Lecture 6:
Part-of-speech tagging

Homework assignments

Schedule:
Week 1: Tue, 08/25 HW0 out
Week 3: Thu, 09/10 HW0 due, HW1 out (TODAY)
Week 6: Thu, 10/01 HW1 due, HW2 out
Week 9: Thu, 10/22 HW2 due, HW3 out
Week 12: Thu, 11/12 HW3 due, HW4 out
Week 15: Tue, 12/08 HW4 due (last lecture)

Points per assignment:
HW0 = 2 points
(Did you submit (on time)? Was it in the right format?)
HW1,HW2,HW3,HW4 = 10 points per assignment

HW0 is due at 10pm today.
HW1 will go out this evening.
HW1 is due at 10pm on Thursday, Oct 1.
We use Compass.

Email us ASAP if you cannot access the Compass page for our class.
4th credit hour

Two choices:
- Research project (alone, or with one other student)
- Literature survey (alone)

Deadlines:
- Before Oct 6: Check your idea with me
- Oct 6 (Wk 6): Proposal due
  (What topic? What papers will you read?)
- Nov 12 (Wk 12): Progress report due
  (Are your experiments on track? Is your paper on track?)
- Dec 10 (Reading Day): Final report due
  (Summary of papers, your system)

4th credit hour: Research Projects

What?
You need to read and describe a few (2-3) NLP papers on a particular task, implement an NLP system for this task and describe it in a written report.

Why?
To make sure you get a deeper knowledge of NLP by reading original papers and by building an actual system.

When?
- Oct 6 (Wk 6): Proposal due (What topic? What papers will you read?)
- Nov 12 (Wk 12): Progress report due (Are your experiments on track?)
- Dec 10 (Reading Day): Final report due (Summary of papers, your system)

4th credit hour: Literature Survey

What?
You need to read and describe several (5-7) NLP papers on a particular task or topic, and produce a written report that compares and critiques these approaches.

Why?
To make sure you get a deeper knowledge of NLP by reading original papers, even if you don’t build an actual system.

When?
- Oct 6 (Wk 6): Proposal due (What topic? What papers will you read?)
- Nov 12 (Wk 12): Progress report due (Is your paper on track?)
- Dec 10 (Reading Day): Final report due (Summary of papers, your system)

Tuesday’s key concepts

Smoothing:
- Good Turing,
- Absolute Discounting,
- Kneser Ney

Intrinsic evaluation of language models:
- Perplexity

Extrinsic evaluation of language models:
- Task-based evaluation
- Word Error Rate
Part-of-speech (POS) tagging

Why POS tagging?

POS tagging is a prerequisite for further analysis:

—Speech synthesis:
  How to pronounce "lead"?
  Insult or inSULT, OBJect or obJECT, OVERflow or overFLOW,
  DIScount or disCOUNT, CONtent or conTENT

—Parsing:
  What words are in the sentence?

—Information extraction:
  Finding names, relations, etc.

—Machine Translation:
  The noun "content" may have a different translation from the adjective.

POS Tagging

Words often have more than one POS:

- The **back** door (adjective)
- On my **back** (noun)
- Win the voters **back** (particle)
- Promised to **back** the bill (verb)

The POS tagging task is to determine the POS tag for a particular instance of a word.

Since there is ambiguity, we cannot simply look up the correct POS in a dictionary.

These examples from Dekang Lin
Defining a tagset

Word classes

Open classes:
- Nouns, Verbs, Adjectives, Adverbs

Closed classes:
- Auxiliaries and modal verbs
- Prepositions, Conjunctions
- Pronouns, Determiners
- Particles, Numerals

(see Appendix for details)

Tagsets for English

We have to agree on a standard inventory of word classes.

Most taggers rely on statistical models; therefore the tagsets used in large corpora become de facto standard.

Tagsets need to capture semantically or syntactically important distinctions that can easily be made by trained human annotators.
How much ambiguity is there?

How many tags does each word type have?
(Original Brown corpus: 40% of tokens are ambiguous)

<table>
<thead>
<tr>
<th></th>
<th>87-tag Original Brown</th>
<th>45-tag Treebank Brown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unambiguous (1 tag)</td>
<td>44,019</td>
<td>38,857</td>
</tr>
<tr>
<td>Ambiguous (2–7 tags)</td>
<td>5,490</td>
<td>8,844</td>
</tr>
<tr>
<td>Details:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 tags</td>
<td>4,967</td>
<td>6,731</td>
</tr>
<tr>
<td>3 tags</td>
<td>411</td>
<td>1,621</td>
</tr>
<tr>
<td>4 tags</td>
<td>91</td>
<td>357</td>
</tr>
<tr>
<td>5 tags</td>
<td>17</td>
<td>90</td>
</tr>
<tr>
<td>6 tags</td>
<td>2 (well, beat)</td>
<td>32</td>
</tr>
<tr>
<td>7 tags</td>
<td>2 (still, down)</td>
<td>6 (well, set, round, open, fit, down)</td>
</tr>
<tr>
<td>8 tags</td>
<td>4 (‘s, half, back, a)</td>
<td></td>
</tr>
<tr>
<td>9 tags</td>
<td>3 (that, more, in)</td>
<td></td>
</tr>
</tbody>
</table>

NB: These are just the tags words appeared with in the corpus. There may be unseen word/tag combinations.

Evaluation metric: accuracy

How many words in the unseen test data can you tag correctly?
State of the art on Penn Treebank: around 97%.

Compare your model against a baseline
Standard: assign to each word its most likely tag (use training corpus to estimate P(tw))
Baseline performance on Penn Treebank: around 93.7%

... and a (human) ceiling
How often do human annotators agree on the same tag? Penn Treebank: around 97%
Qualitative evaluation

Generate a **confusion matrix** (for development data):
How often was tag i mistagged as tag j:

<table>
<thead>
<tr>
<th></th>
<th>IN</th>
<th>JJ</th>
<th>NN</th>
<th>NNP</th>
<th>RB</th>
<th>VBD</th>
<th>VBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>.2</td>
<td>.2</td>
<td>.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJ</td>
<td>.2</td>
<td>3.3</td>
<td>2.1</td>
<td>1.7</td>
<td>.2</td>
<td>2.7</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>8.7</td>
<td>.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NNP</td>
<td>.2</td>
<td>3.3</td>
<td>.4</td>
<td>.1</td>
<td>.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB</td>
<td>2.2</td>
<td>2.0</td>
<td>.5</td>
<td>.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBD</td>
<td>.3</td>
<td>.5</td>
<td>.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBN</td>
<td>2.8</td>
<td>2.6</td>
<td>.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See what errors are causing problems:
- Noun (NN) vs ProperNoun (NNP) vs Adj (JJ)
- Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)

Is POS-tagging a solved task?

Penn Treebank POS-tagging accuracy
\[ \approx \text{human ceiling} \]

Yes, but:
- Other languages with more complex morphology need much larger tagsets for tagging to be useful, and will contain many more distinct word forms in corpora of the same size
  - They often have much lower accuracies

Building a POS tagger

Statistical POS tagging

\[
\begin{align*}
\text{she}_1 & \quad \text{promised}_2 & \quad \text{to}_3 & \quad \text{back}_4 & \quad \text{the}_5 & \quad \text{bill}_6 \\
\mathbf{w} &= [w_1, \quad w_2, \quad w_3, \quad w_4, \quad w_5, \quad w_6] \\
\mathbf{t} &= [t_1, \quad t_2, \quad t_3, \quad t_4, \quad t_5, \quad t_6] \\
&\quad \text{PRP}_1, \quad \text{VBD}_2, \quad \text{TO}_3, \quad \text{VB}_4, \quad \text{DT}_5, \quad \text{NN}_6
\end{align*}
\]

What is the most likely sequence of tags \( \mathbf{t} \) for the given sequence of words \( \mathbf{w} \) ?
Statistical POS tagging

What is the most likely sequence of tags \( t \) for the given sequence of words \( w \)?

\[
\begin{align*}
\arg\max_t P(t|w) &= \arg\max_t \frac{P(t,w)}{P(w)} \\
&= \arg\max_t P(t,w) \\
&= \arg\max_t P(t)P(w|t)
\end{align*}
\]

\( P(t,w) \) is a generative (joint) model.

Hidden Markov Models are generative models which decompose \( P(t,w) \) as \( P(t)P(w|t) \)

\[\]

\( P(w|t) \): Generating \( w=w_1…w_n \)

We assume that words are independent of each other, and depend only on their own POS-tag:

\[
P(w_1w_2…w_{n-1}w_n|t_1t_2…t_{n-1}t_n) = \prod_i P(w_i|w_{i-1},w_{i-1},t_{i-1}…t_{i-1}t_n) \\
:= def \prod_i P(w_i|t_i)
\]

\[\]

\( P(t): Generating \ t=t_1…t_n \)

We make the same Markov assumption as in language modeling:

\[
P(t_1t_2…t_{n-1}t_n) = P(t_1)P(t_2|t_1)…P(t_n|t_1…t_{n-1}) = \prod_{i=1}^n P(t_i|t_1…t_{i-1}) \\
:= def \prod_{i=1}^n P(t_i|t_{i-1}…t_{i-1+n})
\]

We define an n-gram model over POS tags

\[\]

Hidden Markov Models

HMM models are generative models of \( P(w,t) \) (because they model \( P(w|t) \) rather than \( P(t|w) \))

They make two independence assumptions:

a) approximate \( P(t) \) with an N-gram model

b) assume that each word depends only on its POS tag

\[\]
An example HMM

Transition Matrix $A$

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>N</th>
<th>V</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>0.8</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0.7</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>0.6</td>
<td></td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0.8</td>
<td></td>
<td></td>
<td>0.2</td>
</tr>
</tbody>
</table>

Emission Matrix $B$

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>man</th>
<th>ball</th>
<th>throws</th>
<th>sees</th>
<th>red</th>
<th>blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0.7</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
</tr>
</tbody>
</table>

Initial state vector $\pi$

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>N</th>
<th>V</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HMMs as probabilistic automata

An HMM defines

Transition probabilities:

$P( t_i | t_{i-1} )$

- Emission probabilities:

$P( w_i | t_i )$

Using HMMs for tagging

- The input to an HMM tagger is a sequence of words, $w$.
  The output is the most likely sequence of tags, $t$, for $w$.

- For the underlying HMM model, $w$ is a sequence of output symbols, and $t$ is the most likely sequence of states (in the Markov chain) that generated $w$.

$$
\arg\max_t P( t | w ) = \arg\max_t \frac{P(w, t)}{P(w)} = \arg\max_t P(w, t) = \arg\max_t P( w | t ) P( t | States_{HMM} ) \frac{t | States_{HMM}}{P(t | States_{HMM})}
$$

How would the automaton for a trigram HMM with transition probabilities $P( t_i | t_{i-2}, t_{i-1} )$ look like?

What about unigrams or n-grams?
Learning an HMM

Where do we get the transition probabilities \( P(t_j | t_i) \) (matrix \( A \)) and the emission probabilities \( P(w_j | t_i) \) (matrix \( B \)) from?

Case 1: We have a POS-tagged corpus.
- This is learning from labeled data, aka “supervised learning”

Case 2: We have a raw (untagged) corpus and a tagset.
- This is learning from unlabeled data, aka “unsupervised learning”

HMM definition

A HMM \( \lambda = (A, B, \pi) \) consists of

- a set of \( N \) states \( Q = \{q_1, \ldots, q_N\} \) with \( Q_0 \subseteq Q \) a set of initial states and \( Q_F \subseteq Q \) a set of final (accepting) states
- an output vocabulary of \( M \) items \( V = \{v_1, \ldots, v_M\} \)
- an \( N \times N \) state transition probability matrix \( A \) with \( a_{ij} \) the probability of moving from \( q_i \) to \( q_j \).
  \( \sum_{j=1}^{N} a_{ij} = 1 \quad \forall i \)
  \( 0 \leq a_{ij} \leq 1 \quad \forall i, j \)
- an \( N \times M \) symbol emission probability matrix \( B \) with \( b_{ij} \) the probability of emitting symbol \( v_j \) in state \( q_i \)
  \( \sum_{j=1}^{M} b_{ij} = 1 \quad \forall i \)
  \( 0 \leq b_{ij} \leq 1 \quad \forall i, j \)
- an initial state distribution vector \( \pi = (\pi_1, \ldots, \pi_N) \) with \( \pi_i \) the probability of being in state \( q_i \) at time \( t = 1 \).
  \( \sum_{i=1}^{N} \pi_i = 1 \quad 0 \leq \pi_i \leq 1 \quad \forall i \)

We count how often we see \( t_j \) and \( w_j, t_i \) etc. in the data (use relative frequency estimates):

Learning the transition probabilities:

\[
P(t_j | t_i) = \frac{C(t_j, t_i)}{C(t_i)}
\]

Learning the emission probabilities:

\[
P(w_j | t_i) = \frac{C(w_j, t_i)}{C(t_i)}
\]

We might use some smoothing, but this is pretty trivial...
Outlook: Dynamic Programming for HMMs

The three basic problems for HMMs

We observe an output sequence \( w = w_1 \ldots w_N \):
\( w = \text{"she promised to back the bill"} \)

**Problem I (Likelihood):** find \( P(w | \lambda) \)
Given an HMM \( \lambda = (A, B, \pi) \), compute the likelihood of the observed output, \( P(w | \lambda) \)

**Problem II (Decoding):** find \( Q = q_1 \ldots q_T \)
Given an HMM \( \lambda = (A, B, \pi) \), what is the most likely sequence of states \( Q = q_1 \ldots q_N \approx t_1 \ldots t_N \) to generate \( w \)?

**Problem III (Estimation):** find \( \arg\max \lambda P(w | \lambda) \)
Find the parameters \( A, B, \pi \) which maximize \( P(w | \lambda) \)

How can we solve these problems?

I. **Likelihood** of the input:
Compute \( P(w | \lambda) \) for the input \( w \) and HMM \( \lambda \)

II. **Decoding (= tagging) the input:**
Find the best tags \( t^* = \arg\max_t P(t | w, \lambda) \) for the input \( w \) and HMM \( \lambda \)

III. **Estimation (= learning the model):**
Find the best model parameters \( \lambda^* = \arg\max_\lambda P(t, w | \lambda) \)
for the training data \( w \)

These look like hard problems: With \( T \) tags, every input string \( w_1 \ldots n \) has \( T^n \) possible tag sequences

Can we find efficient (polynomial-time) algorithms?

Dynamic programming

We will use a general technique called dynamic programming to solve these problems.

- We will recursively decompose each of these problems into smaller subproblems that can be solved efficiently
- There is only a polynomial number of subproblems.
- We will store the solution of each subproblem in a common data structure
- Processing this data structure takes polynomial time
Solution: Dynamic programming

I. Likelihood of the input:
Compute $P(w | \lambda)$ for the input $w$ and HMM $\lambda$
$\Rightarrow$ Forward algorithm

II. Decoding (=tagging) the input:
Find best tags $t^* = \text{argmax}_t P(t | w, \lambda)$ for the input $w$ and HMM $\lambda$
$\Rightarrow$ Viterbi algorithm

III. Estimation (=learning the model):
Find best model parameters $\lambda^* = \text{argmax}_\lambda P(t, w | \lambda)$ for training data $w$
$\Rightarrow$ Forward-Backward algorithm

Appendix: English parts of speech
Nouns

Nouns describe entities and concepts:
Common nouns: dog, bandwidth, dog, fire, snow, information
- Count nouns have a plural (dogs) and need an article in the singular (the dog barks)
- Mass nouns don’t have a plural (*snows) and don’t need an article in the singular (snow is cold, metal is expensive). But some mass nouns can also be used as count nouns: Gold and silver are metals.

Proper nouns (Names): Mary, Smith, Illinois, USA, France, IBM

Penn Treebank tags:
NN: singular or mass
NNS: plural
NNP: singular proper noun
NNPS: plural proper noun

Adjectives

Adjectives describe properties of entities: blue, hot, old, smelly,…

Adjectives have an...
… attributive use (modifying a noun):
  the blue book
… and a predicative use (e.g. as argument of be):
  The book is blue.

Many gradable adjectives also have a...
…comparative form: greater, hotter, better, worse
…superlative form: greatest, hottest, best, worst

Penn Treebank tags:
JJ: adjective    JJR: comparative     JJS: superlative

(Verbs

Verbs describe activities, processes, events:
eat, write, sleep, ….  
Verbs have different morphological forms:
infinite (to eat), present tense (I eat), 3rd pers sg. present tense (he eats),
past tense (ate), present participle (eating), past participle (eaten)

Penn Treebank tags:
VB: infinitive (base) form
VBD: past tense
VBG: present participle
VBD: past tense
VBN: past participle
VBZ: 3rd person singular present tense

Adverbs

Adverbs describe properties of events/states.
- Manner adverbs: slowly (slower, slowest) fast, hesitantly,…
- Degree adverbs: extremely, very, highly,…
- Directional and locative adverbs: here, downstairs, left
- Temporal adverbs: yesterday, Monday,…

Adverbs modify verbs, sentences, adjectives or other adverbs:
  Apparently, the very ill man walks extremely slowly

NB: certain temporal and locative adverbs (yesterday, here)
can also be classified as nouns

Penn Treebank tags:
RB: adverb    RBR: comparative adverb    RBS: superlative adverb
Auxiliary and modal verbs

Copula: be with a predicate
She is a student. I am hungry. She was five years old.

Modal verbs: can, may, must, might, shall,…
She can swim. You must come

Auxiliary verbs:
- Be, have, will when used to form complex tenses:
  He was being followed. She has seen him. We will have been gone.
- Do in questions, negation:
  Don’t go. Did you see him?

Penn Treebank tags:
MD: modal verbs

Conjunctions

Coordinating conjunctions conjoin two elements:
X and/or/but X
[ [John]NP and [Mary]NP] NP,
[ [Snow is cold]S but [fire is hot]S ]S.

Subordinating conjunctions introduce a subordinate
(embedded) clause:
[ He thinks that [snow is cold]S ]S
[ She wonders whether [it is cold outside]S ]S

Penn Treebank tags:
CC: coordinating
IN: subordinating (same as preposition)

Prepositions

Prepositions occur before noun phrases to form a prepositional phrase (PP):
on/in/under/near/towards the wall,
with(out) milk,
by the author,
despite your protest

PPs can modify nouns, verbs or sentences:
I drink [coffee [with milk]]
I [drink coffee [with my friends]]

Penn Treebank tags:
IN: preposition
TO: ‘to’ (infinitival ‘to eat’ and preposition ‘to you’)

Particles

Particles resemble prepositions (but are not followed by a noun phrase) and appear with verbs:

come on
he brushed himself off
turning the paper over
turning the paper down

Phrasal verb: a verb + particle combination that has a different meaning from the verb itself

Penn Treebank tags:
RP: particle
Pronouns

Many pronouns function like noun phrases, and refer to some other entity:
- Personal pronouns: I, you, he, she, it, we, they
- Possessive pronouns: mine, yours, hers, ours
- Demonstrative pronouns: this, that,
- Reflexive pronouns: myself, himself, ourselves
- Wh-pronouns (question words):
  what, who, whom, how, why, whoever, which

Relative pronouns introduce relative clauses
  the book that [he wrote]

Penn Treebank tags:
- PRP: personal pronoun
- PRP$: possessive
- WP: wh-pronoun

Determiners

Determiners precede noun phrases:
- Articles: the, an, a
- Demonstratives: this, these, that
- Quantifiers: some, every, few,…

Penn Treebank tags:
- DT: determiner