Named Entities, Named Entity Tagging and Machine Learning

Adam Meyers
New York University
Outline

• What is a Named Entity?
• HMM NE tagging
• Inferences based on less Information
• Combining Evidence: Maximum Entropy
What is a Named Entity?

- **Definition 1**: A single or multi-word expression that meets any of the following criteria:
  - is a proper noun phrase phrase
    - *Adam L. Meyers, PhD.*
    - *Professor Meyers*
    - *New York University*
  - is a proper adjective phrase, e.g., *Latin American*
  - has external distribution of NP, but different internal structure
    - *January 3, 2012*
    - *Five Hundred Thirty*
    - *waffles@cs.nyu.edu*

- **Definition 2**: A class of words and multi-word expressions defined by specifications tuned to information extraction tasks (can conflict with 1 by including “normal” nouns)
  - [http://nlp.cs.nyu.edu/ene/](http://nlp.cs.nyu.edu/ene/) is a large NE hierarchy following definition 2.
What is a Proper Noun (Phrase)?

- Definition: A name of something that is (in English) capitalized even in non-initial position, typically representing unique individual objects. Proper nouns don't typically take determiners.

- What's unique?
  - Is *Adam Meyers* a proper NP even though there are more than one person with that name?
  - Are *Thursday* or *September 3* proper NPs even though there are more than one instance of these days?
  - What about car models such as the *Fiesta* which represent a type of objects rather than a specific object.
  - Color terms, e.g., *azure, salmon, peach, …* identify unique types, just like car models, yet they are not technically proper nouns

- Capitalization can be inconsistent
  - fields of study (like *computer science*) are capitalized inconsistently
  - different languages use different capitalization conventions
Internal Structure of Person Names

- NP → First_Name
- NP → (TitleP)?(First_Name)? (Middle_Name|Initial)?Last_Name (Post_Honorific)?
- TitleP →(Mod)* Title
- Mod → vice | assistant | assist. | deputy, …
- Title → Mr. | Ms. | Mrs. | Miss | Master | Dr. | President, …
- First_Name → Adam | Jenny | Joshua | Nurit | Giancarlo | Ralph | Cristina | Satoshi | Heng | Xiang | Shasha | Wei | Ang | Bonan | …
- Last_Name → Meyers | Matuk | Lee | Grishman | Mota | Sekine | Ji | Li | Liao | Xu | Min | …
- Post_Honorific → Esq. | Jr. | Sr. | I | II | III | PhD. | …
- Note: specifications vary about whether titles and Post_Honorifics are or are not part of the name (ACE excludes titles)
Structure of Organization/Location/… Names

• Many Different Structures Possible
  – *Advanced Micro Devices* (ORG, normal NP)
  – *Council of Indian Nations* (ORG, normal NP)
  – *Yucatan Peninsula* (LOC, normal NP)
  – *United States of America* (GPE, normal NP)
  – *Ford Motors, Inc.* (ORG, NP plus right modifier)
  – *Alcoholics Anonymous* (ORG, NP plus right modifier)
  – *Head, Heart, Hands, Health* (list of nouns)
  – *Alfac* (ORG, newly coined single word)
  – *Addis Abba* (GPE, two foreign words)
  – *Merrill Lynch* (ORG, Person name structure)
  – *Nobody Can Beat the Wiz* (ORG, normal S)
  – *Hi Ho* (SONG, idiom)

• Unambiguous (like fixed phrases)
  – Name of ORG: *Advanced Micro Devices* (Advanced modifies Devices)
  – *[Advanced biology] textbook* vs. *Advanced [biology textbook]*
Some Other Entities

- Numbers and Quantities
  - Twenty Five Thousand, Five Hundred Fifty Eight
  - $200 million

- Times and Dates (not always names)
  - January 3, 2011
  - Ten o'clock
  - 10:30
  - last Thursday
  - St. Valentine's Day

- Addresses (street, email, url, …)
  - 1313 Mockingbird Lane, New York, NY 10003
  - hm1313@cs.nyu.edu
  - http://nlp.cs.nyu.edu/people/meyers.html
ACE Named Entities

• ACE Specifications online (name mentions only)

• GPE – location with a government
  – city, state, county, country
  – people, physical location, government

• Location – geographical location
  – lake, mountain, ..

• Facility – man-made structure
  – bridge, street, building

• Person – person or group of people

• Organization – group of people with structure
  – commercial, government, club, non-profit
The ACE Task

- 2000-2008 Government-sponsored shared tasks (or bake-offs)
- Full Entity task
  - Annotation of mentions
    - Names, common noun, pronoun phrases that fall into the semantic classes (ultimately a superset of previous slide)
  - Coreference
    - Entity = Sets of mentions that refer to the same thing
- Other tasks
  - Relations: between two entities
    - located, part-whole, family, employment, ...
  - Events: entities are arguments of predicates
    - Movement, attack, be_born, marry, die, business_merge, declare_backruptcy, ...
- Languages: English, Chinese, Arabic
Some Historical Notes

• Before ACE, NEs were introduced in 1995 as part of the MUC6 government task

• The ACE task and several other NE tasks extended MUC6 in various ways.

• Other NE tasks, both government and SIG sponsored:
  – CONLL 2002-2003: English, Dutch, German, Spanish
  – IREX 1998-1999: Japanese (co-chairs: Sekine at NYU and Isahara at CRL)
  – SIGHan 2006: Chinese
  – TAC/KBP 2009 – Present: English (NIST)
Markov Chains (review)

• Markov chain is a WFSA in which an input string uniquely determine path through the Automaton
  – Q = set of states: q0 (start state), …, qF (final state)
    • q0 and qF are special in that they are not associated with observations
  – A = transition probability matrix A, each $a_{i,j}$ representing the probability of
    moving from state i to state j, such that $\sum_{j=1}^{n} a_{i,j} = 1 \forall i$

• Assumptions
  – In an N-order markov model, a particular state depends on the previous N states. So far we have focused on first-order models (bigrams)
  – All outgoing edges from a node sum to 1
    • $\sum_{j=1}^{n} a_{i,j} = 1 \forall i$

• Alternative (equivalent) formulation regarding initial/final states
  – Substitute transition probabilities from initial states and from final states with probabilities that particular states will be initial or final.
HMM (review)

- Hidden Markov model combines hidden events (indirect predictions) with Markov chains (transition probabilities are called prior probabilities)

- Adds following 2 things to Markov chains
  - $O_1 \ldots O_T$ – a sequence of $T$ observations
  - $B = b_i(O_t)$ – observation likelihoods – each likelihood that observation $O_t$ will occur, given state $i$

- Additional Assumption: Likelihoods depend only on the states in which they occur
Named Entity Task

• Similar to POS tagging and Chunking

• Typical manual markup
  – `<LABEL> ... </LABEL>` (label = PER, GPE, ...)
  – States in HMM could correspond to:
    • Being inside constituents of each of the labeled types and being outside.

• Example POS/Chunking-like tagset:
  – `B_PER, I_PER, B_GPE, I_GPE, B_ORG, I_ORG, B_LOC, I_LOC, ...`, `NOT_NAME`
  – A popular way to label transitions for HMM (and other) NE taggers.
Nymble: an HMM NE tagger

- NEs: organization, person, location, time, date, percent, money
- Bikel, et. al. (1996) – basis of next few slides
- Name Classes (NC): NE classes + other
- Begin and Internal tags are implied
  - John/PER Smith/PER /OTHER Mary/PER Smith/PER
  - No B-PER tag is mentioned in paper, but priors for initial words in a PER sequence are different than for subsequent elements of PER
- HMM using Viterbi algorithm
- Each word is an ordered pair: <word, features>
  - True/False features involving upper/lowercase/capitalization, digit/letter/punctuation, 1st word, etc.
  - <John <False,...,True,True,...>> Only firstWord and initCap are True
  - <Smith <False,...,False,True,...>> Only initCap is True
  - <, <False,...,True,...>> Only Other is True
- Includes Backoff Model:
  - different (weighted) levels of prior probabilities are combined
  - bigrams, NCs, words, features, …
Nymbol – Probabilities Used

- Probability assumed to consist of:
  - Likelihood (of the word/prob sequence) X Priors (transitions between states)

- Probability of Tag Sequence NC given Input Token Sequence $W$
  
  - $Pr(NC|W) = \frac{Pr(W|NC)}{Pr(W)} \times Pr(NC)$ # Bayes Rule
  
  - $Pr(W|NC) \times Pr(NC) \approx Pr(W|NC) \times Pr(NC)$ # $Pr(W)$ ignored (same for any tag seqs)
  
  - $Pr(W|NC) = \text{Likelihood} \quad Pr(NC) = \text{Prior}$

- Likelihood Approximated as based only on its NC (as with HMM)
  
  - $Pr(W|NC) \approx \prod_{i=1}^{n} P(w_i|nc_i)$
  
  - Backoff: withhold 10–20% of training data for OOV model
    
    - Base probabilities above on words in this subcorpus, but not in the regular training corpus.
    
    - Assume words found only in the held-out (10-20%) are “unknown words” and calculate all of the above probabilities based on the occurrence of these words in this subcorpus.

- Prior calculated differently for different parts of the sequence
  
  - Details on next Slide
Nymbol – Prior Probabilities

- Prior for 1\textsuperscript{st} word of a NC: 
  \[ Pr(\text{NC} | \text{NC}_{-1}, w_{-1}) \times Pr(<w, f>_{\text{first}} | \text{NC}, \text{NC}_{-1}) \]
  - \[ Pr(\text{NC} | \text{NC}_{-1}, w_{-1}) \]  
  - \[ Pr(<w, f>_{\text{first}} | \text{NC}, \text{NC}_{-1}) \]  
    # e.g., Mr. precedes B-PER
    # e.g., O precedes capitalized B-PER

- Prior for subsequent words of NC:
  - \[ Pr(<w, f> | <w, f>_{-1}, \text{NC}) \]  
    # sequences of same/diff class

- Probability that the current word ends an NC:
  - \[ Pr(<+ \text{ end}+, \text{other} > | <w, f>_{\text{final}}, \text{NC}) \]
Nymbol – Backoff for Prior Probabilities

- **Name Class Bigrams**
  
  $Pr(NC|NC_{-1}, w_{-1}) \subset Pr(NC|NC_{-1}) \subset Pr(NC) \ldots \frac{1}{\text{number of NCs}}$

- **First Word Bigrams**
  
  $Pr(<w, f>_fist|NC, NC_{-1}) \subset Pr(<w, f>|<\text{begin}+, \text{other}>, NC) \subset$
  
  $Pr(<w, f>|NC) \subset \ldots Pr(w|NC) \times Pr(f|NC) \subset \frac{1}{\text{vocab_size}} \times \frac{1}{\text{number_features}}$

- **Non-First Word Bigrams**
  
  $Pr(<w, f>|<w, f>_{-1}, NC) \subset Pr(<w, f>|NC) \subset$
  
  $Pr(w|NC) \times Pr(f|NC) \ldots \subset \frac{1}{\text{vocab_size}} \times \frac{1}{\text{number_features}}$
Smoothing (in Nymbol)

- Order Models by amount of Info: $M_1 \subset M_2 \subset M_3 \subset M_4 \ldots$
- Apply weight $\Lambda$ to the back-off model and $1 - \Lambda$ to the initial model
  - This is called smoothing
- $\Lambda$ based on relative sample sizes of $M$ and $M'$
  - In model $M$, $\Pr(X|Y)$ is based on the count of $Y$ (more info)
  - In model $M'$, $\Pr(X|Y')$ is based on the count of $Y'$ (backoff model)
  - $c(Y') > c(Y)$ e.g., suppose $Y = NC_{-1 \cdot w_{-1}}$ and $Y' = NC_{-1}$
- $\Lambda$ favors backing off to more frequent and less diverse models

\[\lambda = \left(1 - \frac{c(Y)}{c(Y')}\right) \times \frac{1}{1 + \frac{\text{unique_outcomes}(Y')}{c(Y')}}\]

- $1^{st}$ factor: Positive if $Y' > Y$ and increases as $Y'$ increases
- $2^{nd}$ factor: .5 if $Y'$ is maximally diverse and approaches 1 as the number of diverse outcomes decreases to 1
If Lots of Evidence, Do Machine Learning

• Suppose you want to combine lots of features together and take advantage of any correlation to predict outcomes
• Methods for doing this fall into the area called machine learning
• These methods include: Maximum Entropy, Support Vector Machines, Naive Bayes, Conditional Random Fields, Neural Networks, and several others.
• Supervised or Unsupervised
  – **Supervised**: Methods in which statistical models are “trained” based on manually annotated text. *
    • We will focus on these.
  – **Unsupervised**: Methods in which statistical models are based on assumptions about un-annotated data
High Level Description of ML

• Input:
  – Correctly Annotated Data
    • Training Corpus
    • Test or Development Corpus
  – An observable set of features
• Machine Learning Algorithms
  – Methods for combining evidence and making predictions
• Tookits for Multiple Machine Learning Algorithms
  – JAVA
    • WEKA: http://www.cs.waikato.ac.nz/ml/weka/
    • MALLET: http://mallet.cs.umass.edu/
  – Python
    • NLTK's classification package (Chapter 6)
    • Also: http://scikit-learn.org/ [I know less about this one]
Making and Tuning ML Systems

• Experiment with Different ML Algorithms
  – Use the same set of features
  – Toolkits make switching easy
  – May help to understand some differences
    • Speed/complexity → limit size of training data
    • Assumptions about Feature Independence
  – Tweaking features, making new algorithms and making new more efficient versions of current ML algorithms

• Experiment with Different Sets of Features
  – Keep algorithm fixed
  – Vary numbers of features
  – Possible strategy: use as many features as possible
    • When these systems work, it cannot always be explained why
  – Possible strategy: use features that can be expected to make a prediction

• Possible to make an excellent ML system while treating algorithms as black boxes
Log Linear Classifiers (Important for Understanding Maximum Entropy)

• A log linear classifier
  – Extract features (real number) from input
  – Multiply each feature by a weight
  – Use this total as an exponent

\[
p(c|x) = \frac{1}{Z} \times e^{\sum_i w_i f_i}
\]

• \(c = \text{class}, \ x = \text{observation}, \ Z \text{ normalizing factor}, \ w_i \ \text{and} \ f_i \)
  are features and weights (both depending on \(c\))
• \(Z \) makes all probabilities sum to 1
• \(e = \text{mathematical constant, approximately} \ 2.718\)
Linear Regression

• Tasks that map input features to output
  – linear regression (real numbers)
  – linear classifier (discrete classes)

• Combining feature weights
  – \( y = \sum_{i=0}^{N} w_i \times f_i \) assuming \( f_0 = 1 \)
  – Expressed compactly in dot product notation: \( y = w \cdot f \)

• Regression line \((y = mx + b)\) line that fits data (for features \(x,y)\)
  – \( m \) that minimizes cost of difference of predicted \( (y_{pred}^{(j)}) \) vs observed \( (y_{obs}^{(j)}) \)
  – \( y_{pred}^{(j)} = \sum_{i=0}^{M} w_i \times f_i^{(j)} \)
  – \( \text{cost} \ (W) = \sum_{j=0}^{M} (y_{pred}^{(j)} - y_{obs}^{(j)})^2 \)
  – Normalize cost by squaring, not absolute value
    • Outliers have an effect, adding absolute values would allow them to be ignored
  – Deriving Weights (proof/implementation omitted)
    • Let \( M \) = a matrix: observations = columns, features = rows
    • Let \( v \) = a vector of predicted values
    • \( W = (M^T \cdot M)^{-1} \cdot M^T \cdot \tilde{v} \)
Logistic Regression

- If we assume binary values (true|false or 1|0)
  - $p(y=\text{true}|x)=\frac{e^{w \cdot f}}{1+e^{w \cdot f}}$ and $p(y=\text{false}|x)=\frac{e^{-w \cdot f}}{1+e^{-w \cdot f}}$
  - The dot product of features:
    - $w \cdot f = \ln\left(\frac{p(\text{true})}{p(\text{false})}\right)$
    - A number between positive and negative infinity
  - Our observation should be labeled true if:
    - $p(\text{true}|x) > p(\text{false}|x)$
    - Or if $w \cdot f = \ln\left(\frac{p(\text{true})}{p(\text{false})}\right) > 0$
  - This equation is the hyperplane dividing the space of features into 2 predicted outcomes.
    - View features as dimensions for Cartesian Geometry
    - Learning these weights will not be covered here
Maximum Entropy

- Multinomial logistic regression: generalization of logistic regression to cover more than 2 classes, aka, Maximum Entropy
- Features have 2 values: 1 (True) or 0 (False)
- Linear regression for classes $C = \{c_1, \ldots, c_C\}$

$$p(c | x) = \frac{1}{Z} \times e^{\sum w_i f_i}$$

$$Z = \sum_{c' \in C} e^{\sum_{i=0}^{N} W_{c_i} f_i}$$

$$p(c | x) = \frac{\sum_{i=0}^{N} w_{c_i} f_i(c, x)}{\sum_{c' \in C} \sum_{i=0}^{N} w_{c'} f_i(c', x)}$$
Maximum Entropy 2

- For each observation x and class c, we can find the probability of c given x:
  \[
p(c | x) = \frac{\sum_{i=0}^{N} w_{ci} f_i(c, x)}{\sum_{c' \in C} \sum_{i=0}^{N} w_{c'i} f_i(c', x)}
  \]

- We can choose the most probable classification:
  \[
  \hat{c} = \arg\max_{c \in C} P(c | x)
  \]

- Or the most probable sequence of classifications as in a MEMM (Maximum Entropy Markov Model)

- Principle of Maximum Entropy: the principle best representing the current state of knowledge is the principle consistent with the data that has the highest entropy (level of uncertainty)
MEMM

- Most probable tag set $T$ given the word sequence $W$
  \[ \hat{T} = \arg\max_T P(T \mid W) \]

- Prob of states $Q = q_1, \ldots, q_n$ given observations $O = o_1, \ldots, o_n$ when MEMM is simulating an HMM:
  \[ P(Q \mid O) = \prod_{i=1}^{n} P(q_i \mid q_{i-1}, o_i) = \prod_{i=1}^{n} P(o_i \mid q_i) \times \prod_{i=1}^{n} P(q_i \mid q_{i-1}) \]

- MEMMs can and do incorporate more features.
  - HMM features, capitalization features, Nymbol-like features, prefixes, suffixes, letter combinations (which may indicate word origin), etc.

- Other “Machine Learning” paradigms: Bayes' networks, Support Vector Machines, Perceptron, ...
Readings

• J & M Chapter 6
• NLTK – Sections 6 and 7.5
• ACE Named Entity Specifications
  – Read First 3 sections
• Bikel, et. al. (1997). *Nymble: a High-Performance Learning Name-finder*. In 5th Conference on Applied NLP
Homework 6 – Slide 1

• This homework involves setting up features and running a MEMM, without getting into the details of Maximum Entropy – this is an exercise in “feature engineering”, not in perfecting or implementing machine learning algorithms.

• Download and unpack the following file
  – [Link](http://cs.nyu.edu/courses/fall15/CSCI-UA.0480-006/NounGroupsforNLPClass.zip)
  – This is an expanded version of the data from homework 4 – it encodes Noun Groups

• If you are using JAVA to write your program, download and unpack this file:
  – This is the OpenNLP Maxent package

• If you are not using JAVA, download and unpack this file:
  – [Link](http://cs.nyu.edu/courses/fall15/CSCI-UA.0480-006/MaxentWrapper.zip)
  – This is wrapper allows you to run the OpenNLP maxent package if you do not write your code in JAVA
Homework 6 – Slide 2
The Data and The Scorer

• NounGroupsforNLPClass.zip includes the data and the scoring program
• train.np and dev.np are modified versions of train.pos and dev.pos from homework 4
  – They contain BIO tags in an additional column
    • B-NP – indicates a token begins a Noun Group
    • I-NP – indicates a token is inside a Noun Group
    • O – indicates a token is outside of a Noun Group
• The scoring program and instructions include:
  – conlleval – the evaluation program
  – sample.test – a sample system output file
  – Scorer_Instructions
Homework 6 – Slide 3

• Your system will take dev.np as input and append to the end of each line a space, followed by your system's predicted class (B-NP, I-NP or O), just like sample.test

• Then you can run the scorer
  – conlleval < yourOutputFile

• The scorer will give your system's precision and recall.
Homework 6 – Slide 4
Using the OpenNLP Maxent Package

- Input to the Maxent package should be in the following format:
  - Token $f_1 = v_1 f_2 = v_2 f_3 = v_3 ...$
  - Where $f_n = v_n$ are a feature and value pair, e.g., current_POS=NN, previous_POS=NNP, etc.

- In order to use previous predictions of NC classes, you have to run the system twice and use the output of the first run as features for the second run, e.g., $previous\_NC=B-NP$ is a feature you can derive by running the system once and then rerunning based on the output of the first pass.
• We will be using the opennlp Maxent package
  – If you are using JAVA, Download the package from:
    • http://sourceforge.net/projects/maxent/files/latest/download?source=files
  – There are (at least) 2 ways you can use it:
    • If you are coding it in JAVA, you can import it
      – See Ralph's instructions for a different NLP assignment:
        http://cs.nyu.edu/courses/spring15/CSCI-GA.2590-001/asgn6.html
    • Otherwise, you can use the wrapper written by Ang Sun (an NYU PhD from several years ago):
      – http://cs.nyu.edu/courses/spring12/CSCI-GA.2590-001/MaxentWrapper.zip
      – The README in the zip file provide instructions
• Regardless of which programming language you use, please read the following information about the Maxent package you are using as it may be helpful for your term project
  – http://maxent.sourceforge.net/about.html
Homework 6 – Slide 6

• Create a MEMM Noun Group Chunker
  – Your system should train on the data in train.np
    • Record features that are likely to predict NG boundaries
    • Use the Maxent package
  – You should test your system on the data in dev.np
    • Identify these features in your development corpus
    • Use Maxent package and Viterbi decoder (reuse your HW #4 code) to predict the B-NP, I-NP, O tags in the development corpus
  – Miao will test it on some test data that we have set aside.
• Format of the training file for the maxent package
  – One line per word: $feature_1=value_1, feature_2=value_2 \ldots feature_N=value_N, NC$
  • NC is a member of {B-NP,I-NP,O}
• You should choose experiment with features you think will work, e.g.,
  – Features of current word: the word itself, POS, stemmed version of word, etc.
  – Features of previous word, Features of two words back, Features of following word
  – Other: capitalization, features of the sentence, your own special dictionary, etc.
• The Maxent package will provide:
  – For each word in the corpus you are testing, probabilities that the word should be classified as B-NP, I-NP, or O (and these probabilities should sum to 1)
• When you are satisfied with how your system runs against the development corpus, you should submit the following:
  – Your program, along with instructions to run it and a list of the features that you are using.