Feature-Based Classifiers

- Exponential (log-linear, maxent, logistic, Gibbs) models:
  - Have features \( f(c, d) \in \mathbb{R} \), with weights \( \lambda_i \), often indicator functions of a condition and class \( f(c, d) = \Phi(c, d, x) \).
  - Use the linear combination \( \sum \lambda_i f_i(c, d) \) to produce a probabilistic model:
    \[
    P(c | d, \lambda) = \frac{\exp \left( \sum \lambda_i f_i(c, d) \right)}{\sum \exp \left( \sum \lambda_i f_i(c, d) \right)}
    \]
    - Makes votes positive.
    - Normalizes votes.
  - The weights are the parameters of the probability model, combined via a "soft max" function.
  - Choose parameters \( \lambda \) that maximize the conditional likelihood of the data according to this model.

Feature Overlap

- Maxent models handle overlapping features well.
- Unlike a NB model, there is no double counting!

Example: NER Overlap

Grace is correlated with PERSON, but does not add much evidence on top of already knowing prefix features.

Feature Weights

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>PERS</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current word</td>
<td>Grace</td>
<td>0.03</td>
<td>0.06</td>
</tr>
<tr>
<td>Beginning bigram</td>
<td>at</td>
<td>0.45</td>
<td>0.04</td>
</tr>
<tr>
<td>Current POS tag</td>
<td>NNP</td>
<td>0.45</td>
<td>0.45</td>
</tr>
<tr>
<td>Prev and cur tags</td>
<td>IN NNP</td>
<td>-0.10</td>
<td>0.14</td>
</tr>
<tr>
<td>Previous state</td>
<td>Other</td>
<td>-0.70</td>
<td>-0.92</td>
</tr>
<tr>
<td>Current signature</td>
<td>x</td>
<td>0.80</td>
<td>0.46</td>
</tr>
<tr>
<td>Prev word, cur sig</td>
<td>O-x</td>
<td>0.68</td>
<td>0.17</td>
</tr>
<tr>
<td>Prev cur tag</td>
<td>NNP</td>
<td>-0.69</td>
<td>0.37</td>
</tr>
<tr>
<td>P. state, p cur sig</td>
<td>O-x</td>
<td>-0.20</td>
<td>0.82</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>-0.58</td>
<td>2.68</td>
</tr>
</tbody>
</table>

Feature Interaction

- Maxent models handle overlapping features well, but do not automatically model feature interactions.

If you want interaction terms, you have to add them:

A disjunctive feature would also have done it (alone):
Feature Interaction

- For loglinear/logistic regression models in statistics, it is standard to do a greedy stepwise search over the space of all possible interaction terms.
- This combinatorial space is exponential in size, but that’s okay as most statistics models only have 4–8 features.
- In NLP, our models commonly use hundreds of thousands of features, so that’s not okay.
- Commonly, interaction terms are added by hand based on linguistic intuitions.

Classification

- What do these joint models of $P(\lambda)$ have to do with conditional models $P(C|D)$?
- Think of the space $C \times D$ as a complex $X$.
  - $C$ is generally small (e.g., 2-100 topic classes)
  - $D$ is generally huge (e.g., space of documents)
- We can, in principle, build models over $P(C,D)$.
- This will involve calculating expectations of features (over $C \times D$):
  $$E(f) = \sum_{d \in D, c \in C} P(c,d)f(c,d)$$
- Generally impractical: can’t enumerate $d$ efficiently.

Classification II

- $D$ may be huge or infinite, but only a few $d$ occur in our data.
- What if we add one feature for each $d$ and constrain its expectation to match our empirical data?
  $$V(d) \in D \quad P(d) = \hat{P}(d)$$
- Now, most entries of $P(c,d)$ will be zero.
- We can therefore use the much easier sum:
  $$E(f) = \sum_{d \in D, c \in C} P(c,d)f(c,d)$$

Classification III

- But if we’ve constrained the $D$ marginals
  $$V(d) \in D \quad P(d) = \hat{P}(d)$$
  then the only thing that can vary is the conditional distributions:
  $$P(c,d) = P(c|d)\hat{P}(d)$$
  $$= P(c|d)\hat{P}(d)$$
- This is the connection between joint and conditional maxent / exponential models:
  - Conditional models can be thought of as joint models with marginal constraints.
  - Maximizing joint likelihood and conditional likelihood of the data in this model are equivalent!

Example: NER Interaction

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>PERS</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>at</td>
<td>0.73</td>
<td>0.94</td>
</tr>
<tr>
<td>Current word</td>
<td>Grace</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>Beginning bigram</td>
<td>at</td>
<td>0.45</td>
<td>0.04</td>
</tr>
<tr>
<td>Current POS tag</td>
<td>NNP</td>
<td>0.47</td>
<td>0.45</td>
</tr>
<tr>
<td>Prev and context</td>
<td>IN NNP</td>
<td>-0.16</td>
<td>0.14</td>
</tr>
<tr>
<td>Prev state</td>
<td>Other</td>
<td>-0.70</td>
<td>0.92</td>
</tr>
<tr>
<td>Prev state, cur sig</td>
<td>O Xx</td>
<td>0.86</td>
<td>0.46</td>
</tr>
<tr>
<td>Prev cur tag, sig</td>
<td>O Xx</td>
<td>0.68</td>
<td>0.17</td>
</tr>
<tr>
<td>P state, p-c cur sig</td>
<td>O Xx</td>
<td>-0.20</td>
<td>0.42</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>0.58</td>
<td>2.45</td>
</tr>
</tbody>
</table>

Easy Quiz Question!

- Suppose we have a 1 feature maxent model built over observed data as shown.
- What is the constructed model’s probability distribution over the four possible outcomes?

<table>
<thead>
<tr>
<th>Empirical</th>
<th>Features</th>
<th>Expectations</th>
<th>Probabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B  2 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C  2 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D  1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Issues of Scale

• Lots of features:
  – NLP maxent models can have millions of features.
  – Even storing a single array of parameter values can have a substantial memory cost.

• Lots of sparsity:
  – Overfitting very easy – need smoothing!
  – Many features seen in training will never occur again at test time.

• Optimization problems:
  – Feature weights can be infinite, and iterative solvers can take a long time to get to those infinities.

Smoothing: Issues

• Assume the following empirical distribution:

<table>
<thead>
<tr>
<th>Heads</th>
<th>Tails</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>t</td>
</tr>
</tbody>
</table>

• Features: (Heads), (Tails)

We’ll have the following model distribution:

\[
P_{\text{heads}} = \frac{e^{\lambda h}}{e^{\lambda h} + e^{\lambda t}} \quad P_{\text{tails}} = \frac{e^{\lambda t}}{e^{\lambda h} + e^{\lambda t}}
\]

• Really, only one degree of freedom \( (\lambda = \lambda_H - \lambda_T) \)

Smoothing: Early Stopping

• In the 4/0 case, there were two problems:
  – The optimal value of \( \lambda \) was \( \infty \), which is a long trip for an optimization procedure.
  – The learned distribution is just as spiked as the empirical one – no smoothing.

• One way to solve both issues is to just stop the optimization early, after a few iterations.
  – The value of \( \lambda \) will be finite (but presumably big).
  – The optimization won’t take forever (clearly).
  – Commonly used in early maxent work.

Smoothing: Priors (MAP)

• What if we had a prior expectation that parameter values wouldn’t be very large?
• We could then balance evidence suggesting large parameters (or infinite) against our prior.
• The evidence would never totally defeat the prior, and parameters would be smoothed (and kept finite!).
• We can do this explicitly by changing the optimization objective to maximum posterior likelihood:

\[
\log P(C, \lambda | D) = \log P(\lambda) + \log P(C | D, \lambda)
\]

Smoothing: Priors

• Gaussian, or quadratic, or L2 priors:
  – Intuition: parameters shouldn’t be large.
  – Formalization: prior expectation that each parameter will be distributed according to a gaussian with mean \( \mu \) and variance \( \sigma^2 \).

\[
P(\lambda) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\lambda - \mu)^2}{2\sigma^2}\right)
\]

• Penalizes parameters for drifting too far from their mean prior value (usually \( \mu=0 \)).
  – \( 2\sigma^2=1 \) works okay.
Example: NER Smoothing

Because of smoothing, the more common prefix and single-tag features have larger weights even though entire-word and tag-pair features are more specific.

<table>
<thead>
<tr>
<th>Local Context</th>
<th>Feature Type</th>
<th>Feature</th>
<th>PERS</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prev</td>
<td>Cur</td>
<td>Next</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>Other</td>
<td>at</td>
<td>0.73</td>
<td>0.94</td>
</tr>
<tr>
<td>Word</td>
<td>Grace</td>
<td>0.03</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Beginning Weight</td>
<td>IN</td>
<td>0.45</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>Current-Seg Log</td>
<td>NNP</td>
<td>0.47</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>Prev and cur tags</td>
<td>IN NNP</td>
<td>-0.10</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>Previous state</td>
<td>Other</td>
<td>-0.70</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>Current signature</td>
<td>x</td>
<td>0.80</td>
<td>0.46</td>
<td></td>
</tr>
<tr>
<td>Prev state, cur sig</td>
<td>O-x-x</td>
<td>0.68</td>
<td>0.37</td>
<td></td>
</tr>
<tr>
<td>P. state - p-cur sig</td>
<td>O-x-x</td>
<td>-0.20</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>Prev-cur-next sig</td>
<td>x-Xx-Xx</td>
<td>-0.69</td>
<td>0.37</td>
<td></td>
</tr>
<tr>
<td>P. state - p-cur sig</td>
<td>O-x-x</td>
<td>-0.20</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>-0.58</td>
<td>2.68</td>
<td></td>
</tr>
</tbody>
</table>

Example: POS Tagging

• From Toutanova et al., 2003:

<table>
<thead>
<tr>
<th>Overall Accuracy</th>
<th>Unknown Word Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Smoothing</td>
<td>96.54</td>
</tr>
<tr>
<td>With Smoothing</td>
<td>97.10</td>
</tr>
</tbody>
</table>

• Smoothing helps:
  - Softens distributions.
  - Pushes weight onto more explanatory features.
  - Allows many features to be dumped safely into the mix.
  - Speeds up convergence (if both are allowed to converge!)

Smoothing: Priors

• If we use gaussian priors:
  - Trade off some expectation-matching for smaller parameters.
  - When multiple features can be recruited to explain a data point, the more common ones generally receive more weight.
  - Accuracy generally goes up!

  • Change the objective (assume \( \mu_i = 0 \)):
    \[
    \log P(C, \lambda | D) = \log P(C | D, \lambda) - \log P(\lambda)
    \]
    \[
    \log P(C, \lambda | D) = \sum_{i \in \text{words}} P(c_i | \lambda) + \sum_i \frac{1}{2} \log \frac{\lambda_i}{\sigma^2}
    \]

  • Change the derivative:
    \[
    \frac{\partial \log P(C, \lambda | D)}{\partial \lambda_i} = \text{actual}(f_{i,C}) - \text{predicted}(f_{i,C}) - \frac{\lambda_i}{\sigma^2}
    \]

Smoothing: Virtual Data

• Another option: smooth the data, not the parameters.

  • Example:
    - Equivalent to adding two extra data points.
    - Similar to add-one smoothing for generative models.
  
  • Hard to know what artificial data to create!

Smoothing: Count Cutoffs

• In NLP, features with low empirical counts were usually dropped.
  - Very weak and indirect smoothing method.
  - Equivalent to locking their weight to be zero.
  - Equivalent to assigning them gaussian priors with mean zero and variance zero.
  - Dropping low counts does remove the features which were most in need of smoothing...
  - ... and speeds up the estimation by reducing model size...
  - ... but count cutoffs generally hurt accuracy in the presence of proper smoothing.

  • We recommend: don’t use count cutoffs unless absolutely necessary.

Sequence Inference

- Maximum Entropy Models
- CG / L-BFGS
- Quadratic Penalties
- NLP Issues

- Sequence Model
- Inference
- Optimization
- Smoothing

- Feature Extraction
- Label
- Classifier Type

- Sequence Data
- Local Data
- Feature Extraction
- Smoothing

- CG
- Label
- Optimization
- Sequence Level
- Local Level
- Inference
MEMM inference in systems

- For a Conditional Markov Model (CMM) a.k.a. a Maximum Entropy Markov Model (MEMM), the classifier makes a single decision at a time, conditioned on evidence from observations and previous decisions.
- A larger space of sequences is explored via search

Two ways to Search: Beam Inference

- Beam inference:
  - At each position keep the top \( k \) complete sequences.
  - Extend each sequence in each local way.
  - The extensions compete for the \( k \) slots at the next position.
- Advantages:
  - Fast; and beam sizes of 3–5 are as good or almost as good as exact inference in many cases.
  - Easy to implement (no dynamic programming required).
- Disadvantage:
  - Inexact: the globally best sequence can fall off the beam.

Two ways to Search: Viterbi Inference

- Viterbi inference:
  - Dynamic programming or memoization.
  - Requires small window of state influence (e.g., past two states are relevant).
- Advantages:
  - Exact: the global best sequence is returned.
- Disadvantage:
  - Harder to implement long-distance state-state interactions (but beam inference tends not to successfully capture long-distance resurrection of sequences anyway).

Part-of-speech tagging: Generative HMM Tagging Models of Brants 2000

- Highly competitive with other state-of-the-art models
- Trigram HMM with smoothed transition probabilities
- Capitalization feature becomes part of the state – each tag state is split into two e.g.
  \( \text{NN} \rightarrow \text{<NN,cap>}, \text{<NN,not cap>} \)
- Suffix features for unknown words

\[
P(w|\text{tag}) = P(\text{tag}|w) P(w|\text{suffix})
= \hat{P}(\text{suffix}) \hat{P}(\text{tag}|\text{suffix}) \hat{P}(\text{tag})
\]

\[
\hat{P}(\text{tag}|\text{suffix}) = \lambda_1 \hat{P}(\text{tag}|\text{suffix}) + \lambda_2 \hat{P}(\text{tag}|\text{suffix}_1) + \ldots + \lambda_N \hat{P}(\text{tag})
\]

MEMM Tagging Models -II

- Ratnaparkhi (1996): local distributions are estimated using maximum entropy models
  - Previous two tags, current word, previous two words, next two words, suffix, prefix, hyphenation, and capitalization features for unknown words
- Toutanova et al. (2003)
  - Richer features, bidirectional inference, better smoothing, better unknown word handling

<table>
<thead>
<tr>
<th>Model</th>
<th>Overall Accuracy</th>
<th>Unknown Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMM (Ratn. 1996)</td>
<td>96.63</td>
<td>85.56</td>
</tr>
<tr>
<td>HMM (Brants 2000)</td>
<td>96.7</td>
<td>85.5</td>
</tr>
<tr>
<td>MEMM (T. et al 2003)</td>
<td>97.24</td>
<td>89.04</td>
</tr>
</tbody>
</table>
Another sequence model: Conditional Random Fields (CRFs)

A whole-sequence conditional model rather than a chaining of local models.

The space of $c$'s is now the space of sequences

- But if the features $f$ remain local, the conditional sequence likelihood can still be calculated exactly using dynamic programming

Training is slower, but CRFs avoid causal-competition biases

These (or a variant using a max margin criterion) are seen as the state-of-the-art these days, and fairly standardly used

NER Results: CoNLL (2003) Named Entity Recognition task

Task: Predict semantic label of each word in text

<table>
<thead>
<tr>
<th>Foreign</th>
<th>NNP</th>
<th>I-NP</th>
<th>ORG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ministry</td>
<td>NNP</td>
<td>I-NP</td>
<td>ORG</td>
</tr>
<tr>
<td>spokesman</td>
<td>NN</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>Shen</td>
<td>NNP</td>
<td>I-NP</td>
<td>PER</td>
</tr>
<tr>
<td>Guofang</td>
<td>NNP</td>
<td>I-NP</td>
<td>PER</td>
</tr>
<tr>
<td>told</td>
<td>VBD</td>
<td>I-VP</td>
<td>O</td>
</tr>
<tr>
<td>Reuters</td>
<td>NNP</td>
<td>I-NP</td>
<td>ORG</td>
</tr>
</tbody>
</table>

Standard evaluation is per entity, not per token

NER Results: Discriminative Model

Increases from better features, a better classification model.

Sequence models? CoNLL 2003 NER shared task Results on English Devset

CoNLL NER Results: CMM Order

Using 3 words works significantly better than using only the current word and the previous two or three tags instead! (Toutanova et al. 2003)
Biomedical NER Motivation

• The biomedical world has a huge body of information, which is growing rapidly.
  – MEDLINE, the primary research database serving the biomedical community, currently contains over 12 million abstracts, with 60,000 new abstracts appearing each month.
  – There is also an impressive number of biological databases containing information on genes, proteins, nucleotide and amino acid sequences, including GenBank, Swiss-Prot, and FlyBase; each contains entries numbering from the thousands to the millions and are multiplying rapidly.
  • Currently, these resources are curated by hand by expert annotators at enormous expense.

Named Entity Recognition

• General NER vs. Biomedical NER

>Christopher Manning< is a professor at >Stanford University<, in >Palo Alto<. >TAR< is an independent transactivation by >Tat< in cells derived from the >CNS< - a novel mechanism of >HIV-1 gene< regulation.

Why is this difficult?

• The list of biomedical entities is growing.
  – New genes and proteins are constantly being discovered, so explicitly enumerating and searching against a list of known entities is not scalable.
  – Part of the difficulty lies in identifying previously unseen entities based on contextual, orthographic, and other clues.
• Biomedical entities don’t have strict naming conventions.
  – Common English words such as period, curved, and for are used for gene names.
  – Entity names can be ambiguous. For example, in FlyBase, “dk” is the gene symbol for the “Clock” gene but it also is used as a synonym of the “period” gene.
• Biomedical entity names are ambiguous
  – Experts only agree on whether a word is even a gene or protein 69% of the time! (Krauthammer et al., 2000)
  – Often systematic polysemies between gene, RNA, DNA, etc.

Finkel et al. (2004) Results

• BioNLP task – Identify genes, proteins, DNA, RNA, and cell types

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>68.6%</td>
<td>71.6%</td>
<td>70.1%</td>
</tr>
</tbody>
</table>

F1 = 2(Precision)(Recall) / (Precision + Recall)

Features: What’s in a Name?

Interesting Features

– Word, and surrounding context
– Word Shapes
  • Map words to simplified representation that encodes attributes such as length, capitalization, numerals, Greek letters, internal punctuation, etc.

<table>
<thead>
<tr>
<th>Varicella-zoster</th>
<th>Xx-XXX</th>
</tr>
</thead>
<tbody>
<tr>
<td>mRNA</td>
<td>xXXX</td>
</tr>
<tr>
<td>CPA1</td>
<td>XXXd</td>
</tr>
</tbody>
</table>

– Character substrings

<table>
<thead>
<tr>
<th>Company</th>
<th>Movie</th>
<th>Place</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cotrimoxazole</td>
<td>Wethersfield</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alien Fury: Countdown to Invasion</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The full task of “Information Extraction”

As a family of techniques:

\[
\text{Information Extraction} = \text{segmentation} + \text{classification} + \text{association} + \text{clustering}
\]

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a “cancer” that stifled technological innovation. Today, Microsoft claims to “love” the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels—the coveted code behind the Windows operating system—to select customers.

“We can be open source. We love the concept of shared source,” said Bill Veghte, a Microsoft VP. “That’s a super-important shift for us in terms of code access.”

Richard Stallman, founder of the Free Software Foundation, countered saying…

Landscape of IE Tasks (1/4):
Degree of Formatting

Text paragraphs without formatting
Grammatical sentences and some formatting & links
Non-grammatical snippets, rich formatting & links
Tables

Landscape of IE Tasks (2/4):
Intended Breadth of Coverage

Web site specific
Genre specific
Wide, non-specific

Amazon.com Book Pages
Resumes
University Names
Formatting
Layout
Language

Landscape of IE Tasks (3/4):
Complexity

E.g. word patterns:

Closed set
U.S. states
He was born in Alabama.
The big oldating sky.

Regular set
U.S. phone numbers
Phone: (413) 545-1323
The CALD main office can be reached at 412-268-1299

Complex pattern
University of Arkansas
P.O. Box 140
Rogel, AR 71625

Ambiguous patterns, needing context and many sources of evidence
Person names:
...was among the six houses sold by Hope Fieldman that year.

Pawel Opalinski, Software Engineer at WhizBang Labs.

Landscape of IE Tasks (4/4):
Single Field/Record

Single entity:
Person: Jack Welch
Location: Connecticut

Binary relationship:
Person-Title
Person: Jack Welch
Title: CEO

N-ary record:
Succession
Company-Location
Company: General Electric
Location: Connecticut

Relation:
Person-Title
Out: Jack Welch
In: Jeffrey Immelt

“Named entity” extraction