# Hybrid Information Extraction Systems for Open Data

## Class 4: Statistical IE

Pablo Ariel Duboue, PhD

Curso de Posgrado Cs. de la Computacion
FaMAF-UNC

# Why Statistical IE?

- Cost reduction
- Measurable quality
- Learning analog
- Generalize over training

# Example Training Data

- From WNUT NER competition
  - tonite O
  - " O
  - running B-tvshow
  - wit I-tvshow
  - mjd I-tvshow
  - " O
  - live O
  - from O
  - 7- O
  - 9pm O
  - eastern O
  - sirius B-company
  - 211 O
  - . O

# Example Features

**O** w[-1]=*planning* w[0]=*the* w[-1]|w[0]=planning|the w[0]|w[1]=the|next DICT=tv.tv_program DICT=people.person word=the word_lower=the
**O** w[-1]=*the* w[0]=*next* w[-1]|w[0]=the|next w[0]|w[1]=next|Disney DICT=tv.tv_program DICT=people.person word=next word_lower=next prefix=n prefix=ne prefix=nex suffix=t suffix=xt
**B-facility** w[-1]=*next* w[0]=*Disney* w[-1]|w[0]=next|Disney w[0]|w[1]=Disney|World DICT=tv.tv_program DICT=people.person word=Disney word_lower=disney prefix=d prefix=di prefix=dis suffix=y suffix=ey suffix=ney INITCAP INITCAP_AND_GOODCAP
**I-facility** w[-1]=*Disney* w[0]=*World* w[-1]|w[0]=Disney|World w[0]|w[1]=World|trip DICT=tv.tv_program DICT=people.person word=World word_lower=world prefix=w prefix=wo prefix=wor suffix=d suffix=ld suffix=rld INITCAP INITCAP_AND_GOODCAP
**O** w[-1]=*World* w[0]=*trip* w[-1]|w[0]=World|trip w[0]|w[1]=trip|. DICT=tv.tv_program DICT=people.person word=trip word_lower=trip prefix=t prefix=tr prefix=tri suffix=p suffix=ip suffix=rip

# Type of Models

- Maximum Entropy
- Conditional Random Fields
- Generalized Graphical Models
- Deep Learning

# Sequence Tagging

- The main problem for applying traditional ML approaches to sequence tagging is the variable size of the input.
- P(first word being of class Company | first word is Disney and second word is Channel) << P(first word being of class Company | first word is Disney and second word is Pictures)
- Markov assumption:
  - The value at time t is only dependent of the value at times t-1, ..., t-k (where k is the **order** of the Markov model)
- Using the Markov assumption we can then train models to make local + context (of order k) decisions.

# Begining-Inside-Out revisited

- The straightforward approach is to use classes "word is under tag-X" (tag-X) "word is not under any tag" (OTHER)
  - But the boundaries are different than the behavior inside a tag (or other)
- We therefore use classes "word starts tag-X" (B-X) "word expands tag-X" (I-X) "word is not under any tag" (O)
  - Given n classes, this means 2n+1 tags

.

# Features

lexical
: the actual lexical item for the current and contextual words

dictionary
: whether the word is in certain word lists (names of companies, countries, states, cities, common first names)

shape
: the ortographic form of the token (all lower case, capitalized, all caps, numeric, etc)

part-of-speech
: features related

# Relation Extraction

- Classification given two entities. Features: (from "Information Extraction: Capabilities and Challenges" by Ralph Grishman)
    - their heads
    - their types (person, organization, ...)
    - their the distance in words
    - the words in between
    - the dependency path between them
    - the words on the dependency path
- Output:
    - true (there's a given relation) or false
    - relation-1 ... relation-n or no-relation (multi-label classification)

# Generative vs. Discriminative ML

- For ML using statistical methods
  - We want the probability (likelihood) of the target given the input features
    - If we have modeled the joint distribution of input features and target class we can obtain this
  - However, that is not required to model the conditional probability
  - Simulation vs. emulation

# Generative Models

- Compute $P(y|x_1, \ldots, x_n)$ via $P(x_1, \ldots, x_n, y)$
- The joint probability enables reversible systems
  - Any variable can be made the target class
- Requieres a "generative story" of how the data came to be and its inter relations
  - Dependencies between variables
- More parameters, therefore needs more data and/or make less efficient use of the data
  - If we only care about the target class, we are modeling too much
  - Modeling a predictive keyboard **and** an information extraction system

# Discriminative Models

- Just model $P(y|x_1, \ldots, x_n)$
- Many times not even model the probability but unnormalized probabilities
  - Likelihoods
  - Enough to distinguish among different values of the target class
- Better in practice
- Less theoretical advantages

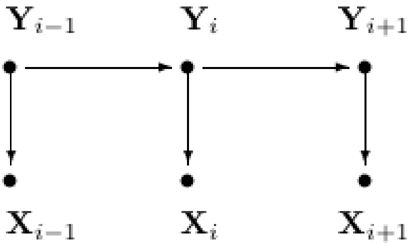
# Conditional Random Fields

- Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data by John Lafferty, Andrew McCallum, Fernando Pereira (ICML 2001)
- Cited by 9083

# Nomenclature

- X = observations (e.g., sentences, i.e., a sequence of words)
- Y = labels (e.g., POS)
- Assume a one-to-one correspondance between states and labels

# HMMs




from Lafferty et al. (2001), Figure 2 (a)

- HMMs and stochastic grammars assigns a joint probability to paired observation and label sequences
  - Trained to maximize the joint likelihood of the training examples
  - Because it is joint needs to enumerate all observation sequences

# HMMs Tasks: Probability of Observations

- The probability of observing a sequence of length $L$:
  - $Y = y(0), y(1), \ldots, y(L-1)$
- is given by :
  - $P(Y) = \sum_X P(Y \mid X) P(X)$
- that sums over all possible hidden-node sequences:
  - $X = x(0), x(1), \ldots, x(L-1)$

# HMMs Tasks: Most Likely Explanation

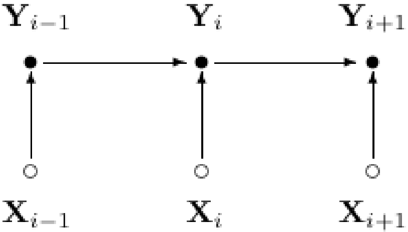
- Viterbi decoding

https://en.wikipedia.org/wiki/Viterbi_algorithm#/media/File:Viterbi_an

# Conditional Models

- Model the probability of labels given the observations
  - No modeling effort for fixed observations
- Observations can be correlated and refer to different levels of abstraction
  - Words and characters, for example

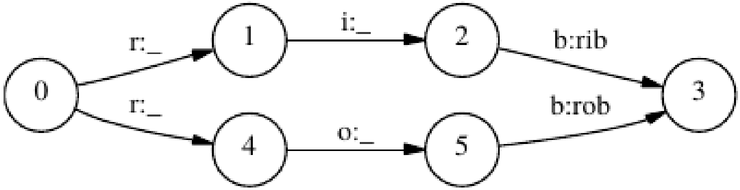
# MEMMs




from Lafferty et al. (2001), Figure 2 (b)

- Maximum Entropy Markov Models: exponential models at each state trained by iterative scaling Maximum Entropy
- Belong to the general class of "next state classifiers"

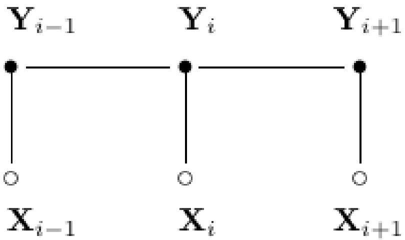
# Problem: Label bias




from Lafferty et al. (2001), Figure 1

- In next state classifiers the Markov assumption can lead to dead ends
  - You shouldn't be in a given state but if you want to estimate what would happen if you are there, there's only one way out and it'll be "highly likely" (i.e., misleading)

# CRFs




from Lafferty et al. (2001), Figure 2 (c)

- CRFs have a single exponential model for the joint probability of the sequence

# CRF Graph

**Definition.** *Let $G = (V, E)$ be a graph such that $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$, so that $\mathbf{Y}$ is indexed by the vertices of $G$. Then $(\mathbf{X}, \mathbf{Y})$ is a* conditional random field *in case, when conditioned on $\mathbf{X}$, the random variables $\mathbf{Y}_v$ obey the Markov property with respect to the graph: $p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \sim v)$, where $w \sim v$ means that $w$ and $v$ are neighbors in $G$.*

from Lafferty et al. (2001), Page 3

- Markov property given a graph: only connected nodes in the graph are dependent
- In IE, graph = simple chain

https://github.com/factorie/factorie

# Vertex and Edge Features

$$p_\theta(\mathbf{y} \,|\, \mathbf{x}) \propto$$
$$\exp\left( \sum_{e \in E, k} \lambda_k f_k(e, \mathbf{y}|_e, \mathbf{x}) + \sum_{v \in V, k} \mu_k g_k(v, \mathbf{y}|_v, \mathbf{x}) \right)$$

from Lafferty et al. (2001), Eq 1

# Conditional Probability at State

$$
\begin{aligned}
M_i(y', y \,|\, \mathbf{x}) &= \exp\left(\Lambda_i(y', y \,|\, \mathbf{x})\right) \\
\Lambda_i(y', y \,|\, \mathbf{x}) &= \textstyle\sum_k \lambda_k\, f_k(e_i, \mathbf{Y}|_{e_i} = (y', y), \mathbf{x}) + \\
&\quad \textstyle\sum_k \mu_k\, g_k(v_i, \mathbf{Y}|_{v_i} = y, \mathbf{x})\ ,
\end{aligned}
$$

from Lafferty et al. (2001), page 4

- In a chain, the conditional probability is a square matrix the size of the label vocabulary

# Using M to compute p

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \frac{\prod_{i=1}^{n+1} M_i(\mathbf{y}_{i-1}, \mathbf{y}_i \mid \mathbf{x})}{\left(\prod_{i=1}^{n+1} M_i(\mathbf{x})\right)_{\text{start},\text{stop}}}$$

from Lafferty et al. (2001), page 4

- Assumes two extra elements, start and stop have been added to the sequence
- The notation $()_{i,j}$ indicates the entry at position $i, j$ in a matrix

# Iterative Scaling

- Training involves calculating $\lambda_k$ and $\mu_k$ given the training data
- Iterative scaling transforms this into a weight update problem for suitable deltas:
  - $\lambda_k \leftarrow \lambda_k + \delta\lambda_k$
  - $\mu_k \leftarrow \mu_k + \delta\mu_k$

# ClearTk

- ClearTK was originally developed by the University of Colorado's Center for Computational Language and Education Research (CLEAR).
- Most of it is available under a BSD license
- "ClearTK 2.0: Design Patterns for Machine Learning in UIMA." by Bethard, Ogren, and Becker (LREC 2014)
- Requires a contributor agreement

# ClearTk Architecture

- Annotations
- Annotators
- ML Engines

# UIMA Feature vs. ML features

- UIMA Feature:
  - A particular aspect of an annotation
  - Lives in the CAS
- ML Feature:
  - An entry in the feature vector
  - Used as input to a ML problem (either as train or test data)
  - Many times lives in text files in the file system or as arrays in RAM at runtime

# ClearTk Feature Extractors

https://cleartk.github.io/cleartk/docs/tutorial/feature_extraction.html

# IOB Tagging

```
public class NamedEntityChunker extends CleartkSequenceAnnotator<String> {
  ...
  private BioChunking<Token, NamedEntityMention> chunking = new BioChunking<>(
      Token.class, NamedEntityMention.class, "mentionType");
  ...
  public void process(JCas jCas) throws AnalysisEngineProcessException {
    for (Sentence sentence : JCasUtil.select(jCas, Sentence.class)) {
      // extract features for each token in the sentence
      List<Token> tokens = JCasUtil.selectCovered(jCas, Token.class, sentence);
      List<List<Feature>> featureLists = new ArrayList<>();
      for (Token token : tokens) {
        List<Feature> features = new ArrayList<>();
        features.addAll(this.extractor.extract(jCas, token));
        features.addAll(this.contextExtractor.extract(jCas, token));
        featureLists.add(features);
      }
      // during training, convert NamedEntityMentions in the CAS into expected classifier outcomes
      if (this.isTraining()) {
        // extract the gold (human annotated) NamedEntityMention annotations
        List<NamedEntityMention> namedEntityMentions = JCasUtil.selectCovered(
            jCas, NamedEntityMention.class, sentence);
        // convert the NamedEntityMention annotations into token-level BIO outcome labels
        List<String> outcomes = this.chunking.createOutcomes(jCas, tokens, namedEntityMentions);
        // write the features and outcomes as training instances
        this.dataWriter.write(Instances.toInstances(outcomes, featureLists));
      }
      // during classification, convert classifier outcomes into NamedEntityMentions in the CAS
      else {
        // get the predicted BIO outcome labels from the classifier
        List<String> outcomes = this.classifier.classify(featureLists);
        // create the NamedEntityMention annotations in the CAS
        this.chunking.createChunks(jCas, tokens, outcomes);
      }
    }
  }
}
```

from Bethard et al. (2014), Fig. 1

# IOB Tagging (cont.)

https://cleartk.github.io/cleartk/docs/tutorial/chunking_classifier.html

# Annotating Training Data with RuTA Workbench

- Quick annotation
- Keyboard interface

# ReasonAnnotator

- CRF using CleartkSequenceAnnotator
- Features: same as tutorial - POS (+ under amount or under company)