-Twitter grammer

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

*

27 million tweets a day

Idea 4

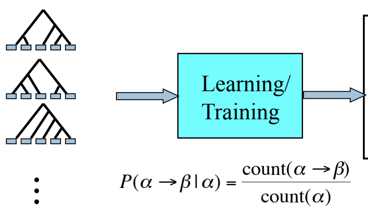
- Viterbi approximation of EM
  - Fast
  - Works ok (but we can do better)
  - Easy to get biased based on initial randomness
- What information is the Viterbi approximation throwing away?
  - We're somewhat randomly picking the best parse
  - We're ignoring all other possible parses
  - Real EM takes these into account

A digression

Why is this called Maximum Likelihood Estimation (MLE)?

Parsed sentences

Grammar



$S \rightarrow NP\ VP$	0.9
$S \rightarrow VP$	0.1
$NP \rightarrow Det\ A\ N$	0.5
$NP \rightarrow NP\ PP$	0.3
$NP \rightarrow PropN$	0.2
$A \rightarrow \varepsilon$	0.6
$A \rightarrow Adj\ A$	0.4
$PP \rightarrow Prep\ NP$	1.0
$VP \rightarrow V\ NP$	0.7
$VP \rightarrow VP\ PP$	0.3

English

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

MLE

- Maximum likelihood estimation picks the values for the model parameters that maximize the likelihood of the training data

parameters

$S \rightarrow NP\ VP$	0.9
$S \rightarrow VP$	0.1
$NP \rightarrow Det\ A\ N$	0.5
$NP \rightarrow NP\ PP$	0.3
$NP \rightarrow PropN$	0.2
$A \rightarrow \varepsilon$	0.6
$A \rightarrow Adj\ A$	0.4
$PP \rightarrow Prep\ NP$	1.0
$VP \rightarrow V\ NP$	0.7
$VP \rightarrow VP\ PP$	0.3

parameter values

model (Θ)

MLE

- Maximum likelihood estimation picks the values for the model parameters that maximize the likelihood of the training data

parameters parameter values

$S \rightarrow NP\ VP$	0.9
$S \rightarrow VP$	0.1
$NP \rightarrow Det\ A\ N$	0.5
$NP \rightarrow NP\ PP$	0.3
$NP \rightarrow PropN$	0.2
$A \rightarrow \varepsilon$	0.6
$A \rightarrow Adj\ A$	0.4
$PP \rightarrow Prep\ NP$	1.0
$VP \rightarrow V\ NP$	0.7
$VP \rightarrow VP\ PP$	0.3

model (Θ)

$$MLE(data) = \underset{\theta}{\arg\max}\, p_\theta(data)$$
$$= \underset{\theta}{\arg\max} \prod_i p_\theta(data_i)$$
$$= \underset{\theta}{\arg\max} \log(\sum_i p_\theta(data_i))$$

If this is what you want to optimize, you can do NO BETTER than MLE!

MLE example

- You flip a coin 100 times. 60 times you get heads.
- What is the MLE for heads?
  - p(head) = 0.60
- What is the likelihood of the data under this model (each coin flip is a data point)?

$$likelihood(data) = \prod_i p_\theta(data_i)$$

$\log(0.60^{60} * 0.40^{40}) = -67.3$

MLE Example

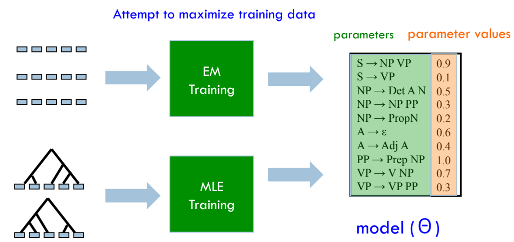
- Can we do any better?

$$likelihood(data) = \prod_i p_\theta(data_i)$$

- p(heads) = 0.5
  - $\log(0.50^{60} * 0.50^{40}) = -69.3$
- p(heads) = 0.7
  - $\log(0.70^{60} * 0.30^{40}) = -69.5$

Expectation Maximization (EM)

- EM also tries to maximized the likelihood of the training data
  - EM works without labeled training data, though!
- However, because we don't have labeled data, we cannot calculate the exact solution in closed form



EM is a general framework

- Create an initial model, θ'
  - Arbitrarily, randomly, or with a small set of training examples
- Use the model θ' to obtain another model θ such that

$$\sum_i \log P_\theta(data_i) > \sum_i \log P_{\theta'}(data_i)$$

i.e. better models data (increased log likelihood)

- Let θ' = θ and repeat the above step until reaching a local maximum
  - Guaranteed to find a better model after each iteration

Where else have you seen EM?

EM shows up all over the place

- Training HMMs (Baum-Welch algorithm)
- Learning probabilities for Bayesian networks
- EM-clustering
- Learning word alignments for language translation
- Learning Twitter friend network
- Genetics
- Finance
- Anytime you have a model and unlabeled data!

E and M steps: creating a better model

Expectation: Given the current model, figure out the expected probabilities of the each example

$p(x|\theta_c)$

What is the probability of each point belonging to each cluster?

What is the probability of sentence being grammatical?

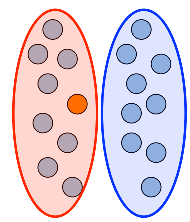
Maximization: Given the probabilities of each of the examples, estimate a new model, θ_c

Just like maximum likelihood estimation, except we use fractional counts instead of whole counts

EM clustering

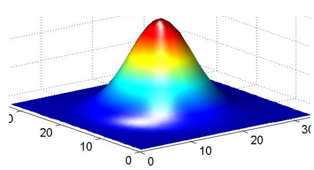
- We have some points in space
- We would like to put them into some known number of groups (e.g. 2 groups/clusters)
- Soft-clustering: rather than explicitly assigning a point to a group, we'll probabilistically assign it

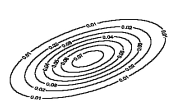
P(red) = 0.75
P(blue) = 0.25



EM clustering
Model: mixture of Gaussians

$$N[x;\mu,\Sigma] = \frac{1}{(2\pi)^{d/2}\sqrt{\det(\Sigma)}}\exp[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)]$$





Covariance determines the shape of these contours

- Fit these Gaussian densities to the data, one per cluster

E and M steps: creating a better model

Expectation: Given the current model, figure out the expected probabilities of the data points to each cluster

$p(x|\theta_c)$ What is the current probability of each point belonging to each cluster?

Maximization: Given the probabilistic assignment of all the points, estimate a new model, θ_c

Do MLE of the parameters (i.e. Gaussians), but use fractional counts based on probabilities (i.e. $p(x \mid \Theta_c)$)

EM example

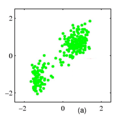


Figure from Chris Bishop

EM example

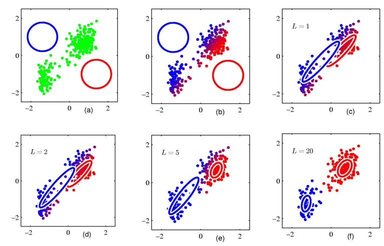


Figure from Chris Bishop

EM for parsing (Inside-Outside algorithm)

Expectation: Given the current model, figure out the expected probabilities of the each example

$p(x \mid \theta_c)$ What is the probability of sentence being grammatical?

Maximization: Given the probabilities of each of the examples, estimate a new model, θ_c

Just like maximum likelihood estimation, except we use fractional counts instead of whole counts

Expectation step

$p(\text{sentence})_{\text{grammar}}$

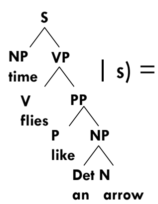
$p(\textit{time flies like an arrow})_{\text{grammar}} = ?$

Note: This is the language modeling problem

Expectation step

p(*time flies like an arrow*)$_{grammar}$ = ?

Most likely parse?

p(

| s) = p(s → NP VP | s) * p(NP → time | NP)

* p(VP → V PP | VP)

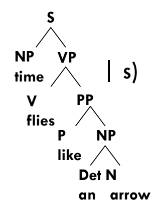
* p(V → flies | V) * …

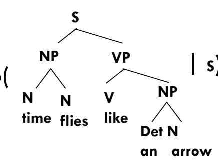
Expectation step

p(*time flies like an arrow*)$_{grammar}$ = ?

Sum over all the possible parses!

Often, we really want: p(*time flies like an arrow* | S)

p(

| s) + p(

| s) + …

Expectation step

p(*time flies like an arrow*)$_{grammar}$ = ?

Sum over all the possible parses!

Often, we really want: p(*time flies like an arrow* | S)

how can we calculate this sum?

Expectation step

p(*time flies like an arrow*)$_{grammar}$ = ?

Sum over all the possible parses!

Often, we really want: p(*time flies like an arrow* | S)

CKY parsing except sum over possible parses instead of max

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S:.05*.5* .000864 =.0000216

S:.03*.0135* .032 =.00001296

For any entry, sum over the possibilities!

$S \rightarrow VP\ PP \quad 0.03$

$S \rightarrow Verb\ NP\ 0.05$

Maximization step

- Calculate the probabilities of the grammar rules using partial counts

MLE

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

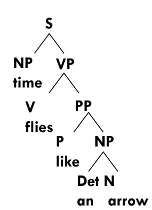
EM

?

Maximization step

Say we're trying to figure out VP -> V PP

MLE



count this as one occurrence

EM

$$p(VP \rightarrow V\ PP \mid \text{time flies like an arrow, S})$$

fractional count based on the sentence and how likely the sentence is to be grammatical

Maximization step

$$p(VP \rightarrow V\ PP \mid \text{time flies like an arrow, S})$$

$$= \frac{p(VP \rightarrow V\ PP, \text{time flies like an arrow} \mid \text{S})}{\text{p(time flies like an arrow} \mid \text{S)}}$$

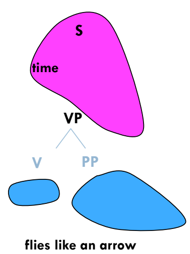
def. of conditional probability

$$= \frac{p(VP \rightarrow V\ PP)\,p(\text{time VP} \mid \text{S})\ \text{p(left - side} \mid \text{V)}\ \text{p(right - side} \mid \text{PP)}}{\text{p(time flies like an arrow} \mid \text{S)}}$$

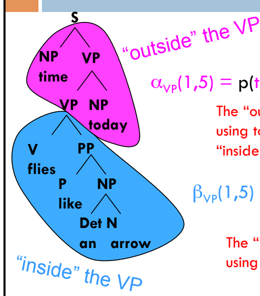
conditional independence as specified by the PCFG

Maximization step

$$\frac{p(VP \rightarrow V\ PP)\,\text{p(time VP | S) p(left - side | V) p(right - side | PP)}}{\text{p(time flies like an arrow | S)}}$$



Inside & Outside Probabilities



$\alpha_{VP}(1,5)$ = p(time VP today | S)

The "outside" probabilities we can calculate using top-down approach (after we have the "inside" probabilities

$\beta_{VP}(1,5)$ = p(flies like an arrow | VP)

The "inside" probabilities we can calculate using a CKY-style, bottom-up approach

EM grammar induction

- The good:
  - We learn a grammar
  - At each step we're guaranteed to increase (or keep the same) the likelihood of the training data
- The bad
  - Slow: $O(m^3n^3)$, where m = sentence length and n = non-terminals in the grammar
  - Lot's of local maxima
  - Often have to use more non-terminals in the grammar than are theoretically motivated (often ~3 times)
  - Often non-terminals learned have no relation to traditional constituents

But…

- If we bootstrap and start with a reasonable grammar, we can often obtain very interesting results

Penn Grammar

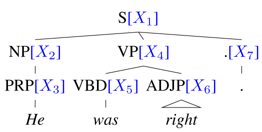
$S \rightarrow NP\ VP$	0.9
$S \rightarrow VP$	0.1
$NP \rightarrow Det\ A\ N$	0.5
$NP \rightarrow NP\ PP$	0.3
$NP \rightarrow PropN$	0.2
$A \rightarrow \varepsilon$	0.6
$A \rightarrow Adj\ A$	0.4
$PP \rightarrow Prep\ NP$	1.0
$VP \rightarrow V\ NP$	0.7
$VP \rightarrow VP\ PP$	0.3

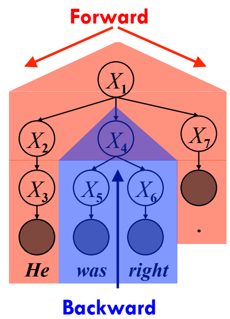
English

Learning Latent Annotations

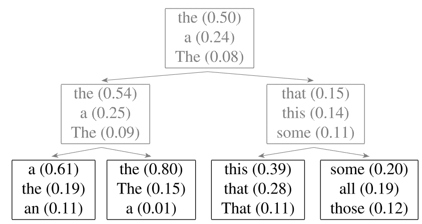
EM algorithm:

- Brackets are known
- Base categories are known
- Only induce subcategories

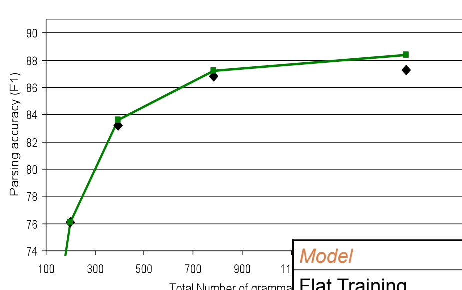




Hierarchical refinement



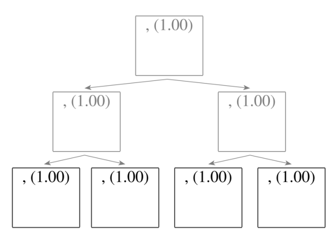
Hierarchical Estimation Results



Model	F1
Flat Training	87.3
Hierarchical Training	88.4

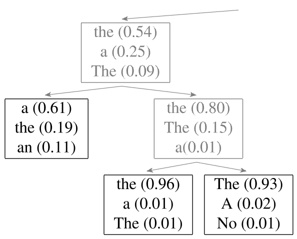
Refinement of the , tag

- Splitting all categories equally is wasteful:

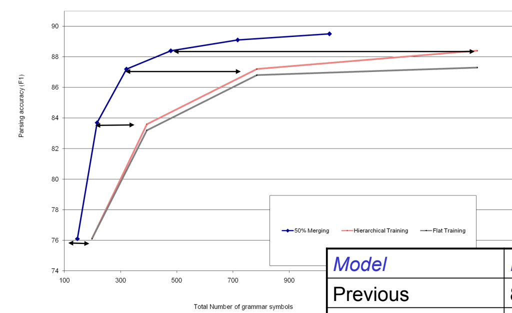


Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful

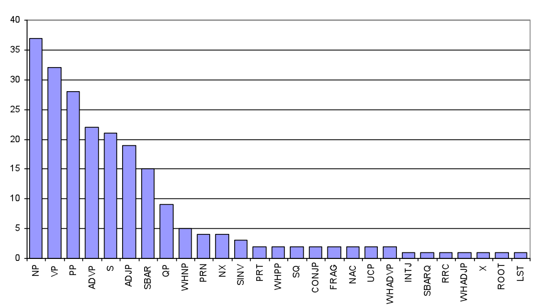


Adaptive Splitting Results

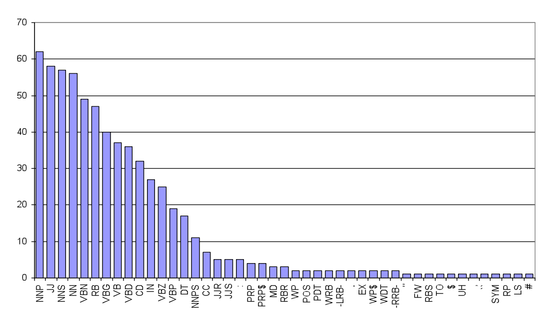


Model	*F1*
Previous	88.4
With 50% Merging	89.5

Number of Phrasal Subcategories



Number of Lexical Subcategories



Final Results (Accuracy)

		≤ 40 words F1	all F1
ENG	Charniak&Johnson '05 (generative)	90.1	89.6
ENG	**Split / Merge**	**90.6**	**90.1**
GER	Dubey '05	76.3	-
GER	**Split / Merge**	**80.8**	**80.1**
CHN	Chiang et al. '02	80.0	76.6
CHN	**Split / Merge**	**86.3**	**83.4**

Still higher numbers from reranking / self-training methods

Finding Word Alignments

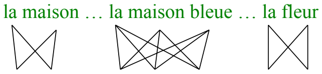
… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

- In machine translation, we train from pairs of translated sentences
- Often useful to know how the words align in the sentences
- Use EM!
  - learn a model of P(french-word | english-word)

Finding Word Alignments

… la maison … la maison bleue … la fleur …



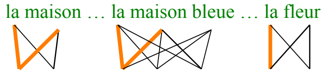
… the house … the blue house … the flower …

All word alignments equally likely

All P(french-word | english-word) equally likely

Finding Word Alignments

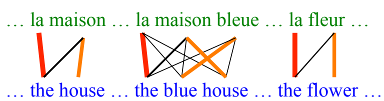
… la maison … la maison bleue … la fleur …



… the house … the blue house … the flower …

"la" and "the" observed to co-occur frequently,
so P(la | the) is increased.

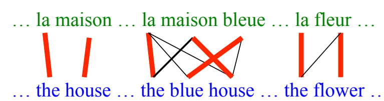
Finding Word Alignments



"house" co-occurs with both "la" and "maison", but P(maison | house) can be raised without limit, to 1.0, while P(la | house) is limited because of "the"

(pigeonhole principle)

Finding Word Alignments



settling down after another iteration

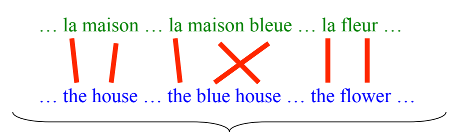
Finding Word Alignments

Inherent hidden structure revealed by EM training!

For details, see

- "A Statistical MT Tutorial Workbook" (Knight, 1999).
  - 37 easy sections, final section promises a free beer.
- "The Mathematics of Statistical Machine Translation" (Brown et al, 1993)
- Software: GIZA++

Statistical Machine Translation



P(maison | house) = 0.411
P(maison | building) = 0.027
P(maison | manson) = 0.020
…

Estimating the model from training data

EM summary

- EM is a popular technique in NLP
- EM is useful when we have lots of unlabeled data
  - we may have some labeled data
  - or partially labeled data
- Broad range of applications
- Can be hard to get it right, though…

Discriminative Parse Reranking

Grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

CKY → best parse

Discriminative Parse Reranking

Grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

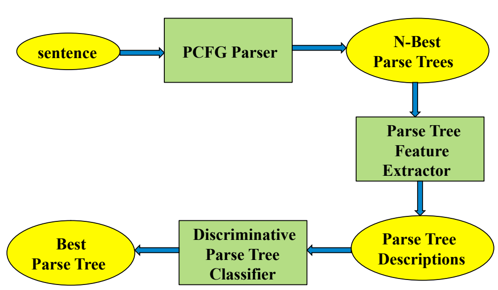
English

CKY →

best parse
2^{nd} best parse
3^{rd} best parse
4^{th} best parse
5^{th} best parse
….

How could we do this?
How might this help us?

Parse Reranking



Sample Parse Tree Features

- Probability of the parse from the PCFG.
- The number of parallel conjuncts.
  - "the bird in the tree and the squirrel on the ground"
  - "the bird and the squirrel in the tree"
- The degree to which the parse tree is right branching.
  - English parses tend to be right branching (cf. parse of "Book the flight through Houston")
- Frequency of various tree fragments, i.e. specific combinations of 2 or 3 rules.

2-Stage Reranking Approach

- Adapt the PCFG parser to produce an ***N-best list*** of the most probable parses in addition to the most-likely one.
- Extract from each of these parses, a set of global features that help determine if it is a good parse tree.
- Train a discriminative classifier (e.g. logistic regression) using the best parse in each N-best list as positive and others as negative.

Evaluation of Reranking

- Reranking is limited by ***oracle accuracy***, i.e. the accuracy that results when an omniscient oracle picks the best parse from the N-best list.
- Typical current oracle accuracy is around F_1=97%
- Reranking can generally improve test accuracy of current PCFG models a percentage point or two

Other Discriminative Parsing

- There are also parsing models that move from generative PCFGs to a fully discriminative model, e.g. ***max margin parsing*** (Taskar *et al.*, 2004).
- There is also a recent model that efficiently reranks all of the parses in the complete (compactly-encoded) parse forest, avoiding the need to generate an N-best list (***forest reranking***, Huang, 2008).

Human Parsing

- How do humans do it?
- How might you try and figure it out computationally/ experimentally?

Human parsing

- Read these sentences
- Which one was fastest/slowest?

John put the dog in the pen with a lock.

John carried the dog in the pen with a bone in the car.

John liked the dog in the pen with a bone.

Human Parsing

- Computational parsers can be used to predict human reading time as measured by tracking the time taken to read each word in a sentence.
- Psycholinguistic studies show that words that are more probable given the preceding lexical and syntactic context are read faster.
  - John put the dog in the pen with a lock.
  - John carried the dog in the pen with a bone in the car.
  - John liked the dog in the pen with a bone.
- Modeling these effects requires an ***incremental*** statistical parser that incorporates one word at a time into a continuously growing parse tree.

Human Parsing

- Computational parsers can be used to predict human reading time as measured by tracking the time taken to read each word in a sentence.
- Psycholinguistic studies show that words that are more probable given the preceding lexical and syntactic context are read faster.
  - John put the dog in the pen with a lock.
  - John put the dog in the pen with a bone in the car.
  - John liked the dog in the pen with a bone.
- Modeling these effects requires an ***incremental*** statistical parser that incorporates one word at a time into a continuously growing parse tree.

Garden Path Sentences

- People are confused by sentences that seem to have a particular syntactic structure but then suddenly violate this structure, so the listener is “lead down the garden path”.
  - The horse raced past the barn fell.
    - vs. The horse raced past the barn broke his leg.
  - The complex houses married students.
  - The old man the sea.
  - While Anna dressed the baby spit up on the bed.
- Incremental computational parsers can try to predict and explain the problems encountered parsing such sentences.