Structured Prediction for Natural Language Processing

Noah A. Smith
Assistant Professor
Carnegie Mellon University
nasmith@cs.cmu.edu

June 14, 2009

A Very Long Relationship

Natural language processing (NLP) and **machine learning** (ML) go back to the 1940s.

NLP to ML: "You give me elegant, well-founded solutions to my problems."

ML to NLP: "You give meaning to my math. You come with data."

But we are seeing signs of strain.

Grievances from NLP

- Scalability: “I asked you to use *all* of the data. Why can’t you ever finish a job?”
- Simplistic models: “Stop assuming things!”
- Infidelity: “Why are you always thinking about classification?”

Grievances from ML

- Data incomplete: “Why can’t you just tell me what you want?”
- Evaluation criteria unclear: “You keep changing your mind!”
- Infidelity: “Why are you always thinking about linguistics?”

This marriage can survive, if both parties learn to understand each other better.

This tutorial is meant to provide a bit of marriage counseling to NLP and ML.

Where We're Going

Goals:

- Discuss several linguistic analysis problems that are examples of **structured prediction**.
- Present algorithmic tools used for *making* structured predictions (**decoding**).
- Present the dominant techniques used in NLP for *learning* to make structured predictions from
  - complete data (**supervisedly**) and
  - incomplete data (**unsupervisedly**).

Where We're *Not* Going

I won't be presenting any experimental comparisons.

I won't go into much detail on datasets, annotation conventions, or linguistic theory (see the references).

I won't go into much detail on implementation (features, tricks, data structures)—you won't be able to walk away from this tutorial and start writing code.

Managing Expectations

Hopefully after this tutorial you will:

- Be (even) more excited about NLP as a playground for ML.
- Understand structured prediction better, and have a broader view of what it encompasses.
- Have higher expectations for what (structured) ML should be able to do.

What's Structured Prediction?

Unfortunately, there is no widely agreed-upon definition!

Two versions, proposed by [Daumé, 2006]:

1. Discrete output representable by (collections of) variable-length vectors in $\{0, 1, \ldots, M\}^L$.
2. Additionally, loss function does not decompose into parts.

Here we take the view that "you know it when you see it" and assume that there may be more than one interesting or useful loss function. (Often our loss functions *will* decompose.)

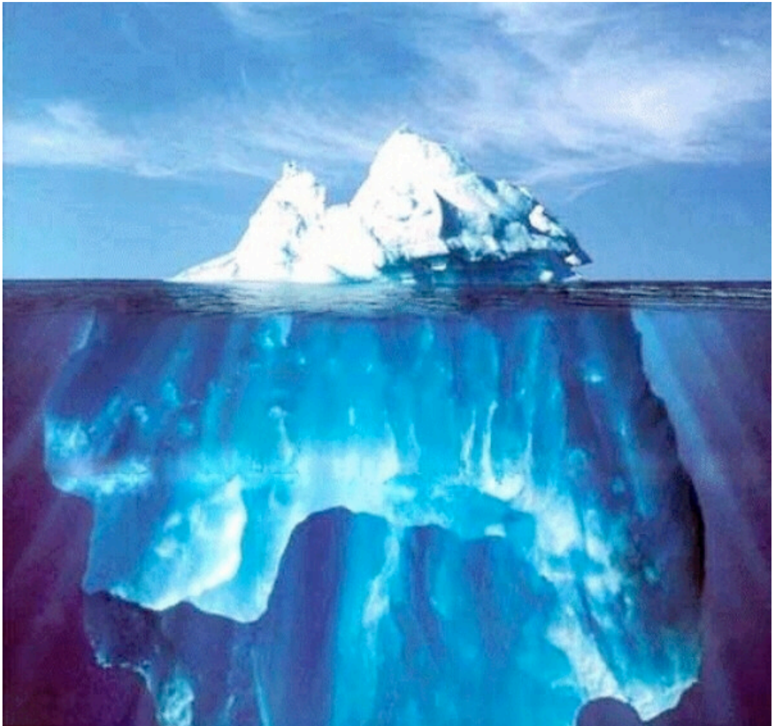
Representations & Data

Don't Think About A Bag of Words

Language and text have structure.

Put another way, the words in a document are not IID.

Linguistics offers many different theories about the relationships among bits of text; we will see some of them.



Some Notation

$\mathcal{X}$: set of possible inputs, often $\mathcal{X} = \Sigma^*$

$\mathcal{Y}$: set of possible *outputs*

Where are the Words?

Problem: segmentation into words (or sentences)

蒙特利尔,又译滿地可,位于魁北克省南部,人口约372万,是加拿大第二大城市。主要使用法语,在法语世界里的地位是仅次于巴黎的第二大城市,也有“小巴黎”的美称。蒙特利尔也是世界最大的双语城市。

[蒙特利尔][,][又][译][滿地可][,][位于][魁北克省][南部][,][人口][约][372万][,][是][加拿大][第二][大][城市][。][主要][使用][法语][,][在][法语][世界][里][的][地位][是][仅][次于][巴黎][的][第二][大][城市][,][也][有[“小巴黎”][的][美称][。][蒙特利尔][也][是][世界][最大][的][双语][城市][。]

Where are the Words?

Problem: segmentation into words (or sentence)s

$\mathcal{X}$: Γ^* (character sequences)

$\mathcal{Y}$: Σ^* (word sequences)

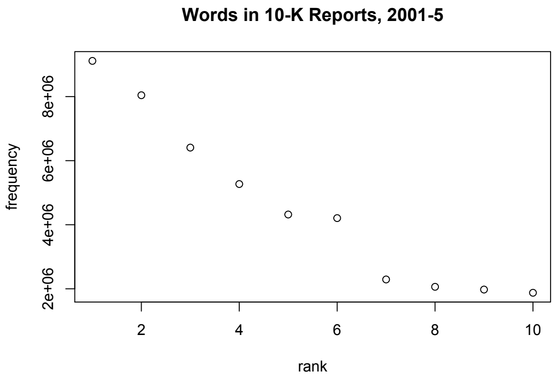
Mostly trivial for English (tokenization), though *sentence* segmentation is a bit harder [Ratnaparkhi, 1996].

The Problem with Words

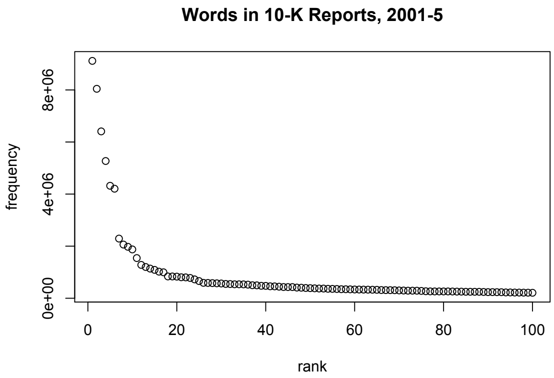
Words proliferate like cockroaches, and they tend to follow Zipfian distributions.

An early step in most language processing is trying to make words more manageable.

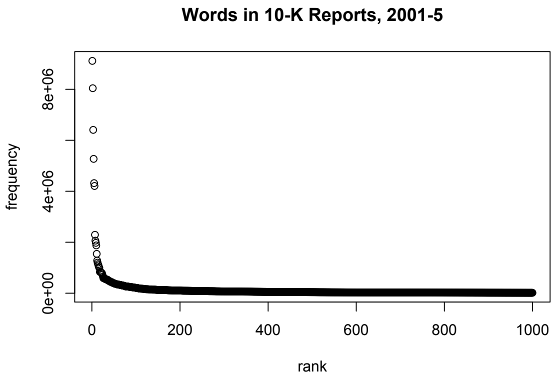
The Problem with Words



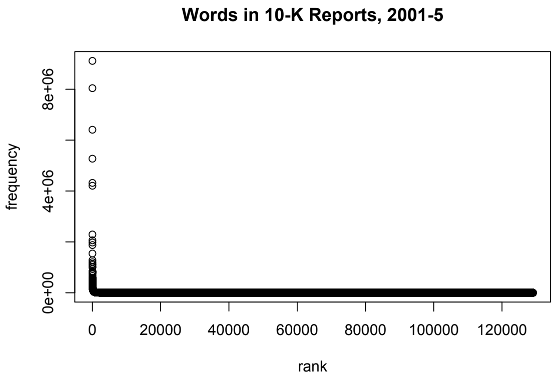
The Problem with Words



The Problem with Words



The Problem with Words



What's in a Word?

Problem: projection into a simpler vocabulary

surface string:

The angle of cats' ears is an important clue to their mood.

tokenized:

the angle of cats ' ears is an important clue to their mood .

stemmed:

the angl of cat ' ear is an import clue to their mood .

lemmatized:

the angle of cat ' ear is an important clue to their mood .

What's in a Word?

$\mathcal{X}$: Σ^* (surface word sequence)

$\mathcal{Y}$: Λ^* (canonical word sequence)

- Stemming: strip suffixes, usually based on some rules [Porter, 1980]
- Lemmatization: reduce words to stems (*gave*, *given*, *give* $\rightarrow$ *give*)

Extreme Tokenization: Parts of Speech

Problem: label each word with its part-of-speech

Det. Noun P. PlN. Pos. PlN. V. Det. Adj. Noun P. Pos.Pr. Noun
The angle of cats ' ears is an important clue to their mood

$\mathcal{X}$: Σ^* (surface word sequence)

$\mathcal{Y}$: Λ^* (part-of-speech tag sequence)

Why *Structure*'s Required

Independent classification of each word, without considering the classes of its neighbors, gets 90% with enough data.

Syntactic *context* is an important clue for disambiguating words.

- *leaves* is probably a verb if it's followed by a proper noun
- *bear* is probably a noun if it's preceded by a determiner

Sequence (i.e., structured) models can achieve around 97% on this task [Toutanova et al., 2003, Shen et al., 2007].

Learning from Data

POS tagging is perhaps the simplest example (and one of the earliest) where it makes sense to have humans label text, then perform supervised learning.

POS conventions are not universal!

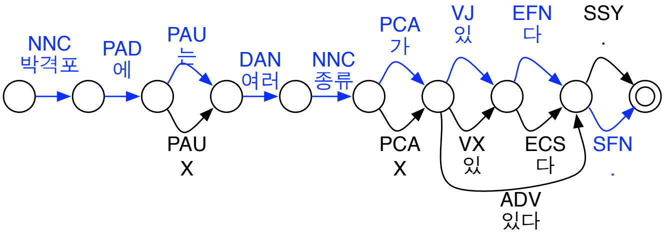
Annotators do not always agree!

Morphology: Dirty Words

If you only speak English, this may come as a surprise.

Problem: morphological disambiguation (breaking words into meaningful morphemes, optionally with tags)

박격포에는 여러 종류가 있다 .



Morphology: Dirty Words

If you only speak English, this may come as a surprise.

Problem: morphological disambiguation (breaking words into meaningful morphemes)

$\mathcal{X}$: Σ^* (surface word sequence)

$\mathcal{Y}$: Λ^* (morpheme sequence)

Agglutinative Morphology

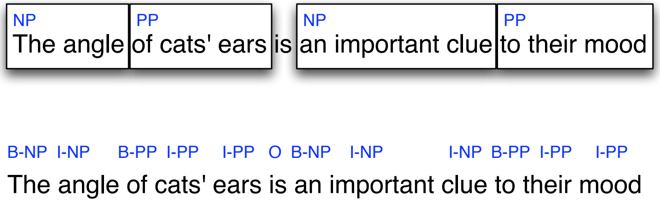
uygarlaştıramadıklarımızdanmışsınızcasına

“(behaving) as if you are among those whom we could not civilize”

(More than 60 million people speak this language.)

Interesting Substrings: Chunks

Problem: find specific kinds of substrings


NP
The angle
PP
of cats' ears
is
NP
an important clue
PP
to their mood
B-NP I-NP B-PP I-PP I-PP O B-NP I-NP I-NP B-PP I-PP I-PP
The angle of cats' ears is an important clue to their mood

Interesting Substrings: Chunks

Problem: find specific kinds of substrings

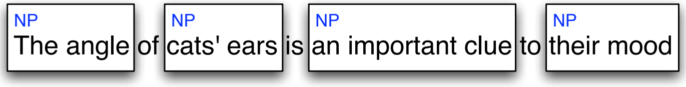
$\mathcal{X}$: Σ^* (sentence)

$\mathcal{Y}$: $((\Lambda \times \{\mathsf{B}, \mathsf{I}\}) \cup \{\mathsf{O}\})^*$ ("I-O-B" labels)

Examples:

- Base noun phrase chunking
- "Shallow parsing" (includes prepositional phrases and verb groups)
- Named entity recognition

Base Noun Phrase Chunking



Named Entity Recognition

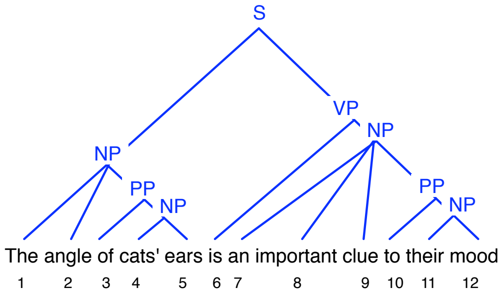
On the streets around Fatemi Square, near the headquarters of the leading opposition candidate, Mir Hussein Moussavi, riot police officers dressed in Robocop gear roared down the sidewalks on motorcycles to disperse and intimidate the clots of pedestrians who gathered to share rumors and dismay.

Named Entity Recognition

On the streets around $\boxed{\text{Fatemi Square}}$, near the headquarters of the leading opposition candidate, $\boxed{\text{Mir Hussein Moussavi,}}$ riot police officers dressed in $\boxed{\text{Robocop}}$ gear roared down the sidewalks on motorcycles to disperse and intimidate the clots of pedestrians who gathered to share rumors and dismay.

Extreme Chunks: Parsing

Problem: find compositional phrases from the whole sentence down to the words



$\langle\langle$S, 1, 12$\rangle$, $\langle$NP, 1, 5$\rangle$, $\langle$PP, 3, 5$\rangle$, $\langle$NP, 4, 5$\rangle$, $\langle$VP, 6, 12$\rangle$, $\langle$NP, 7, 12$\rangle$, $\langle$PP, 10, 12$\rangle$, $\langle$NP, 11, 12$\rangle\rangle$

Extreme Chunks: Parsing

Problem: find compositional phrases from the whole sentence down to the words

$\mathcal{X}$: Σ^* (sentence)

$\mathcal{Y}$: $2^{\Lambda \times \mathbb{N}^2}$ (phrase structure)

Underlying this, often, is some **context-free grammar**.

NL vs. PL

Undergrads who have taken programming languages often want to use the same tools to process natural language. The problem—and the reason NLP is married to ML and not to PL—is that natural language syntax is riddled with ambiguities.

LITTLE HOPE GIVEN BRAIN-DAMAGED WOMAN

LITTLE HOPE GIVEN BRAIN-DAMAGED WOMAN

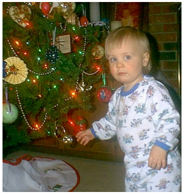
- After Terry's accident, the doctor gave her family the bad news.

LITTLE HOPE GIVEN BRAIN-DAMAGED WOMAN

- After Terry's accident, the doctor gave her family the bad news.
- By September, McCain's campaign staff were not optimistic.

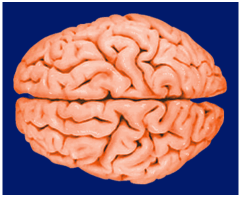
LITTLE HOPE GIVEN BRAIN-DAMAGED WOMAN

- After Terry's accident, the doctor gave her family the bad news.
- By September, McCain's campaign staff were not optimistic.
- It was a strange Christmas for little Hope.



LITTLE HOPE GIVEN BRAIN-DAMAGED WOMAN

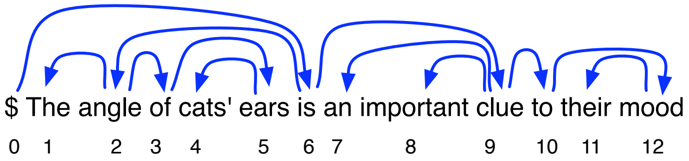
- After Terry's accident, the doctor gave her family the bad news.
- By September, McCain's campaign staff were not optimistic.
- It was a strange Christmas for little Hope.
- If only little Hope had used her gift for good rather than evil ...



Alternative to Phrases: Dependency Parsing

Not everyone agrees on parsing conventions!

Dependency syntax focuses on relationships among words.



$ The angle of cats' ears is an important clue to their mood
0 1 2 3 4 5 6 7 8 9 10 11 12

Alternative to Phrases: Dependency Parsing

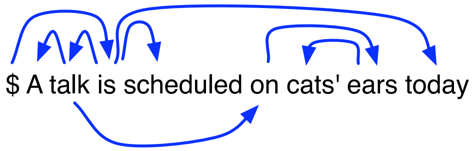
$\mathcal{X}$: Σ^* (sentence)

$\mathcal{Y}$: $\mathbf{y} \in \{0, 1, 2, \ldots, n\} \rightarrow 2^{\{0,1,\ldots,n\}}$

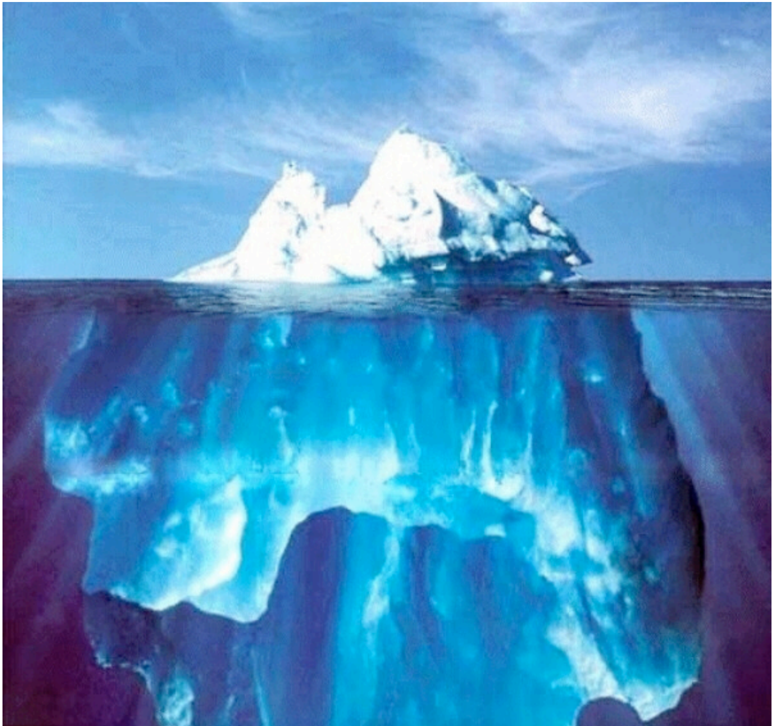
Two Versions of Dependency Parsing

Projective trees correspond to derivations in a special kind of context-free grammar [Gaifman, 1965].

Nonprojective trees don't correspond to CF parses:

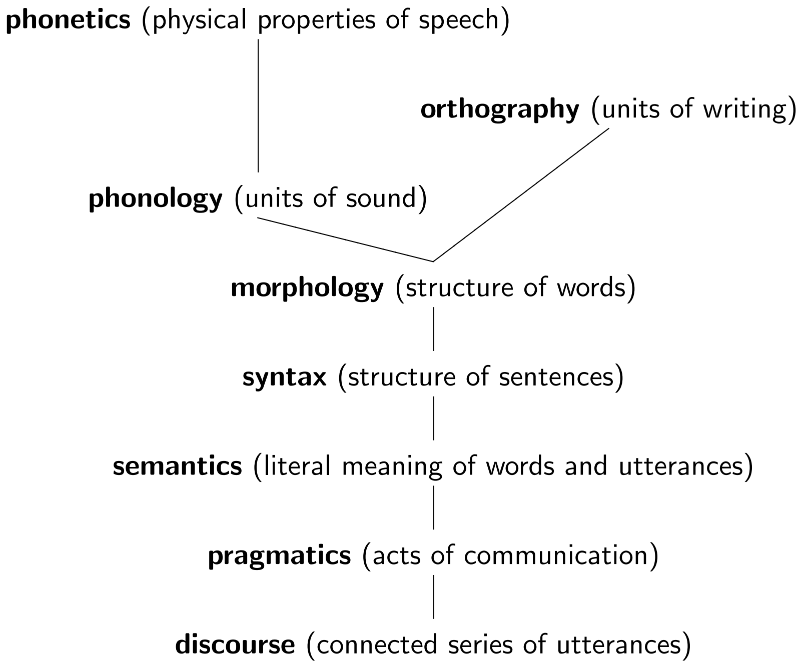


Nonprojectivity is more important in some languages than others.



Interlude: Linguistic Pipeline

As we descend the iceberg, we get closer to the physical, cultural, non-linguistic world where language gets used.

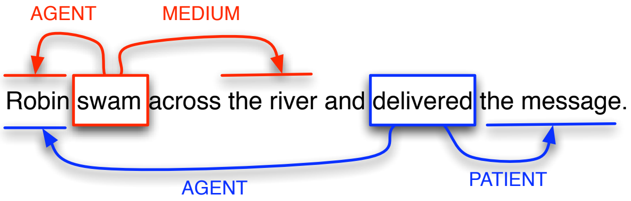

phonetics (physical properties of speech)
orthography (units of writing)
phonology (units of sound)
morphology (structure of words)
syntax (structure of sentences)
semantics (literal meaning of words and utterances)
pragmatics (acts of communication)
discourse (connected series of utterances)

Meaning

Problem: convert a sentence into a canonical meaning representation language

Robin swam across the river and delivered the message.

semantic roles:



first order logic:

∃r, m, p1, p2 SwimTo(Robin, p) ∧ river(r) ∧
Across(r, p1, p2) ∧ Deliver(Robin, m) ∧ Message(m)

Meaning

Problem: convert a sentence into a canonical meaning representation language

$\mathcal{X}$: Σ^* (sentence)

$\mathcal{Y}$: no consensus yet!

- identifying semantic roles of a verb [Palmer et al., 2005]
- first order logic expressions [Zettlemoyer and Collins, 2005]
- problem-specific meaning language [Thompson et al., 1997]

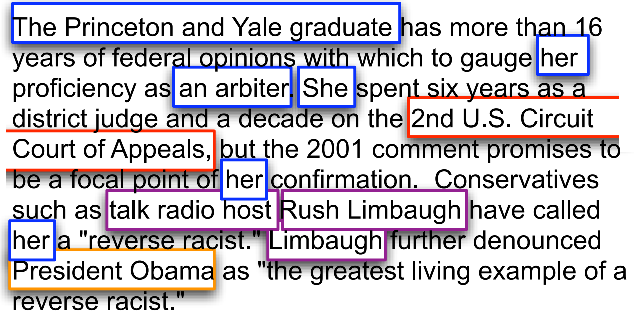
Grounding

Problem: which real-world entities are mentioned and where?

> The Princeton and Yale graduate has more than 16 years of federal opinions with which to gauge her proficiency as an arbiter. She spent six years as a district judge and a decade on the 2nd U.S. Circuit Court of Appeals, but the 2001 comment promises to be a focal point of her confirmation. Conservatives such as talk radio host Rush Limbaugh have called her a "reverse racist." Limbaugh further denounced President Obama as "the greatest living example of a reverse racist."

Grounding

Problem: which real-world entities are mentioned and where?



The Princeton and Yale graduate has more than 16 years of federal opinions with which to gauge her proficiency as an arbiter. She spent six years as a district judge and a decade on the 2nd U.S. Circuit Court of Appeals, but the 2001 comment promises to be a focal point of her confirmation. Conservatives such as talk radio host Rush Limbaugh have called her a "reverse racist." Limbaugh further denounced President Obama as "the greatest living example of a reverse racist."

Grounding

Problem: which real-world entities are mentioned and where in text?

- Entity detection (often seen as a kind of chunking)
- Coreference resolution: which referring expressions corefer?
- Grounding in ontologies

Another Dimension: Multiple Languages

Parallel and comparable corpora are a fascinating type of data.

- Sentence alignment (not widely studied now)
- Word alignment
- Phrase (chunk) alignment
- Tree alignment
- Bilingual parsing
- Automatic bilingual dictionary construction

Major application: **machine translation**

Debates in NLP

All of these tasks are not universally seen as important!

As noted, sentence segmentation and part-of-speech tagging are mostly “solved” (for English).

People who work on applications always question whether a particular level of analysis helps their application, and whether the useful ones deserve intrinsic evaluation (e.g., word alignment).

NLP Problems on the Frontier of Structured Prediction

- Finding the entire predicate-argument structure of a sentence
- Tasks requiring *generation* of text from deeper representations
    - Question answering
    - Translation
    - "Simplification"
- Learning about linguistic *types*—lexicons and ontologies
- Representing discourse structure and dialog structure
- Learning world knowledge from text

Summary of Some Structures

NLP problem	$\mathbf{x} \in \ldots$	$\mathbf{y} \in \ldots$
word segmentation	Γ^*	Σ^*
tokenization, tagging	Σ^*	$\Lambda^{\lvert\mathbf{x}\rvert}$
morphological parsing	Σ^*	Λ^*
chunking	Σ^*	$((\Lambda \times \{\mathsf{B}, \mathsf{I}\}) \cup \{\mathsf{O}\})^{\lvert\mathbf{x}\rvert}$
phrase-structure parsing	Σ^*	$2^{\Lambda \times \mathbb{N}^2}$
dependency parsing	Σ^*	$\{0, \ldots, \lvert\mathbf{x}\rvert\} \rightarrow 2^{\{1, \ldots, \lvert\mathbf{x}\rvert\}}$

Decoding

Notation

$\mathcal{X}$: set of possible inputs, often $\mathcal{X} = \Sigma^*$

$\mathbf{X}$: random variable taking values $\mathbf{x} \in \mathcal{X}$

$\mathcal{Y}$: set of possible *outputs*

$\mathbf{Y}$: random variable taking values $\mathbf{y} \in \mathcal{Y}$

h: prediction function $\mathcal{X} \rightarrow \mathcal{Y}$

A **decoder** is an implementation of h.

Why “Decoder”?

“Structured prediction” is the 2009 metaphor of choice, but before that, circa 1999, it was “source-channel models.”

Source-channel model:

$$p(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = \underbrace{p(\mathbf{Y} = \mathbf{y})}_{\text{source}} \times \underbrace{p(\mathbf{X} = \mathbf{x} \mid \mathbf{Y} = \mathbf{y})}_{\text{channel}} \tag{1}$$

Source-channel decoding:

$$\begin{aligned} h(\mathbf{x}) &= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y} \mid \mathbf{x}) \\ &= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \frac{p(\mathbf{y}) \times p(\mathbf{x} \mid \mathbf{y})}{p(\mathbf{x})} \\ &= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}) \times p(\mathbf{x} \mid \mathbf{y}) \end{aligned} \tag{2}$$

Decoding Defined

Generic definition:

$$h(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} \mathbb{E}_{\mathbf{Y} \sim p(\mathbf{Y} \mid \mathbf{x})} [\ell(\mathbf{y}, \mathbf{x}, \mathbf{Y})] \tag{3}$$

Common case when probabilistic models are used:

$$\begin{aligned} \ell(\mathbf{y}, \mathbf{x}, \mathbf{y}^*) &= 1 - \delta(\mathbf{y}, \mathbf{y}^*) \quad (4) \\ h(\mathbf{x}) &= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y} \mid \mathbf{x}) \quad (5) \end{aligned}$$

Linear Models

Let $\mathbf{g} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ denote a **feature vector function** that embeds input-output pairs in Euclidean space.

Linear models define a score parameterized by weights $\mathbf{w} \in \mathbb{R}^d$:

$$\mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) \tag{6}$$

Decoding with a linear model means finding

$$h(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) \tag{7}$$

Simplest Recipe For Structured Prediction

- Input and output spaces $\mathcal{X}$ and $\mathcal{Y}$
- A feature representation $\mathbf{g}$ (linear model)
- A decoder $h : \mathcal{X} \to \mathcal{Y}$
- A method for learning the parameters $\mathbf{w}$

On "Local" Features

Efficiency of decoding hinges crucially on $\mathbf{g}$.

Specifically, we often assume a specific structure to $\mathbf{g}$: that it can be calculated based on local parts of the structure.

This implies *independence assumptions* in the scoring function $\mathbf{w}^\top \mathbf{g}$.

Extreme version, when $\mathbf{y} = \langle y_1, \ldots, y_n \rangle$ is a sequence:

$$\mathbf{g}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, y_i, i) \tag{8}$$

(This is a non-structured classifier for each y_i!)

Dynamic Programming

Combinatorial optimization problems with optimal substructure and that break down into parts that are densely shared can often be solved efficiently by **dynamic programming**.

This relies on certain *factoring* properties of $\mathbf{g}$. Assume $\mathbf{y}$ breaks into "local parts" $\{\pi_i(\mathbf{y})\}_i$:

$$\mathbf{g}(\mathbf{x}, \mathbf{y}) = \sum_i \mathbf{f}(\mathbf{x}, \pi_i(\mathbf{y})) \tag{9}$$

Key example: Viterbi algorithm.

(Classical) Viterbi Algorithm

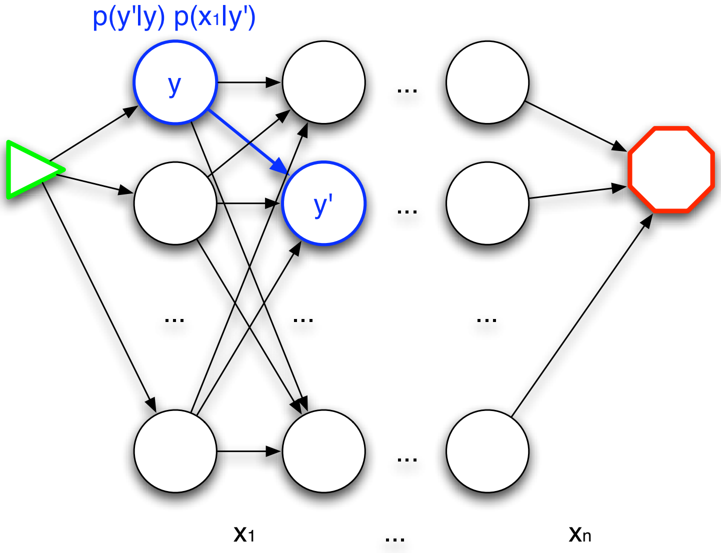
Input: $\mathbf{x} = \langle x_1, x_2, \ldots, x_n, \underbrace{\bigcirc}_{x_{n+1}} \rangle$, each $x_i \in \Sigma$

Output: $\mathbf{y} = \langle \underbrace{\triangleright}_{y_0}, y_1, y_2, \ldots, y_n, \underbrace{\bigcirc}_{y_{n+1}} \rangle$, each $y_i \in \Lambda$

Assumption (HMM): $p(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{n+1} p(y_i \mid y_{i-1}) p(x_i \mid y_i)$

$$\begin{aligned}
\max_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{x}, \mathbf{y}) &= V(\bigcirc, n+1) \qquad (10) \\
V(y, i) &= \max_{y' \in \Lambda} V(y', i-1) \times p(y \mid y') \times p(x_i \mid y) \\
V(\triangleright, 0) &= 1
\end{aligned}$$

Viterbi, Visualized



(Classical) Viterbi Algorithm

Input: $\mathbf{x} = \langle x_1, x_2, \ldots, x_n, \underbrace{\text{⯃}}_{x_{n+1}} \rangle$, each $x_i \in \Sigma$

Output: $\mathbf{y} = \langle \underbrace{\rhd}_{y_0}, y_1, y_2, \ldots, y_n, \underbrace{\text{⯃}}_{y_{n+1}} \rangle$, each $y_i \in \Lambda$

Assumption (HMM): $\underbrace{p(\mathbf{x}, \mathbf{y})}_{\exp \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y})} = \underbrace{\prod_{i=1}^{n+1} p(y_i \mid y_{i-1}) p(x_i \mid y_i)}_{\exp \mathbf{w}^\top \sum_{i=1}^{n+1} \mathbf{f}(\mathbf{x}, y_i, y_{i-1}, i)}$

$$\begin{aligned}
\max_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{x}, \mathbf{y}) &= V(\text{⯃}, n+1) & (11)\\
V(y, i) &= \max_{y' \in \Lambda} V(y', i-1) \times p(y \mid y') \times p(x_i \mid y) \\
V(\rhd, 0) &= 1
\end{aligned}$$

(Generalized) Viterbi Algorithm

Input: $\mathbf{x} = \langle x_1, x_2, \ldots, x_n, \underbrace{\bigcirc}_{x_{n+1}} \rangle$, each $x_i \in \Sigma$

Output: $\mathbf{y} = \langle \underbrace{\triangleright}_{y_0}, y_1, y_2, \ldots, y_n, \underbrace{\bigcirc}_{y_{n+1}} \rangle$, each $y_i \in \Lambda$

Assumption: $\mathbf{g}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n+1} \mathbf{f}(\mathbf{x}, y_i, y_{i-1}, i)$

$$\begin{aligned}
\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) &= \log V(\bigcirc, n+1) \qquad (12) \\
V(y, i) &= \max_{y' \in \Lambda} V(y', i-1) \times \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{x}, y, y', i)) \\
V(\triangleright, 0) &= 1
\end{aligned}$$

What Features Are "Local"?

Generalized Viterbi, as described, permits features that look at any part of the input (words, word shape, spelling features, etc.) and any two *adjacent* output symbols: $\mathbf{f}(\mathbf{x}, y_i, y_{i-1}, i)$.

Some things it still can't do:

- Three consecutive output symbols.
- Output symbols for two instances of the same word.
- How many times have I seen output symbol y?

Other DP Algorithms

- Monotonic sequence alignment/edit distance [Levenshtein, 1965]
- Probabilistic Earley's and CKY (weighted CFG parsing)
- Projective dependency parsing [Eisner, 1996]
- Parsing with other formalisms (combinatory categorial grammar, tree adjoining grammar, etc.)

Generic Dynamic Programming

This technique is so beloved by NLP that there are now:

- generalizations of logic programming for DP [Goodman, 1999]
- semiring-independent solvers [Eisner et al., 2005]
- connections to hypergraph search [Klein and Manning, 2001]
- generalizations of A^* for hypergraphs [Klein and Manning, 2003]
- algorithms for including non-factoring features [Chiang, 2007, Gimpel and Smith, 2009]

Bottom line: great way to think about decoders, often a good way to implement them.

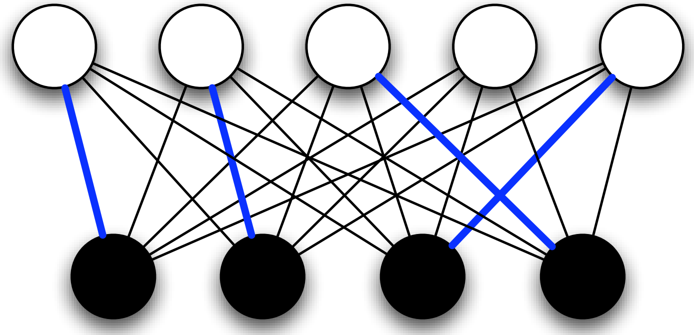
Structures are Graphs

We mention two interesting cases where decoding can be reduced to well-known algorithms for graphs.

Maximum Weighted Bipartite Matching

Input: two sentences $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$ and $\mathbf{x}' = \langle x'_1, x'_2, \ldots, x'_m \rangle$

Output: $\mathbf{y}$ is a matching of $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$; each x_i and each x'_j matches to at most one word in the other sequence



Maximum Weighted Bipartite Matching

Input: two sentences $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$ and $\mathbf{x}' = \langle x'_1, x'_2, \ldots, x'_m \rangle$

Output: $\mathbf{y}$ is a matching of $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$; each x_i and each x'_j matches to at most one word in the other sequence

Assumption: $\mathbf{g}(\mathbf{x}, \mathbf{x}', \mathbf{y}) = \sum_{\langle i,j \rangle \in \mathbf{y}} \mathbf{f}(\mathbf{x}, \mathbf{x}', i, j)$

Solution: Hungarian algorithm [Kuhn, 1955], with $O((n+m)^2 \log(n+m) + nm(n+m))$ runtime

Application: word alignment [Melamed, 2001]

Maximum Weighted (Directed) Spanning Tree

Input: sentence $\mathbf{x} = \langle \underbrace{\triangleright}_{x_0}, x_1, \ldots, x_n \rangle$

Output: $\mathbf{y} \in \{0, 1, \ldots, n\} \rightarrow 2^{\{1,\ldots,n\}}$ defines a 0-arborescence, i.e., a directed spanning tree with x_0 as the root and $x_1, \ldots, x_n$ as vertices; $\mathbf{y}(i)$ denotes y's parent

Assumption: $\mathbf{g}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, \mathbf{y}(i))$

Solution: Chu-Liu-Edmonds algorithm
[Chu and Liu, 1965, Edmonds, 1967] adapted by [Tarjan, 1977], with $O(n^2)$ runtime

Applicaton: nonprojective dependency parsing
[McDonald et al., 2005]

Dependency Parsing Features

The spanning tree approach works when we have only $\mathbf{f}(\mathbf{x}, i, \mathbf{y}(i))$.

- Parent-child features (words, word classes, lemmas)
- Context features (words on either side of the parent or child)
- Distance features

Non-local:

- Sibling features
- Grandparent/grandchild features
- Valency features (how many children?)
- Phrase features

Other Approaches

- Integer linear programming [Germann et al., 2001, Roth and Yih, 2004, Martins et al., 2009]
- Reranking: replace $\mathcal{Y}$ with the k best solutions from a simpler model [Collins, 2000, Charniak and Johnson, 2005]
- Stacking [Kou and Cohen, 2007]
- Belief propagation for structures [Smith and Eisner, 2008]
- Markov chain Monte Carlo methods [Finkel et al., 2006]
- Search [Daumé, 2006]

Note that many such approaches are tightly linked to specific kinds of *learning* algorithms.

Current Hot Topics

- Coarse-to-fine decoding [Charniak and Johnson, 2005]
- Decoding multiple structures at once ("joint" inference) [Cohen and Smith, 2007], among others
- Generic tools for building decoders using DP, ILP, ...

Break #1 (10 minutes)

Next up: Supervised structured natural language processing

Supervised Structured NLP

Recap

- We've seen a bunch of NLP problems presented as structured prediction problems.
- We've discussed the (generic) problem of *decoding* (i.e., making a prediction).
- Next: How to learn the prediction *model* from labeled data.

Learning Setting

Training data: $\tilde{\mathcal{D}} = \langle\langle\tilde{\mathbf{x}}^1, \tilde{\mathbf{y}}^1\rangle, \langle\tilde{\mathbf{x}}^2, \tilde{\mathbf{y}}^2\rangle \ldots, \langle\tilde{\mathbf{x}}^{\tilde{N}}, \tilde{\mathbf{y}}^{\tilde{N}}\rangle\rangle$

Testing data: $\dot{\mathcal{D}} = \langle\langle\dot{\mathbf{x}}^i, \dot{\mathbf{y}}^i\rangle\rangle_{i=1}^{\dot{N}}$

Remember that the annotations $\tilde{\mathbf{y}}^i$ depend heavily on *conventions* and are often subject to debate!

Loss Functions

$\ell : \mathcal{Y} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$

$\ell(\mathbf{y}, \mathbf{x}, \mathbf{y}^*)$ is the cost when $h(\mathbf{x}) = \mathbf{y}$ but the correct answer is $\mathbf{y}^*$.

Training usually looks like:

$$\min_{\mathbf{w}} \sum_{i=1}^{\tilde{N}} \ell(h(\tilde{\mathbf{x}}^i), \tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) + \text{modelcomplexity}(\mathbf{w}) \tag{13}$$

Considerable effort has gone into making the loss function used in training look like the evaluation function we care about on test data.

Loss Functions in NLP

NLP makes this harder with evaluation-time cost functions that:

- are not formally well-defined
- are not widely agreed upon
- are not unique and involve trade-offs
- are extrinsic (embedded in systems)
- change frequently
- require humans

Loss Functions in NLP

For *intrinsic* evaluation:

- tagging: count of words wrongly tagged
- chunking: F_1 of identified chunks, by chunk type
- parsing: count of incorrect phrases in a parse tree or words wrongly attached
- coreference: precision and recall

Admittedly, we'd rather see how these affect performance of real systems. But extrinsic evaluations are expensive.

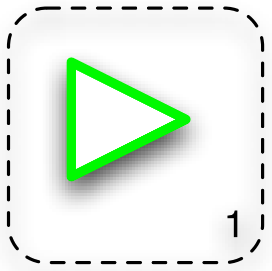
Generative Models

Basic idea: Define a stochastic process that can produce $\mathcal{X} \times \mathcal{Y}$ (or some appropriate subset), then estimate parameters using maximum likelihood (MLE) or maximum *a posteriori* (MAP).

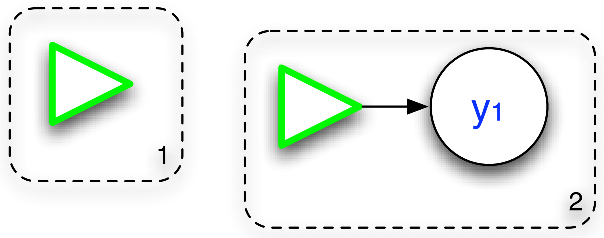
For NLP models, this commonly means a probabilistic grammar (e.g., HMM or PCFG).

Challenge: predicting each piece of structure exactly once limits the effective features $\mathbf{g}$ if we want to use simple estimators ("count and normalize"). There are also generative arbitrary-feature models [Rosenfeld, 1997, Smith et al., 2007].

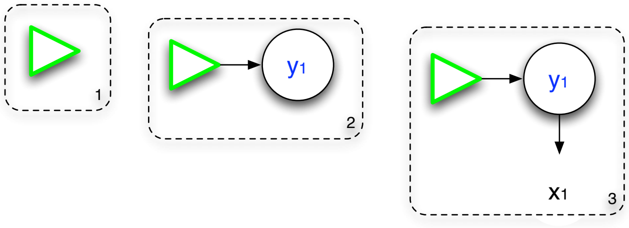
Example: HMM



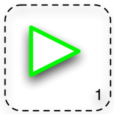
Example: HMM

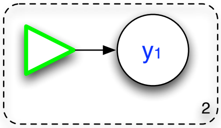


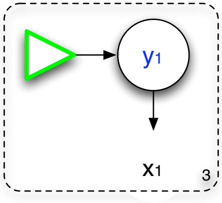
Example: HMM

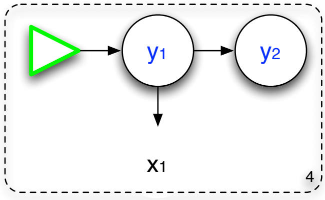


Example: HMM

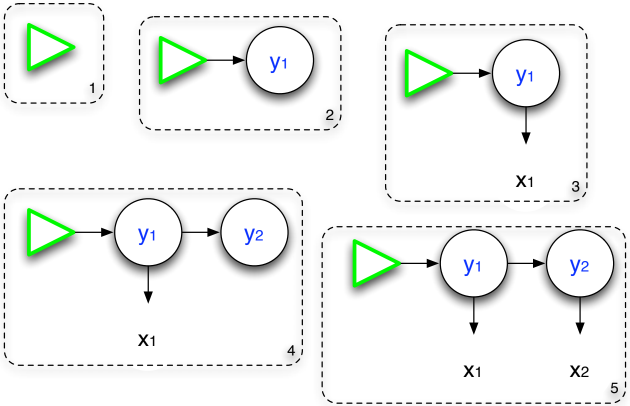




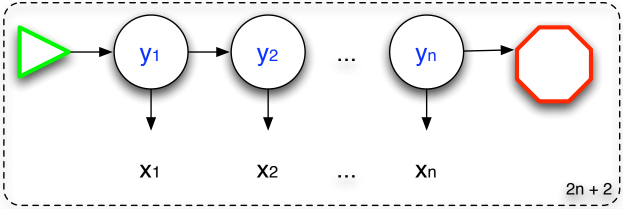




Example: HMM



Example: HMM



Generative Models of Structure

Pros: easy to train

Cons: major restrictions on features due to Markovian independence assumptions, log-loss on $\mathcal{X} \times \mathcal{Y}$

Bottom line: least satisfying from a machine learning perspective, but NLP makes heavy use of them anyway.

Why Be Generative?

If our goal is a particular type of prediction (here, from $\mathcal{X}$ to $\mathcal{Y}$), there's no reason to learn a distribution over $\mathcal{X}$.

(Of course, if we want our model to do *many* types of prediction, then a generative model may be ideal.)

Discriminative methods focus on the decoding function h.

Discrimative Structure Models (Take 1)

Early approaches [Ratnaparkhi et al., 1994, McCallum et al., 2000] broke the output $\mathbf{y} \in \mathcal{Y}$ into "parts" that could be built incrementally, e.g., using **probabilistic automata**.

Each "part" has its own discriminative classification model that depends on earlier decisions (trained using multinomials, multinomial logistic regression, SVMs, decision trees, etc.).

Classic case: Magerman's decision tree parser [Magerman, 1995].

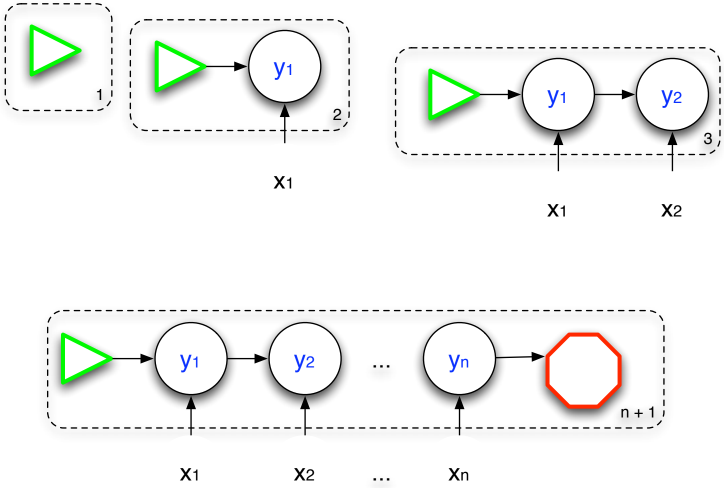
Decoding with Local Decisions

Generally state-space search or dynamic programming implements h.

Greedy version (e.g., [Nivre and Scholz, 2004]):

$$h(\mathbf{x}) = h_\ell(\mathbf{x}, h_{\ell-1}(\mathbf{x}, \cdots h_2(\mathbf{x}, h_1(\mathbf{x})) \cdots)) \tag{14}$$

Example: “Maximum Entropy Markov Model”



What's Wrong?

This formulation might lead to **label bias** [Lafferty et al., 2001]: there's no notion of "you made a bad choice earlier, and now there are no good options." (This is not a problem with search or inference; it is a problem with the model.)

Training doesn't match testing: training always assumes "earlier" h steps were correct.

Expressive power: some h cannot be represented when we make the model incremental [Smith and Johnson, 2007].

[Daumé, 2006] offers a way to train these that tries to take possible later decisions into account.

Local Classifiers

Pros: easy to train, approximate factored loss functions, rich "history-facing" features

Cons: lots of approximations (loss, decoding during training), label bias problem, expressive power

Bottom line: least satisfying analytically, but NLP makes heavy use of them anyway. A widely held view is that richer features often make up for the flaws.

Discriminative Structure Models (Take 2)

Want a global score (like generative models), discriminatively trained (like local classifiers), but with a single loss function on $\mathcal{Y}$.

Solution: **conditional random fields** [Lafferty et al., 2001]

Conditional Random Fields

Log-linear models over $\mathcal{Y}$ given evidence in $\mathcal{X}$:

$$\log p(\mathbf{y} \mid \mathbf{x}) \propto \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) \tag{15}$$

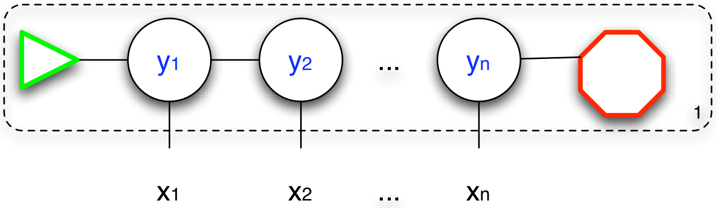
$$p(\mathbf{y} \mid \mathbf{x}) = \frac{e^{\mathbf{w}^\top \mathbf{g}(\mathbf{x},\mathbf{y})}}{z(\mathbf{w}, \mathbf{x})} \quad \begin{array}{l} \leftarrow \text{ exponentiated score} \\ \leftarrow \text{ partition function} \end{array} \tag{16}$$

$$= \exp\left(\mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) - \log z(\mathbf{w}, \mathbf{x})\right) \tag{17}$$

Assume $\mathbf{y}$ breaks into "local parts" $\{\pi_i(\mathbf{y})\}_i$:

$$p(\mathbf{y} \mid \mathbf{x}) = \exp\left(\mathbf{w}^\top \left(\sum_i \mathbf{f}(\mathbf{x}, \pi_i(\mathbf{y}))\right) - \log z(\mathbf{w}, \mathbf{x})\right) \tag{18}$$

CRFs for Sequence Labeling



Training CRFs

CRFs are (conditional) probabilistic models; M(C)LE requires:

$$\max_{\mathbf{w}} \prod_{i=1}^{\tilde{N}} p(\tilde{\mathbf{y}}^i \mid \tilde{\mathbf{x}}^i) \tag{19}$$

$$\equiv \quad \max_{\mathbf{w}} \sum_{i=1}^{\tilde{N}} \log p(\tilde{\mathbf{y}}^i \mid \tilde{\mathbf{x}}^i) \tag{20}$$

$$\equiv \quad \max_{\mathbf{w}} \mathbf{w}^\top \left(\sum_{i=1}^{\tilde{N}} \mathbf{g}(\tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) \right) - \sum_{i=1}^{\tilde{N}} \log z(\mathbf{w}, \tilde{\mathbf{x}}^i) \tag{21}$$

There is no closed-form solution. We must use iterative optimization routines.

CRFs: Implementation

$$\max_{\mathbf{w}} \mathbf{w}^\top \left(\sum_{i=1}^{\tilde{N}} \mathbf{g}(\tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) \right) - \sum_{i=1}^{\tilde{N}} \log z(\mathbf{w}, \tilde{\mathbf{x}}^i) \tag{22}$$

Fortunately, the above is **concave** and **differentiable** with respect to $\mathbf{w}$.

$$\frac{\partial}{\partial w_j} = \sum_{i=1}^{\tilde{N}} \left(g_j(\tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) - \mathbb{E}_{\mathbf{Y} \sim p(\mathbf{y} \mid \tilde{\mathbf{x}}^i)} g_j(\tilde{\mathbf{x}}^i, \mathbf{Y}) \right) \tag{23}$$

CRFs: Implementation

Calculating the objective and first derivatives requires a particular kind of inference that sums over $\mathbf{y}$ for a given $\mathbf{x}$:

$$\eta(\mathbf{w}, \mathbf{x}, r) = \sum_{\mathbf{y} \in \mathcal{Y}} \underbrace{r(\mathbf{x}, \mathbf{y})}_{\text{some function}} \exp \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) \qquad (24)$$

for some function r. The two cases we need are

- $\eta(\mathbf{w}, \mathbf{x}, 1) \equiv z(\mathbf{w}, \mathbf{x})$
- $\eta(\mathbf{w}, \mathbf{x}, g_j) \equiv \frac{\partial z}{\partial w_j}$

If the features factor sufficiently locally, the above are solvable with dynamic programming.

CRFs: Regularization

Like most discriminative learners, CRFs tend to overfit the training data.

Most common solution is to penalize models with large $|\mathbf{w}|$:

$$\max_{\mathbf{w}} \sum_{i=1}^{\tilde{N}} \log p(\tilde{\mathbf{y}}^i \mid \tilde{\mathbf{x}}^i) - C \sum_{j=1}^{d} w_j^2 \qquad (25)$$

This can be seen as "L_2 regularization," or as a MAP estimator with a Gaussian prior (zero means and $\sigma^2\mathbf{I}$ covariance matrix) on $\mathbf{w}$ [Chen and Rosenfeld, 2000].

There's also an L_1 version, but it's less widely used.

Generalizing CRFs to Other Structures

Though the name “conditional random field” is not exactly appropriate, similar models have been used for many problems apart from sequence labeling:

- context-free parsing [Finkel et al., 2008]
- dependency parsing [Smith and Smith, 2007]
- coreference [McCallum and Wellner, 2004]

It’s becoming more common to see approximate inference methods, like those used in graphical models, for dealing with $z(\mathbf{w}, \tilde{\mathbf{x}}^i)$ and $\frac{\partial z}{\partial w_j}$.

Useful Extension of CRF-Like Models

Sometimes some of the annotation is missing!

Perhaps:

- we believe the annotators could have used more fine-grained labels [Matsuzaki et al., 2005, Petrov et al., 2006].
- we believe there are structural patterns that help explain the phenomenon, but we are agnostic about the details [Wang et al., 2007, Das and Smith, 2009],
- or we think humans are incapable of finding or agreeing about those patterns [Blei and McAuliffe, 2008].

CRF-Like Models with Latent Variables

If the missing structure is not the essential desired output, i.e., $\mathbf{X}$ and $\mathbf{Y}$ are observed but some connection between them is missing, we can generalize CRFs (and related) to permit latent variables, here denoted $\mathbf{Z}$.

$$
\begin{aligned}
p(\mathbf{y} \mid \mathbf{x}) &= \sum_{\mathbf{z} \in \mathcal{Z}} p(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) && (26) \\
&= \frac{\sum_{\mathbf{z} \in \mathcal{Z}} \exp \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}, \mathbf{z})}{\sum_{\mathbf{y}' \in \mathcal{Y}, \mathbf{z} \in \mathcal{Z}} \exp \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}', \mathbf{z})} && (27)
\end{aligned}
$$

Example

Cats prefer to avoid the water.

?

Cats hate getting wet.

paraphrase

Cats prefer to avoid the water.

?

Cats prefer running water.

not paraphrase

CRFs

Pros: probabilistic interpretation (builds on graphical models), extends to latent variable models, flexible regularization

Cons: restricted to log-loss approximation to 0-1 loss on $\mathbf{y}$, requires $z(\mathbf{w}, \tilde{\mathbf{x}}^i)$

Bottom line: most promising for scenarios where we wish to preserve uncertainty/ambiguity, but computationally expensive.

Discriminative Structure Models (Take 3)

Structured perceptron [Collins, 2002]:

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{g}(\tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) - \mathbf{g}(\tilde{\mathbf{x}}^i, h(\tilde{\mathbf{x}}^i)) \tag{28}$$

Usually weight vectors over all iterations are averaged or (more expensively) a vote is taken, to deal with the problem of oscillation.

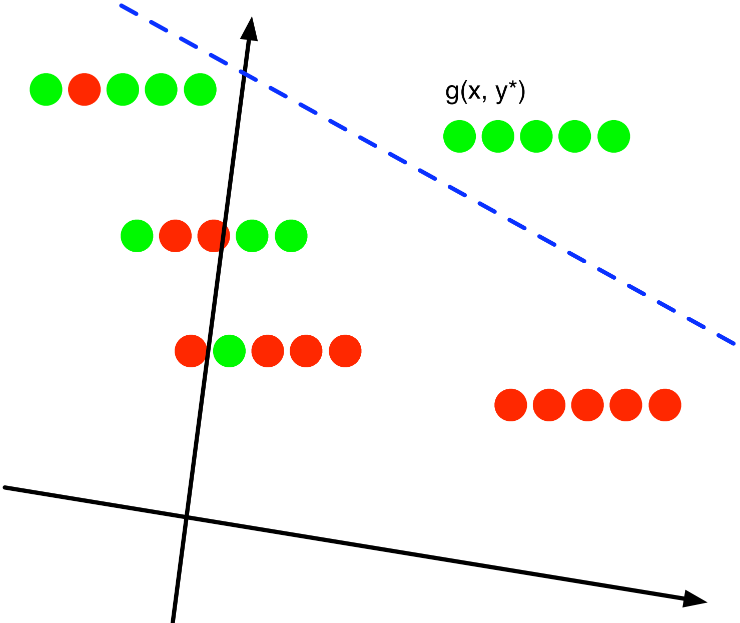
Structured Perceptron

It's very easy to implement. What is it doing?

Let $\not{\mathcal{Y}}_i$ denote $\{\mathbf{y} \in \mathcal{Y} \mid \mathbf{y} \neq \tilde{\mathbf{y}}^i\}$.

- It's searching for a hyperplane in $\mathbb{R}^d$ that puts $\{\mathbf{g}(\tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i)\}_{i=1}^{\tilde{N}}$ on one side and $\{\mathbf{g}(\tilde{\mathbf{x}}^i, \mathbf{y}) \mid \mathbf{y} \in \not{\mathcal{Y}}_i\}_{i=1}^{\tilde{N}}$ on the other.
- It will find such a hyperplane, eventually, if one exists.
- This implies the use of 0-1 loss: every output in $\not{\mathcal{Y}}_i$ is equally bad.

Structured Perceptron



Structured Perceptron

Pros: easy to implement, only need a decoder

Cons: brittle loss function (0-1), no built-in mechanism to avoid overfitting

Bottom line: try this first, once you have a decoder, to see if your features make any sense at all.

Discriminative Structure Models (Take 4)

Recall from classification: the perceptron finds *some* hyperplane, while SVMs look for one with a *wide margin*.

Rationale: wide margins are expected to generalize better.

Further, we'd like to take into account alternative loss functions, not just 0-1.

(Note: SVMs often make people think about *kernels*. Kernels in structured prediction are a hot topic, but out of scope.)

Large Margin Structured Classifiers

General setting [Taskar et al., 2003, Tsochantaridis et al., 2005]:

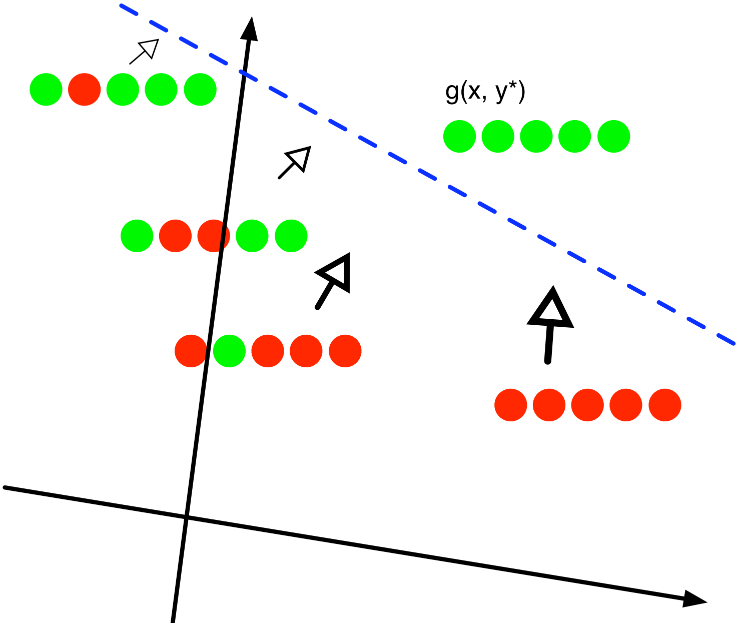
$$\min_{\mathbf{w}\in\mathbb{R}^d,\boldsymbol{\xi}\geq\mathbf{0}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\tilde{N}} \xi_i \tag{29}$$

$\{\xi_i\}_i$ are *slack variables* that express the extent to which the model doesn't do the "right thing" on instance i. The constraints can be $(\forall \mathbf{y}, i)$:

$$\mathbf{w}^\top \left(\mathbf{g}(\tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) - \mathbf{g}(\tilde{\mathbf{x}}^i, \mathbf{y})\right) \geq \tag{30}$$
$$\begin{cases} \ell(\mathbf{y}, \tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) - \xi_{i,\mathbf{y}} & \text{margin rescaling} \\ 1 - \frac{\xi_{i,\mathbf{y}}}{\ell(\mathbf{y}, \tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i)} & \text{slack rescaling version} \\ 0 & \text{reminiscent of perceptron} \end{cases}$$

Exponentially many constraints! M^{3}Ns are a variant with margin rescaling that exploit factored $\mathbf{g}$ and ℓ.

Hyperplane View, Again



Objective, Again

Collapsing the $\boldsymbol{\xi}$ (margin rescaling version):

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\tilde{N}} \quad \Big(-\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) \\ + \max_{\mathbf{y}\in\mathcal{Y}} \Big(\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}^i, \mathbf{y} + \ell(\mathbf{y}, \tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i)\Big)\Big) \tag{31}$$

Loss-Augmented Decoding

Recall standard decoding:

$$h(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}) \tag{32}$$

Loss-augmented decoding (additive, used with margin rescaling):

$$h'(\mathbf{x}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y}' \in \mathcal{Y}} \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}') + \ell(\mathbf{y}', \mathbf{x}, \mathbf{y}) \tag{33}$$

Manageable if $\mathbf{g}$ and ℓ factor the same way; exploited by M^{3}Ns.

Multiplicative version (used in slack rescaling):

$$h'(\mathbf{x}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y}' \in \mathcal{Y}} \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y}') \times \ell(\mathbf{y}', \mathbf{x}, \mathbf{y}) \tag{34}$$

Usually intractable.

Implementation

Specialized algorithms have been developed for both the M^{3}N and SVM$^{\text{struct}}$ variations:

- Subgradient methods [Ratliff et al., 2006] and cutting planes [Tsochantaridis et al., 2005], both of which rely on **loss-augmented decoding**.
- For M^{3}Ns, dual extragradient [Taskar et al., 2005], exponentiated gradient [Bartlett et al., 2004].
- "Passive-aggressive" online algorithms [Crammer et al., 2006]; connection to objectives not entirely clear, but one version is very close to subgradient with $C \rightarrow +\infty$.

“Passive Aggressive” Online Learners

Rather than starting from the objective function, these start with the perceptron update idea and try to augment it with **loss**, **margin**, and/or **regularization**. See [Crammer et al., 2006].

Large Margin Structured Classifiers

Pros: loss function flexibility, lots of recent advances in making training feasible, flexible regularization (in principle)

Cons: can require loss-augmented decoding, no probabilistic interpretation

Bottom line: best approach in theory, and becoming more and more usable.

CRFs Approximate Structured SVMs.

Assume 0-1 loss. We can substitute the $\boldsymbol{\xi}$ in the structured SVM objective to get:

$$\begin{aligned} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\tilde{N}} -\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) + \max_{\mathbf{y}\in\mathcal{Y}} \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}^i, \mathbf{y}) \qquad (35) \\ \approx \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\tilde{N}} -\mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}^i, \tilde{\mathbf{y}}^i) + \underbrace{\log \sum_{\mathbf{y}\in\mathcal{Y}}}_{\text{softmax}} \exp \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}^i, \mathbf{y}) \end{aligned}$$

So with the softmax approximation, we get the CRF objective, with quadratic regularization.

What's Special about Structured *NLP*?

- Good features are worth a lot; they often come from linguistics.
- Inference is expensive, so scalability and fast convergence are important.
- Robustness to imperfect data is important.
- Robustness to inexact inference (including decoding) is also important.
- All things equal, CRFs and large-margin classifiers are both state-of-the-art, though generative models often sneak in as "features."

Connection to (Weighted) Grammars

For several decades, much of NLP was about engineering *grammars* for analyzing these structures.

The *ambiguity* problem led us to statistics.

Grammars can be used to impose hard constraints (i.e., define $\mathcal{Y}$) or soft ones (weighted features) within a structured prediction model.

Examples of other grammar classes made into statistical models: [Schabes, 1992, Abney, 1997, Clark and Curran, 2007].

Current Challenges

- Adapting across domains ($\tilde{\mathcal{D}}$ and $\dot{\mathcal{D}}$ come from different distributions)
- Coping with data sparseness: smoothing, regularization, feature selection
- Noise or uncertainty in annotations
- Dependence on annotation conventions
- Handling features and loss functions that do not factor (approximate structured inference)
- Training expense makes tuning of regularization constants, feature selection, and other “meta-training” onerous.

Break #2 (10 minutes)

Next up: *Unsupervised* structured natural language processing

Unsupervised Structured NLP

Recap

- We've seen a bunch of NLP problems presented as structured prediction problems.
- We've discussed the (generic) problem of *decoding* (i.e., making a prediction).
- We saw a bunch of learning methods for learning these predictors, supervised.
- Next: How to learn the prediction *model* from *un*labeled data.

Learning Setting

Training data: $\tilde{\mathcal{D}} = \langle \tilde{\mathbf{x}}^1, \tilde{\mathbf{x}}^2, \ldots, \tilde{\mathbf{x}}^{\tilde{N}} \rangle$

Testing data: $\dot{\mathcal{D}} = \langle \langle \dot{\mathbf{x}}^i, \dot{\mathbf{y}}^i \rangle \rangle_{i=1}^{\dot{N}}$

Why “Un,” Not “Semi”?

Semisupervised learning: some annotated training data and a large set of unannotated training data.

There are many approaches to this. In practice, they mostly boil down to one of:

- Combining a supervised objective and an unsupervised objective
- Using unlabeled data to create features or regularization terms for supervised learners
- Bootstrapping algorithms that gradually modify the training dataset for a supervised learner
- Unsupervised learning with constraints obtained from the annotated data

So we consider “pure” unsupervised learning, both for use on its own and within semisupervised learning.

How Unsupervised is Unsupervised?

For any talk on unsupervised NLP, there will be at least one member of the audience left unconvinced about whether the approach was “really” unsupervised.

We will use the term broadly—if unconventionally—to mean that the training data are incomplete. Sometimes the “missing parts” are not intrinsically interesting (often called “latent-variable” models), but sometimes they are the desired output (what most people call “unsupervised”).

Notation

$\mathcal{X}$: set of possible inputs, often $\mathcal{X} = \Sigma^*$

$\mathbf{X}$: random variable taking values $\mathbf{x} \in \mathcal{X}$

$\mathcal{Y}$: set of possible *outputs*, **unseen even during training**

$\mathbf{Y}$: random variable taking values $\mathbf{y} \in \mathcal{Y}$

Expectation-Maximization

Most people's first exposure comes from mixtures of Gaussians for clustering data in $\mathbb{R}^d$.

Assume a generative model $p \in \mathcal{P}$ over $\mathcal{X} \times \mathcal{Y}$.

MLE:

$$\max_{p \in \mathcal{P}} \prod_{i=1}^{\tilde{N}} p(\tilde{\mathbf{x}}^i) \equiv \max_{p \in \mathcal{P}} \sum_{i=1}^{\tilde{N}} \log p(\tilde{\mathbf{x}}^i) \equiv \max_{p \in \mathcal{P}} \sum_{i=1}^{\tilde{N}} \log \sum_{\mathbf{y} \in \mathcal{Y}} p(\tilde{\mathbf{x}}^i, \mathbf{y}) \tag{36}$$

EM

EM can be seen as a coordinate ascent algorithm that alternates between choosing $p \in \mathcal{P}$ and auxiliary distibutions $q_i \in \mathcal{Q}$.

$$\max_{p \in \mathcal{P}, \{q_i\}_{i=1}^{\tilde{N}} \in \mathcal{Q}} \sum_{i=1}^{\tilde{N}} \log p(\tilde{\mathbf{x}}^i) - \boldsymbol{D}_{KL}(q_i \| p(\cdot \mid \tilde{\mathbf{x}}^i)) \tag{37}$$

The **E step** optimizes with respect to each q_i, holding p fixed, by solving

$$q_i(\mathbf{y}) \leftarrow p(\mathbf{y} \mid \tilde{\mathbf{x}}^i) \tag{38}$$

The **M step** optimizes p, holding the q_i fixed; the result looks just like supervised MLE for family $\mathcal{P}$ with fractional counts for $\mathbf{y}$ values:

$$\max_{p \in \mathcal{P}} \sum_{i=1}^{\tilde{N}} \sum_{\mathbf{y} \in \mathcal{Y}} q_i(\mathbf{y}) \log p(\tilde{\mathbf{x}}^i, \mathbf{y}) \tag{39}$$

Structured EM (E step)

In structured prediction, $\mathcal{Y}$ is huge, so q_i needs to be represented compactly, usually using sufficient statistics. In NLP, the E step almost always uses dynamic programming.

The sufficient statistics usually look a lot like $\frac{\partial z}{\partial w_j}$, and the algorithms are almost identical to those for CRF models.

Structured EM (M step)

Making the M step efficient generally requires us to restrict $\mathcal{P}$ to be a relatively simple family, e.g., stochastic grammars.

EM for HMMs—"Baum-Welch training"—is one of the earliest examples of EM.

One Interpretation of EM

EM is just trying to accomplish:

$$\max_{p \in \mathcal{P}} \sum_{i=1}^{\tilde{N}} \log \sum_{\mathbf{y} \in \mathcal{Y}} p(\tilde{\mathbf{x}}^i, \mathbf{y}) \tag{40}$$

The whole algorithm can be derived from the above; typically:

- The E step exploits independence assumptions in $p(\tilde{\mathbf{x}}^i, y)$.
- The M step manages the constraint that $p \in \mathcal{P}$.

Alternative optimization methods are possible!

Examples of EM for NLP

- Context-free grammars [Carroll and Charniak, 1992]
- Word alignment [Brown et al., 1993]
- Part-of-speech tagging [Merialdo, 1994]
- Coreference resolution [Charniak, 2001]
- Bracketing structure [Klein and Manning, 2002]
- Dependency structure [Klein and Manning, 2004]

EM for NLP

- Generative models that work in supervised mode often fail unsupervised [Carroll and Charniak, 1992].
- Success is most probable when the problem is highly constrained, the initialization is carefully designed, and/or the model comes with strong bias.
- Rich features pose a problem, because MLE for generative log-linear models over $\mathcal{X} \times \mathcal{Y}$ have complicated, possibly divergent z.

Discriminative, Unsupervised Learning?

This is sort of a contradiction.

Unsupervised learning assumes that $\langle \tilde{\mathbf{y}}^i \rangle_i^{\tilde{N}}$ are *unknown*.

Discriminative learning aims to minimize loss on training data, training toward $\langle \tilde{\mathbf{y}}^i \rangle_i^{\tilde{N}}$, the missing information. So loss of any hypothesis $\mathbf{y}$ is also unknown.

One way of mixing discriminative methods with unsupervised learning: **contrastive estimation**, which maximizes $p(\mathbf{X} = \mathbf{x} \mid \pi(\mathbf{X}) = \pi(\mathbf{x}))$ [Smith and Eisner, 2005]

Contrastive Estimation

MLE for generative log-linear models with hidden variable $\mathbf{Y}$:

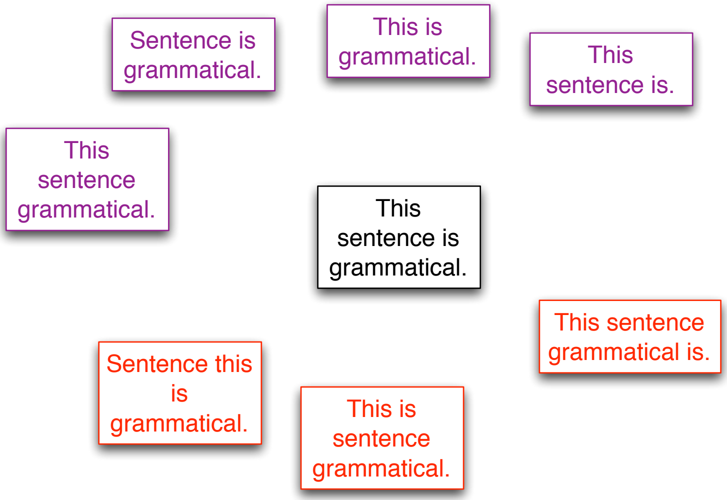
$$\max_{\mathbf{w}} \sum_{i=1}^{\tilde{N}} \log p(\tilde{\mathbf{x}}^i) \equiv \max_{\mathbf{w}} \sum_{i=1}^{\tilde{N}} \log \frac{\sum_{\mathbf{y} \in \mathcal{Y}} \exp \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}^i, \mathbf{y})}{\sum_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{y} \in \mathcal{Y}} \exp \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \mathbf{y})} \tag{41}$$

Numerator is like z for CRFs (expensive). Denominator is even worse!

Idea: make this a conditional model by conditioning against a variable $\mathbf{N}$ that constrains $\mathbf{X}$:

$$\max_{\mathbf{w}} \sum_{i=1}^{\tilde{N}} \log p(\tilde{\mathbf{x}}^i \mid \tilde{\mathbf{n}}^i) \equiv \max_{\mathbf{w}} \sum_{i=1}^{\tilde{N}} \log \frac{\sum_{\mathbf{y} \in \mathcal{Y}} \exp \mathbf{w}^\top \mathbf{g}(\tilde{\mathbf{x}}^i, \tilde{\mathbf{n}}^i, \mathbf{y})}{\sum_{\mathbf{x} \in \tilde{\mathbf{n}}^i} \sum_{\mathbf{y} \in \mathcal{Y}} \exp \mathbf{w}^\top \mathbf{g}(\mathbf{x}, \tilde{\mathbf{n}}^i, \mathbf{y})} \tag{42}$$

Contrastive Neighborhoods for Sentences


Sentence is grammatical.
This is grammatical.
This sentence is.
This sentence grammatical.
This sentence is grammatical.
This sentence grammatical is.
Sentence this is grammatical.
This is sentence grammatical.

The Linguistic Intuition Behind Contrastive Estimation

$\tilde{\mathbf{x}}^i$ is a positive example (e.g., a sentence).

Any grammatical sentence suggests implicitly that a certain set of sentences is *ungrammatical* (or less grammatical), namely, those that are perturbations of it. We call these the **neighborhood**; for $\tilde{\mathbf{x}}^i$, they are $\tilde{\mathbf{n}}^i \subseteq \mathcal{X}$.

On average, $\tilde{\mathbf{n}}^i$ should be less good than $\tilde{\mathbf{x}}^i$. Instead of simply making $\tilde{\mathbf{x}}^i$ more likely, we make it likely at the *expense* of $\tilde{\mathbf{n}}^i$.

Depending on how we represent $\tilde{\mathbf{n}}^i$, we can make learning efficient (dynamic programming).

Bayesian Methods in NLP

Bayesian Methods in NLP

The general idea is to manage more of our uncertainty; not just about $\mathbf{y}$, but also about $\mathbf{w}$. There are two flavors for models:

- **Parametric** approaches: dimensionality of $\mathbf{w}$, features, and $\mathcal{Y}$ are all assumed known.
- **Nonparametric** approaches: richness of the model depends on the data.

There are two methodologies:

- **Full** Bayesian: write down prior, do inference on the data.
- **Empirical** Bayesian: estimate parameters of prior from data.

Bayesian Unsupervised Structured Prediction

(I use the linear model notation.)

$$p(\mathbf{x} \mid \mathbf{v}) = \int \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{x}, \mathbf{y} \mid \mathbf{w}) p(\mathbf{w} \mid \mathbf{v}) d\mathbf{w} \tag{43}$$

- **Full** Bayesian: write down $\mathbf{v}$, figure out distribution over $\mathbf{y}$ (and maybe $\mathbf{w}$).
- **Empirical** Bayesian: estimate $\mathbf{v}$ from data.

Bayesian Unsupervised Structured Prediction

$$p(\mathbf{x} \mid \mathbf{v}) = \int \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{x}, \mathbf{y} \mid \mathbf{w}) p(\mathbf{w} \mid \mathbf{v}) d\mathbf{w} \tag{44}$$

Common choices for $\mathbf{v}$:

- Dirichlet distribution(s) [Blei et al., 2003, Goldwater and Griffiths, 2007, Johnson et al., 2007]
- (Hierarchical) Pitman-Yor process [Teh, 2006, Goldwater et al., 2006]
- Logistic normal distribution(s) [Blei and Lafferty, 2006, Cohen et al., 2008]

Example: Latent Dirichlet Allocation Model [Blei et al., 2003]

$\mathcal{X}$: $(\Sigma^*)^*$ (document collection)

$\mathcal{Y}$: $(\{1, \ldots, K\}^*)^*$ ("topic" for each word token) and $(\triangle^K)^*$ (topic distribution per document)

$\mathbf{w}$: $\boldsymbol{\beta} = \{\beta_{x|y}\}_{x \in \Sigma, y \in \{1,\ldots,K\}}$ are word distributions given topics, $\boldsymbol{\theta}$ are document-specific topic distributions

$\mathbf{v}$: Dirichlet distributions over $\boldsymbol{\theta}$

$$p(\mathbf{x}, \mathbf{y}, \mathbf{w} \mid \mathbf{v}) = \text{Dirichlet}(\boldsymbol{\theta} \mid \mathbf{v}) \prod_{i=1}^{n} \theta_{y_i} \beta_{x_i|y_i} \tag{45}$$

Empirical Bayes: learn $\mathbf{v}$ and $\boldsymbol{\beta}$

Examples of Bayesian NLP

- Topic models [Blei et al., 2003]
- Word segmentation [Goldwater et al., 2006]
- Part-of-speech tagging [Goldwater and Griffiths, 2007]
- Syntactic category refinement
  [Liang et al., 2007, Finkel et al., 2007]
- Dependency parsing [Cohen et al., 2008]
- Coreference resolution [Haghighi and Klein, 2007]

Approximate Inference

In principle, the challenge is the same as approximate inference for supervised CRFs with non-local features.

Here, there are usually integrals thrown into the mix, to deal with $\mathbf{w}$ (which is continuous).

Variational inference is widely used and often exploits dynamic programming with structured $\mathbf{y}$ [Cohen et al., 2008].

Uncertainty About Bayesian NLP

While mathematically elegant, Bayesian methods haven't yet revolutionized NLP.

Pros: a "language" for talking about hidden structure and modeling it, as well as breaking independence assumptions; generic techniques for (approximate) inference and learning, priors could be a way to encode knowledge

Cons: many approximations for efficiency, not always clear how to encode *linguistic* knowledge in priors

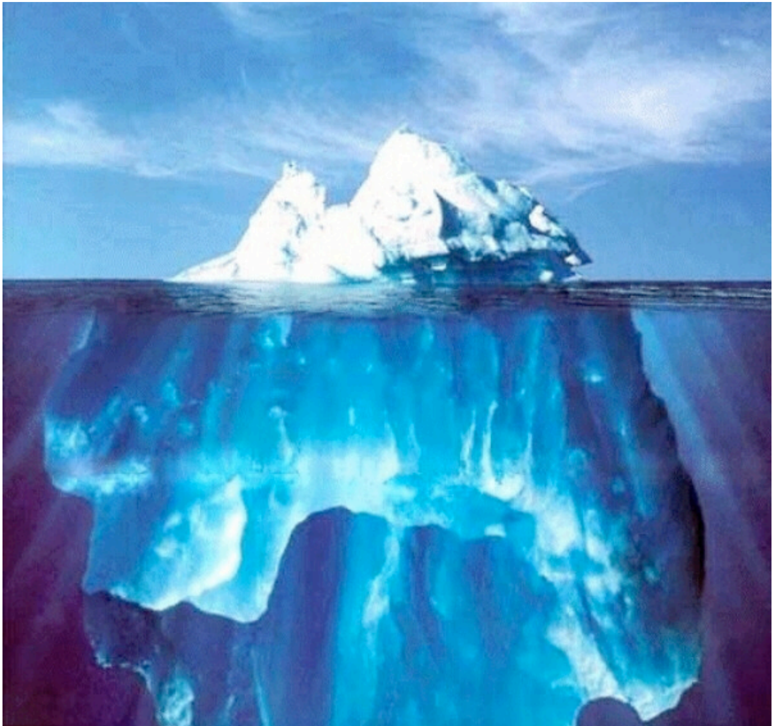
A New Twist?

We've discussed how lots of NLP problems can be transformed into the language of machine learning, using structured prediction.

If you consider *hierarchical* generative models (Bayesian or not), they begin to look like *grammars*.

Do probabilistic grammars offer a platform for describing graphical models? See, e.g., [Johnson et al., 2006].

Credit for suggesting this idea goes to Mark Johnson.



Important Points About Unsupervised Learning

- In unsupervised learning, the objective function is usually not convex (but see [Xu et al., 2006]) ; we mainly hope for local optima. Initialization can therefore be very important.
- Local optima seem to explode combinatorially because of symmetries among models.
- Bias (priors, constraints, contrast neighborhoods, etc.): very important for avoiding degenerate solutions.
- Unsupervised NLP is basically like unsupervised everything else, only harder.

Tutorial Summary

- Discussed linguistic analysis problems that are examples of **structured prediction**.
- Presented algorithmic tools used for *making* structured predictions (**decoding**).
- Presented the dominant techniques used in NLP for *learning* to make structured predictions from
    - complete data (**supervisedly**) and
    - incomplete data (**unsupervisedly**).

Themes

NLP is a great proving ground for ML ideas.

NLP “tasks” only partially fit ML expectations: loss, assumptions, and even the output space are always up for debate.

Separating models from learning algorithms and both from inference methods and loss functions has some advantages, and NLP is headed in that direction.

Further abstraction: availability of annotated/unannotated training data, and how much you trust it.

Acknowledgments

- Students in “Language and Stats II” at CMU in 2006–8
- *Ph.D. students*: Shay Cohen, Dipanjan Das, Kevin Gimpel, Mike Heilman, André Martins, Nathan Schneider, Tae Yano
- *Colleagues*: William Cohen, Mike Collins, Hal Daumé, Jason Eisner, Sharon Goldwater, Mark Johnson, Dan Klein, John Lafferty, Chris Manning, Fernando Pereira, David Smith

References I

Abney, S. P. (1997).
Stochastic attribute-value grammars.
Computational Linguistics, 23(4):597–617.

Bartlett, P. L., Collins, M., Taskar, B., and McAllester, D. (2004).
Exponentiated gradient algorithms for large-margin structured classification.
In *NIPS*.

Blei, D. and McAuliffe, J. (2008).
Supervised topic models.
In *Advances in NIPS 20*.

Blei, D., Ng, A., and Jordan, M. (2003).
Latent Dirichlet allocation.
Journal of Machine Learning Research, 3:993–1022.

Blei, D. M. and Lafferty, J. D. (2006).
Correlated topic models.
In *Advances in NIPS*.

Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993).
The mathematics of statistical machine translation: Parameter estimation.
Computational Linguistics, 19(2):263–311.

Carroll, G. and Charniak, E. (1992).
Two experiments on learning probabilistic dependency grammars from corpora.
Technical report, Brown University.

References II

Charniak, E. (2001).
Unsupervised learning of name structure from coreference data.
In *Proc. of NAACL*.

Charniak, E. and Johnson, M. (2005).
Coarse-to-fine n-best parsing and maxent discriminative reranking.
In *Proc. of ACL*.

Chen, S. and Rosenfeld, R. (2000).
A survey of smoothing techniques for ME models.
IEEE Transactions on Speech and Audio Processing, 8(1):37–50.

Chiang, D. (2007).
Hierarchical phrase-based translation.
Computational Linguistics, 33(2):201–228.

Chu, Y. J. and Liu, T. H. (1965).
On the shortest arborescence of a directed graph.
Science Sinica, 14:1396–1400.

Clark, S. and Curran, J. R. (2007).
Wide-coverage efficient statistical parsing with CCG and log-linear models.
Computational Linguistics, 33(4).

Cohen, S. B., Gimpel, K., and Smith, N. A. (2008).
Logistic normal priors for unsupervised probabilistic grammar induction.
In *NIPS 21*.

Cohen, S. B. and Smith, N. A. (2007).
Joint morphological and syntactic disambiguation.
In *Proc. of EMNLP-CoNLL*.

References III

Collins, M. (2000).
Discriminative reranking for natural language parsing.
In *Proc. of ICML*.

Collins, M. (2002).
Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms.
In *Proc. of EMNLP*.

Crammer, K., Dekel, O., Keshet, J., Shalev-Schwartz, S., and Singer, Y. (2006).
Online passive aggressive algorithms.
Journal of Machine Learning Research, 7.

Das, D. and Smith, N. A. (2009).
Paraphrase identification as probabilistic quasi-synchronous recognition.
In *Proc. of ACL-IJCNLP*.

Daumé, H. (2006).
Practical Structured Learning Techniques for Natural Language Processing.
PhD thesis, University of Southern California, Los Angeles, CA.

Edmonds, J. (1967).
Optimum branchings.
Journal of Research of the National Bureau of Standards, 71B:233–240.

Eisner, J. (1996).
Three new probabilistic models for dependency parsing: An exploration.
In *Proc. of COLING*.

References IV

Eisner, J., Goldlust, E., and Smith, N. A. (2005).
Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language.
In *Proc. of HLT-EMNLP*.

Finkel, J. R., Grenager, T., and Manning, C. D. (2007).
The infinite tree.
In *Proc. of ACL*.

Finkel, J. R., Kleeman, A., and Manning, C. D. (2008).
Efficient, feature-based, conditional random field parsing.
In *Proc. of ACL-HLT*.

Finkel, J. R., Manning, C. D., and Ng, A. Y. (2006).
Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines.
In *Proc. of EMNLP*.

Gaifman, H. (1965).
Dependency systems and phrase-structure systems.
Information and Control, 8.

Germann, U., Jahr, M., Knight, K., Marcu, D., and Yamada, K. (2001).
Fast decoding and optimal decoding for machine translation.
In *Proc. of ACL*.

Gimpel, K. and Smith, N. A. (2009).
Cube summing, approximate inference with non-local features, and dynamic programming without semirings.
In *Proc. of EACL*.

Goldwater, S. and Griffiths, T. L. (2007).
A fully Bayesian approach to unsupervised part-of-speech tagging.
In *Proc. of ACL*.

References V

Goldwater, S., Griffiths, T. L., and Johnson, M. (2006).
Contextual dependencies in unsupervised word segmentation.
In *Proc. of COLING-ACL*.

Goodman, J. (1999).
Semiring parsing.
Computational Linguistics, 25(4):573–605.

Haghighi, A. and Klein, D. (2007).
Unsupervised coreference resolution in a nonparametric Bayesian model.
In *Proc. of ACL*.

Johnson, M., Griffiths, T., and Goldwater, S. (2007).
Bayesian inference for PCFGs via Markov chain Monte Carlo.
In *Proc. of NAACL*.

Johnson, M., Griffiths, T. L., and Goldwater, S. (2006).
Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models.
In *Proc. of NIPS*.

Klein, D. and Manning, C. (2001).
Parsing and hypergraphs.
In *Proc. of IWPT*.

Klein, D. and Manning, C. D. (2002).
A generative constituent-context model for improved grammar induction.
In *Proc. of ACL*.

Klein, D. and Manning, C. D. (2003).
A* parsing: Fast exact Viterbi parse selection.
In *Proc. of HLT-NAACL*, pages 119–126.

References VI

Klein, D. and Manning, C. D. (2004).
Corpus-based induction of syntactic structure: Models of dependency and constituency.
In *Proc. of ACL*.

Kou, Z. and Cohen, W. W. (2007).
Stacked graphical models for efficient inference in Markov random fields.
In *Proceedings of SDM*.

Kuhn, H. W. (1955).
The Hungarian method for the assignment problem.
Naval Research Logistics Quarterly, 2:83–87.

Lafferty, J., McCallum, A., and Pereira, F. (2001).
Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
In *Proc. of ICML*.

Levenshtein, V. (1965).
Binary codes capable of correcting spurious insertions and deletions of ones.
Problems of Information Transmission, 1:8–17.

Liang, P., Petrov, S., Jordan, M., and Klein, D. (2007).
The infinite PCFG using hierarchical Dirichlet processes.
In *Proc. of EMNLP-CoNLL*.

Magerman, D. M. (1995).
Statistical decision-tree models for parsing.
In *Proc. of ACL*.

Martins, A. F. T., Smith, N. A., and Xing, E. P. (2009).
Concise integer linear programming formulations for dependency parsing.
In *Proc. of ACL-IJCNLP*.

References VII

Matsuzaki, T., Miyao, Y., and Tsujii, J. (2005).
Probabilistic CFG with latent annotations.
In *Proc. of ACL*.

McCallum, A., Freitag, D., and Pereira, F. (2000).
Maximum entropy Markov models for information extraction and segmentation.
In *Proc. of ICML*.

McCallum, A. and Wellner, B. (2004).
Conditional models of identity uncertainty with application to noun coreference.
In *NIPS*.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005).
Non-projective dependency parsing using spanning tree algorithms.
In *Proc. of HLT-EMNLP*.

Melamed, I. D. (2001).
Empirical Methods for Exploiting Parallel Texts.
MIT Press.

Merialdo, B. (1994).
Tagging English text with a probabilistic model.
Computational Linguistics, 20(2):155–72.

Nivre, J. and Scholz, M. (2004).
Deterministic dependency parsing of English text.
In *Proc. of COLING*.

Palmer, M., Gildea, D., and Kingsbury, P. (2005).
The Proposition bank: A corpus annotated with semantic roles.
Computational Linguistics, 31(1).

References VIII

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006).
Learning accurate, compact, and interpretable tree annotation.
In *Proc. of COLING-ACL*.

Porter, M. F. (1980).
An algorithm for suffix stripping.
Program, 14(3):130–137.

Ratliff, N., Bagnell, J., and Zinkevich, M. (2006).
Subgradient methods for maximum margin structured learning.
In *ICML Workshop on Learning in Structured Outputs Spaces*.

Ratnaparkhi, A. (1996).
A maximum entropy part-of-speech tagger.
In *Proc. of EMNLP*.

Ratnaparkhi, A., Roukos, S., and Ward, R. T. (1994).
A maximum entropy model for parsing.
In *Proc. of ICSLP*.

Rosenfeld, R. (1997).
A whole sentence maximum entropy language model.
In *Proc. of ASRU*.

Roth, D. and Yih, W.-T. (2004).
A linear programming formulation for global inference in natural language tasks.
In *Proc. of CoNLL*.

Schabes, Y. (1992).
Stochastic lexicalized tree-adjoining grammars.
In *Proc. of COLING*.

References IX

Shen, L., Satta, G., and Joshi, A. K. (2007).
Guided learning for bidirectional sequence classification.
In *Proc. of ACL*.

Smith, D. A. and Eisner, J. (2008).
Dependency parsing by belief propagation.
In *Proc. of EMNLP*.

Smith, D. A. and Smith, N. A. (2007).
Probabilistic models of nonprojective dependency trees.
In *Proc. of EMNLP-CoNLL*.

Smith, N. A. and Eisner, J. (2005).
Contrastive estimation: Training log-linear models on unlabeled data.
In *Proc. of ACL*.

Smith, N. A. and Johnson, M. (2007).
Weighted and probabilistic context-free grammars are equally expressive.
Computational Linguistics, 33(4):477–491.

Smith, N. A., Vail, D. L., and Lafferty, J. D. (2007).
Computationally efficient M-estimation of log-linear structure models.
In *Proc. of ACL*.

Tarjan, R. E. (1977).
Finding optimum branchings.
Networks, 7(1):25–36.

Taskar, B., Guestrin, C., and Koller, D. (2003).
Max-margin Markov networks.
In *Advances in NIPS 16*.

References X

Taskar, B., Lacoste-Julien, S., and Jordan, M. (2005).
Structured prediction via the extragradient method.
In *NIPS*.

Teh, Y. W. (2006).
A hierarchical bayesian language model based on Pitman-Yor processes.
In *Proc. of ACL*.

Thompson, C. A., Mooney, R. J., and Tang, L. R. (1997).
Learning to parse natural language database queries into logical form.
In *Proc. of Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003).
Feature-rich part-of-speech tagging with a cyclic dependency network.
In *Proc. of HLT-NAACL*.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005).
Large margin methods for structured and interdependent output variables.
Journal of Machine Learning Research, 6:1453–1484.

Wang, M., Smith, N. A., and Mitamura, T. (2007).
What is the Jeopardy model? a quasi-synchronous grammar for QA.
In *Proc. of EMNLP-CoNLL*.

Xu, L., Wilkinson, D., Southey, F., and Schuurmans, D. (2006).
Discriminative unsupervised learning of structured predictors.
In *Proc. of ICML*.

Zettlemoyer, L. S. and Collins, M. (2005).
Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars.
In *Proc. of UAI*.