Named Entities, Named Entity Tagging and Machine Learning

Adam Meyers
New York University
10/23/2011 Draft
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/ 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: $q_0$ (start state), ..., $q_F$ (final state)
    - $q_0$ and $q_F$ 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 $\sum_{j=1}^{n} a_{i,j} = 1 \forall i$
  - $\sum_{j=1}^{n} a_{i,j} = 1 \forall i$

- 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 in 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

\[
Pr(NC|W) = \frac{Pr(W|NC)}{Pr(W)} \times Pr(NC) \quad \# \text{Bayes Rule}
\]

\[
Pr(W|NC) \approx Pr(W|NC) \times Pr(NC) \quad \# \text{Pr(W) ignored (same for any tag seqs)}
\]

\[
Pr(NC) = \text{Prior}
\]

- 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: \[ Pr(NC|NC_{-1}, w_{-1}) \times Pr(<w, f>_{\text{first}}|NC, NC_{-1}) \]
  
  \begin{itemize}
  \item \( Pr(NC|NC_{-1}, w_{-1}) \)  
    \begin{itemize}
    \item \# e.g., Mr. precedes B-PER
  \item \( Pr(<w, f>_{\text{first}}|NC, NC_{-1}) \)  
    \begin{itemize}
    \item \# e.g., O precedes capitalized B-PER
  \end{itemize}
  \end{itemize}

• Prior for subsequent words of NC:
  
  \begin{itemize}
  \item \( Pr(<w, f>|<w, f>_{-1}, NC) \)  
    \begin{itemize}
    \item \# sequences of same/diff class
  \end{itemize}
  \end{itemize}

• Probability that the current word ends an NC:
  
  \begin{itemize}
  \item \( Pr(<+ \text{ end }+, \text{ other }>|<w, f>_{\text{final}}, NC) \)
  \end{itemize}
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

\[ \Pr(<w,f>_{\text{first}}|NC, NC_{-1}) \subset \Pr(<w,f>|<+ \text{ begin}+, \text{ other}>NC) \subset \]
\[ \Pr(<w,f>|NC) \subset \ldots \Pr(w|NC) \times \Pr(f|NC) \subset \frac{1}{\text{vocab size}} \times \frac{1}{\text{number features}} \]

• 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: \( M_1 \subset M_2 \subset M_3 \subset M_4 \ldots \)

• Models M and M' with weights \( \Lambda \) and \( 1 - \Lambda \)
  – This is called smoothing

• \( \Lambda \) based on relative sample sizes of M and M'
  – In model M, \( \text{Pr}(X|Y) \) is based on the count of Y
  – In model M', \( \text{Pr}(X|Y') \) is based on the count of Y'
  – \( c(Y') > c(Y) \) e.g., suppose \( Y = \text{NC}_{-1}, w_{-1} \) and \( Y' = \text{NC}_{-1} \)
    – Assume \( Y' \) has N unique outcomes, e.g., \( Y' \) is 8 in above example if 8 different NCs can precede w

• \( \Lambda \) favors more frequent Y and more diverse Y'

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

- Unsupervised Methods: no annotation
- Weakly Supervised Methods: less detailed annotation or very little annotation
- Active Learning:
  - Only manually annotate outliers
- Other “less expensive” methods
  - Less annotation
  - Domain adaptation
  - Etc.
- These methods require that we find ways to estimate probabilities rather than always “learning” them from annotated corpora.
Estimating Likelihoods

- Probability of sequence of n states Q and sequence of n hidden events O: likelihoods X prior prob
  - Likelihoods = \( \prod_{i=1}^{n} P(O_i|q_i) \)
  - Prior Prob = \( \prod_{i=1}^{n} P(q_i|q_{i-1}) \)
  - Formula: \( P(O,Q)=\prod_{i=1}^{n} P(O_i|q_i) \times \prod_{i=1}^{n} P(q_i|q_{i-1}) \)

- Total probability of observation sequence:
  - Product of probabilities of a sequence of observations given all sequences of states: \( P(O)=\prod_{Q} P(O|Q)P(Q) \)
  - Given T observations there are \( N^T \) possible paths of states

- Problem: How do we efficiently compute this over \( N^T \) paths?
- Solution: Use Forward Algorithm \( O(N^2T) \) rather than \( O(N^T) \)
  - Uses dynamic programming (stores partial results in look up table to prevent duplicate calculations)
Forward Algorithm (Like Viterbi w/o Backpointers)

- **Factors**
  - $\text{forward}_{t-1}(i) = \text{the previous forward path probability}$
  - $a_{ij} = \text{transition probability (prior)}$
  - $b_j(o_t) = \text{likelihood of observation } o_t \text{ given (current) state } j$

- **Algorithm**
  Create $\text{forward}$, a $[N+2,T]$ prob matrix  # N states+initial+final
  
  for state $s$ from 1 to $N$ do
  
  \[
  \text{forward}[s,1] \leftarrow a_{0,s} \times b_s(o_1) \quad \# \text{Transitions from initial state}
  \]

  for time $t$ from 2 to $T$ do
  
  for each state $s$ from 1 to $N$ do
  
  \[
  \text{forward}[s,t] \leftarrow \sum_{s'=1}^{N} \text{forward}[s',t-1] \times a_{s',s} \times b_s(o_t) \quad \# \text{Set cells recursively}
  \]

  \[
  \text{forward}[qF,T] \leftarrow \sum_{s=1}^{N} \text{forward}[s,T] \times a_s,qF \quad \# \text{Transitions to final state}
  \]

- $\text{forward}[qF,T] = \text{probability of observed sequence}$
The Backwards Algorithm (~Forward from F to 0)

- **Purpose:** find prob of observations from time+1 to F, given time
- **Algorithm**

  Create $\text{backward}$, a $[N+2,T]$ prob matrix  
  # assume time = 0

  for state $s$ from 1 to $N$ do
  
    $\text{backward}[s] \leftarrow a_{s,F}$  
    # Transitions to Final State

  for time $t$ from $T-1$ down to 1 do
  
    for each state $i$ from $N$ down to 1 do
    
      $\text{backward}_t(i) \leftarrow \sum_{j=1}^{N} a_{i,j} \times b_j(O_{t+1}) \times \text{backward}_{t+1}(j)$  
      \hspace{1cm} # Set cells recursively

  $\text{backward}(0) = \sum_{j=1}^{N} a_{0,j} \times b_j(o_1) \times \text{backward}(j)$  
  \hspace{1cm} # Transitions from initial state

  return backward(0)
Forward-Backward Algorithm I

• Purpose: To “Learn” Parameters for HMM
  – Learning Prior Probabilities (transition)
  – Learning Likelihoods (emission)
• Transition Probability: \( a_{i,j} = \frac{C(i \rightarrow j)}{\sum_{q \in Q} C(i \rightarrow q)} \)
  – Can't calculate it – we can only estimate
• Smoothing is an estimation procedure (backoff, FB, ...) that attempts to capture important patterns in data
• Given any arbitrary point \( t \), find probabilities:
  – going from \( q_0 \) to \( q_t \) using the **forward** algorithm
  – going from \( q_t \) to \( q_F \) using the **backward** algorithm
• Then combine the forward and backward probabilities
Forward-Backward II

• Initialize A (transition probabilities) and B (likelihoods)
  – Ex: P(outgoing_edge(X)) = 1/(degree(X)); P(each_word) = 1/vocab_size
  • Assumes small training corpus (not always the case)

• Repeat the following 2 steps until the results converge
  – (Re)estimate counts based on A and B (Expectation Step)
    • $\gamma_t(j) = \frac{\text{forward}(j) \times \text{backward}(j)}{P(\text{Observed \_Sequence} | \text{Current \_Model})} \quad \forall t \land j$  # implied double loop
    • $\xi_t(i, j) = \frac{\text{forward}(i) \times a_{i,j} \times b_j(O_{t+1}) \times \text{backward}_{t+1}(j)}{\text{forward}_j(N)} \quad \forall t, i, \land j$  # implied triple loop
      # denominator = prob(utterance)
      # numerator = forward \* backward \* prob(i→j) \* likelihood(t+1)
  – (Re)calculate A and B based on (new) counts (Maximization Step)
    $\hat{a}_{i,j} = \sum_{t=1}^{T-1} \xi_t(i, j) \quad \# \text{implied loops (for i, j and z)}$
    $\# \hat{a}_{i,j} = (\text{expected } i \rightarrow j) / (\text{expected } i \rightarrow \text{anything})$
    $\hat{b}_j(V_k) = \frac{\sum_{i=1}^{T} \gamma_{t}(j)}{\sum_{i=1}^{T} \gamma_{t}(j)} \quad \# \hat{b}_j(V_k) = (\text{expected } V_k \text{ in state } j) / (\text{expected state } j)$

• Return A (priors, i.e., all $\hat{a}_{i,j}$) and B (likelihoods, all $\hat{b}_j(V_k)$)
Lots of Evidence

• 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
• Maximum Entropy is one such method
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

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

• \( c \) = class, \( x \) = observation, \( Z \) normalizing factor, \( w_i \) and \( f_i \) are features and weights (both depending on \( c \))
• \( Z \) makes all probabilities sum to 1
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 \( y_{obs}^{(j)} \)
  - \( y_{pred}^{(j)} = \sum_{i=0}^{M} w_i \times f_i^{(j)} \)
  - Cost: \( \text{cost}(W) = \sum_{j=0}^{M} (y_{pred}^{(j)} - y_{obs}^{(j)})^2 \)
  - 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
  - \( W = (M^T \cdot M)^{-1} \cdot M^T \cdot \vec{v} \)
Logistic Regression

• If we assume binary values (true|false or 1|0)
  \[ p(y=true|x) = \frac{e^{w \cdot f}}{1 + e^{w \cdot f}} \quad \text{and} \quad p(y=false|x) = \frac{e^{-w \cdot f}}{1 + e^{-w \cdot f}} \]

  – 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:
  \[ p(\text{true}|x) > p(\text{false}|x) \]
  \[ \text{Or if} \quad w \cdot f = \ln\left(\frac{p(\text{true})}{p(\text{false})}\right) > 0 \]

  – 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

- Multinominal 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 \( C = \{c_1, \ldots, c_C\} \)

\[
p(c|x) = \frac{1}{Z} \times e^{\sum_i w_i f_i} \\
Z = \sum_{c' \in C} e^{\sum_{i=0}^{N} w_{c'f_i}}
\]

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

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

- 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 $W$
  $$\hat{T} = \arg\max_T P(T|W)$$

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

- 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: Baysean networks, Support Vector Machines, Perceptron, ...
Readings

- J & M Chapter 6
- NLTK – Read and Try examples from section 7.5
- ACE Named Entity Specifications
  - Read First 3 sections
Homework #5 – Slide 1

• Download the following file:
  – http://cs.nyu.edu/courses/spring12/CSCI-GA.2590-001/NounGroupsforNLPClass.zip

• Unpack this zip file will produce the following files:
  – conlleval, Scorer_Instructions, sample.test
    • Evaluation software, instructions, and a sample output file
  – dev.np and train.np
    • dev and training files using the same corpus as the POS tagging exercise in Homework #4
    • These are the .pos files, plus an additional field with one of these symbols:
      – B-NP – Beginning of Base NP (Noun Group)
      – I-NP – Inside of Base NP (Noun Group)
      – O – Outside of Base NP
Homework # 5 – Slide 2

• 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/spring10/G22.2590-001/asgn8.html
        » Use the links I provide for opennlp, rather than Ralph's, which
          no longer work
    • Otherwise, you can use the wrapper that Ang wrote:
      – 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 # 5 – Slide 3

• 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
  – Ang will test it on some test data that we have set aside.
Homework # 5 – Slide 4

• Format of the training file for the maxent package
  – One line per word: \textit{feature}_1 = \textit{value}_1 \textit{feature}_2 = \textit{value}_2 \ldots \textit{feature}_N = \textit{value}_N \textit{NC}
  • \textit{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 for Ang to run it
  – A pdf file "\text{FirstName}_\text{LastName}_\text{HW5Features}.pdf" briefly describes the features that you chose to use.