Lecture notes by Edward Loper

Course: CIS 530 (Intro to NLP)
Professor: Steven Bird
Institution: University of Pennsylvania

`http://www.cis.upenn.edu/~cis530`

Monday, October 9, 2000

1 Chart Parsing

It's dynamic programming!

1.1 Always use the fundamental and initialization rules:

Fundamental rule

```
If a chart contains:
  i, j, X0: X1 ... ● Xi ... Xn
  j, k, Xi: Y1 ... Ym ●
Then add:
  i, k, X0: X1 ... Xi ● ... Xn
```

i.e., if one arc wants to eat an Xi, and there's an Xi at the appropriate place, then eat it.

Note that nothing is deleted from the chart

Initialization rule ($\mathbf{I_0}$)

Add edges for each word, according to a lexicon. For example, for the word "the" at the beginning of a sentence, add:

```
0, 1, ART: the
```

1.2 Vary the rule invocation strategy

To do top-down or bottom-up, we just add new initialization and recursion rules. $\mathrm{I_{TD}}$, T. D. Rule, B. U. Rule...

Top-down

For top-down, run $\mathrm{I_0}$, then run $\mathrm{I_{TD}}$:

```
For each rule:
  X → X1 ... Xn
Where X is a category that can span the chart (S), add:
  0, 0, X: ● X1 ... Xn
```

i.e., Add a zero-length edge at the beginning of the sentence, that wants to consume a S.

Then run the fundamental rule and the T. D. rule:

```
When adding an edge:
  i, j, X → ... ● Y ...
then, for each rule Y → Y1 ... Yn, add:
  j, j, Y → ● Y1 ... Yn
```

i.e., if we have an edge where we're trying to consume a Y, add zero-length edges that can output a Y.

Bottom-up

First, run $\mathrm{I_0}$. There is no special $\mathrm{I_{BU}}$.

Then run the fundamental rule and the B. U. rule:

```
When adding an edge:
  i, j, Y: Y1 ... Ym •
then for every rule X → Y X2 ... Xn, add:
  i, i, X: • Y X2 ... Xn
```
i.e., if we just found a Y, then look for rules that consume Y's, and add arcs for them.

1.3 Hybrids

Type-sensitive parsing

You can do bottom-up parsing of one type of word (e.g., verbs) and top-down parsing of everything else. Make words type sensitive. E.g., replace bottom up with:

```
When adding an edge:
  i, j, V: Y1 ... Ym •
then for every rule VP → V X2 ... Xn, add:
  i, i, VP: • V X2 ... Xn
```
And top-down with:

```
When adding an edge:
  i, j, X → ... • Y ...
where Y ≠ VP, for each rule Y → Y1 ... Yn, add:
  j, j, Y → • Y1 ... Yn
```
(or something like that)

1.4 Completeness

If we change the rule strategy, can we be sure we'll get everything? If there is a correct parse, are we guaranteed to get it?

2 Improving Grammar Coverage

What about features? Add features to rules? E.g., allow something like:

```
X[± f]
```
and have rules like:

```
X[α f] → ... Y[α F] ...
```

3 Partial Parsing

- chinks and chunks
- tag transitions (NN-VB closes an NP)
- cascade - sequence of strata
- use regexps directly on pos tags: NP = DT JJ* NN

Wednesday, October 18, 2000

4 Dealing With Words

4.1 Meaning

3 Types:

- referential (thing/event/etc)
- social (info about speaker)
- affective (info about speaker's attitudes/feelings)

4.2 Resources

Wordnet

http://www.cogsci.princeton.edu/~wn/

has an API for C.

synsets related to hypernyms, hyponyms, etc. meronym - part/whole (door is meronym of house). holonym - whole/part (house is holonym of door).

Contains LDOCE, Collin's dictionary

CELEX

http://www.kun.nl/celex/

pronunciation lexicon for english, dutch, german. gives pronunciation (ipa) with stress, frequency, part of speech, etc.

4.3 Acquisition

How do we find the words to begin with?

child language acquisition: children need to find word boundries. What techniques do they use? Use stress? ('puter for computer)

exploit distributional regularities (phonotactic constraints) of words..

form word hypotheses, and minimize size(lexicon)+size(encoding of sentences)

Build a trie, or letter tree, or prefix tree. looks like a huffman encoding. (trie is from "retrieval")

http://hissa.nist.gov/dads/HTML/trie.html

4.4 Spelling Errors

Soundex

A way to deal with mis-spellings of known words

1. retain the first letter of the name, and drop all occurances of a, e, h, i, o, u, w, and y in other positions.
2. assign the following numbers to the remaining letters after the first: bfpv$\rightarrow$ 1, cgjkqsxz$\rightarrow$ 2, dt$\rightarrow$ 3, mn$\rightarrow$ 5, r$\rightarrow$ 6

3. if two or more letters with the same code were adjacent in the original name, omit all but the first.
4. convert to the form "letter, digit, digit, digit" by stripping/padding (with zeros)

edward → edrd → e363

Levenshtein Distance

A way to cluster similar words.. Gives a distance measure between any two words.

Alignment: line them up to maximize matched letters:

```
In--dustry
Interest--
in....st..
```

Trace:

```
Industry
  -substitute D by T-
  -shift ...-
  ...
Interest
```

count 1 for insertion/deletion, 2 for substitution:

```
In--dustry
Interest--
0011220011 → Levenshtein distance = 8
```

Every string has k-nearest neighbors.

Levenshtein distance could be weighted.. some substitutions (eg. dt) might be penalized less than others (eg. qr).

Monday, October 23, 2000

5 Statistical NLP

5.1 Motivation

Where might we want a stochastic grammar? i.e., a symbolic grammar with probabilities assocaited with each rule?

- acquisition
- variation
- change
- adult monolingual speaker? must show evidence..

There are unusual word sequences. A broad coverage grammar must assign structures – we simply get too many structures for any decent size sentence.. How do we represent which ones are better..?

Use a weighted grammar instead of parsing preferences.

5.2 Probability

Bayes theorem

```
           P(B|A)P(A)
P(A|B) = --------------
              P(B)
```

Example

```
A = instance of construction
B = program reports a hit
P(A) = 0.001
P(B|A) = 0.9
P(B|~A) = 0.01
P(B) ≈ 0.0109
P(A|B) = P(B|A)P(A)/P(B) ≈ 0.0826
```

Wednesday, October 25, 2000

Make best inference based on the available data and any prior knowledge, and revise our position as new information comes to light.

Prior probabilities: P(hyp)

likelihood function: P(data|hyp) – learn via experiments

Posterior probability: P(hyp|data) – use bayes' theorem

Book: Sivia (1996) Data Analysis: A bayesian Tutorial (oxford university press)

6 Information Theory

6.1 The Noisy Channel Model

- Communicate messages over a channel
- Maximize throughput & accuracy in presence of noise
- Compression vs. accuracy

```
w → encoder -x→ channel -y→ decoder → w'
```

6.2 Entropy

```
H(p) = ∑ p(x) log p(x)
```

Entropy can be thought of as the average length of a message needed to transmit the outcome of some random variable.

Entropy of alphabet where P(p)=P(k)=P(u)=P(i)=1/8 and P(t)=P(a)=1/4:

```
(+ (* 4 (/ 3.0 8)) (* 2 (/ 2.0 4))) = 2.5
```

Relative Entropy

```
D(p\|q) = ∑ p(x) log(p(x)/q(x))
```

Consider the two probability distributions:

	p	t	k	a	u	i
q	1/8	1/4	1/8	1/4	1/8	1/8
p	1/8	1/4	1/8	1/8	1/8	1/4

```
D(p\|q) = (-∑ p(x)log q(x)) - (-∑ p(x)log p(x))
        = ∑ p(x) log(p(x)/q(x))
```

In our example:

```
D(p\|q) = 1/8 log(1/2) + 1/4 log(2)
        = -1/8 + 1/4 = 1/8
```

So the relative entropy D(p\|q) = 1/8. This is the extra message length needed to transmit, on average.

Inefficiency of assuming that the distribution is q(x), when it's really p(x).

Cross Entropy

Perplexity

perplexity = 2^{e}ntropy

7 Markov Models (Ch. 9 Manning)

```
p(xi | xi-1, xi-2, ..., x1, x0)
```

Unless we have a LOT of data, this is pretty sparse.

Sparse data problem.

So approximate:

```
p(xi | xi-1, xi-2, ..., x1, x0)
≈ p(xi)                      ; monogram model
≈ p(xi | xi-1)               ; bigram model
≈ p(xi | xi-1, xi-2)         ; trigram model
```

Bigram model can be trivially converted into a finite state model. A markov model (i.e., FSM with probabilities on transitions).

Try using backoff: use higher order models where possible, lower order models when necessary.

Assign probabilities to:

- nodes (chance of starting there)
- arcs: chance of transitioning to a state
- labels on arc: chance of taking each letter if you take that transition.

```
[0.1] ---⟶ [0.8]
  ↑           | a: 0.4
  |0.7     1.0| b: 0.5
  |   0.3     ↓ c: 0.1
[0.1] ---⟶ [0.0]
```

Wednesday, November 1, 2000

```
a b c d e ε → b c d e f
  split ε among extra chars (f)
```

Defining a Markov Model:

```
S: {state}   set of states
K: {letter}  output alphabet
Π: S→ p      Initial state probabilities
A: S× S→ p    State transition probabilities
B: S× S× K→ p  Output letter probabilities
```

7.1 Finding P(observation)

Use a Trellis. (Forwards algorithm)

```
      t=0    t=1         t=2
s1    0.8    0.126       0.17433
s2    0.3    0.574       0.20993
```

P(ab)=0.3836

(+ (* .8 .3 .5) (* .2 .3 .1)) (+ (* .8 .7 .9) (* .2 .7 .5))

(+ (+ (* .126 .3 .5) (* .574 .3 .9)) (+ (* .574 .7 .5) (* .126 .7 .1)))

Draw Trellis with arrows/probabilities?

```
S1 .8 ---→.32 ---→
      \--/      \--/
      /  \      /  \
S2 .2 ---→.13 ---→
```

7.2 Find most likely path (Viterbi algorithm)

Simply find most likely partial path? Keep back pointer to its most likely preceeding state..

Partial path probability:

$$\delta_j(t+1) = \max_{1\le\ i\le\ n}\ \delta(t) a_{ij} b_{ijo_t}$$

Note that $a_{ij}b_{ijo_t}$ is probability of transitioning from i to j with output o_t..

δ keeps track of the probability of the path ending at the given point..

Backpointers:

$$\psi_j(t+1) = \mathrm{argmax}_{1\le\ i\le\ N}\ \delta(t) a_{ij} b_{ijo_t}$$

(argmax means "give me the i for which the expression is maximized")

7.3 What paramaters A, B, Π maximize P for given observations?

(Expectation Maximization algorithm / Baum-Welsh algorithm)

Iteratively adjust A, B, Π to make training data more likely. Throw a lot of training data at it, keep track of how many times we take different arcs, adjust probabilities accordingly.. Hill-climbing.

8 Collocations

We could simply look for frequent bigrams, but we get a lot of bigrams like ”of the” and ”has been”.. We could use a stop list, but the real problem is that we want to deal with is whether they occur together more often then chance: take into account the frequencies of the individual words.

Consider the following graph:

```
 |  B   W |
-+--------+----
M|  5   5 | 10
F| 15  15 | 30
-+--------+----
 | 20  20 |
```

The probabilities in the squares are exactly what we expect. So χ^2=0.

Formula for expectation:

```
           row i total   col j total
E(i, j) = ------------ × ----------- × total
              total         total
```

```
E(i, j) = P(i)P(j) × total
```

If we observe:

```
 |  B   W |
-+--------+----
M|  4   6 | 10
F| 16  14 | 30
-+--------+----
 | 20  20 |
```

Then the probabilities no longer match our expectations.

Find:

```
(Aij - Eij)²
-------------
    Eij
```

gives:

```
 |  B     W
-+----------
M| 1/5  1/5
F| 1/15 1/15
```

χ^2 is simply the sum of these numbers..

Monday, November 6, 2000

9 Hand Crafted System Notes

9.1 Training & Testing

- sophistication of model
- representativeness of training data
- overfitting
- details of parameterization

9.2 Tradeoffs

- simplicity vs coverage (threshold effects)

9.3 Brittleness

- small changes in earlier parts of regexp can strongly affect later parts.
- overlapping effects: chaotic to maintain

9.4 Not scalable

9.5 Easier to relate to theory

10 Building Language Models

Take data generated by an unknown PDF, and make inferences about that PDF.

1. divide data into equivlanace classes: sparse data problem
2. find estimators for equivelance classes
3. combine multiple estimators

Choice of classificatory features: we're basically dividing data into "bins," where we assume that everything in the bin functions the same.. e.g., when tagging, put all occurances of the same word in the same bin for a 0th order tagger, or all occurances of the same word with the same prior tag in the same bin for a 1st order tagger.

Strict n-gram modelsl have sparse data problems.. use probablistic backoff? Use an HMM.

Thursday, November 9, 2000

11 LDC

LDC maintains:

- data
- programs
- standards and best practices

Members have perpetual rights to each corpus released in the year in which they join.

˜8 full time and 30 part time transcribers.

LDC functions to:

- distribute/publish data
- create corpera
- research (best practices)

11.1 Switchboard Corpus

- telephone recordings
- 2430 conversations @ 6min each
- 3 million words, >500 speakers
- transcribed and word-aligned

11.2 ACE – automatic content extraction

- Identify nominal entries in a news story
- Classify according to type
- Co-index mentions of a single entity within the story.

11.3 TDT

- newswire, broadcast radio, broadcast tv

Tuesday, November 28, 2000

12 Unisys: Natural Language Understanding

(Bharathi Palle)

- IVR: Interactive voice response
- ASR: Automatic speech recognizer
- TTS: Text to speech engines
- NL understanding engines

Types of speech applications:

- command and control
- dictation
- dialogue-driven (mixed initiative)

12.1 ASR

- feature extraction
- phoneme recognition (based on pre-defined acoustic model)
- phoneme-based HMMs/Viterbi search on features for words..
- use grammars to reduce ambiguity/search space

ASR features

- hardware vs. software
- speaker dependant vs independant
- continuous vs isolated vs digit/spelling
- grammar formats
- language models (incl. land-line vs cell-phone)
- vocab size
- n-best
- programming interface

12.2 NLI

- normalize different inputs to a simple grammar, e.g. normalize "I want info" to "GET$_{\mathrm{I}}$NFORMATION"

13 Darpa Communicator (Lockheed Martin)

Apply research projects to DOD-type engineering applications. Middle-man between researchers and govt.. MIT: Gallaxy II? TINA?

staged approach

14 CL across govt, industry, & academia

Birth of "Language Engineering" Plug-and-play

15 Journals

- Computational Linguistics
- Natural Language Understanding
- Studies in NLP

Tuesday, December 5, 2000

16 Exam Topics

- Materials from the table in the course homepage
  - lecture notes
  - readings
  - assignments
  - (no python)
  - (not Brent & Cartwright, articles in parens)
  - Questions involving computation
  - e.g.:
    * Compute relative etropy of x wrt y
    * Apply Baye's theorem
    * Demonstrate Viterbi algorithm for a given HMM & output
    * Use cosine measure, given doc. stats.
    * Compute avg. precision score for ranked retrieval set
    * Questions involving prose
    * e.g.:
    * Why is NLP hard?
    * What is the sparse data problem & hos is it addressed in language modeling?
    * What are collocations, and why is bigram freq. a poor way to find them?
    * Sentence X contains a structural ambiguity – draw tree diagrams and discuss.
    * Suppose you want to solve problem X – how would you integrate components we've talked about to solve the problem?
    * Describe a lexical resource and discuss different ways it could be used in NLP.
    * How can linguistics benefit from statistical NLP?
    * Questions Between

17 Humanitarian applications of NLP

- ubiquitous computing (?)
- second language learning
- alternative input/HCI
- language documentation/preservation
  - 52% languages spoken by <10,000 people
  - 28% languages spoken by <1,000 people
  - need to documet/preserve languages