Why POS tagging?

POS tagging is a prerequisite for further analysis:

- Speech synthesis:
  How to pronounce “lead”?
  Intersect 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.
Defining a tag set

We have to define an inventory of labels for the word classes (i.e. the tag set)

- Most taggers rely on models that have to be trained on annotated (tagged) corpora. Evaluation also requires annotated corpora.
- Since human annotation is expensive/time-consuming, the tag sets used in a few existing labeled corpora become the de facto standard.
- Tag sets need to capture semantically or syntactically important distinctions that can easily be made by trained human annotators.

Defining an annotation scheme

A lot of NLP tasks require systems to map natural language text to another representation:

- POS tagging: Text → POS tagged text
- Syntactic Parsing: Text → parse trees
- Semantic Parsing: Text → meaning representations
- …: Text → …

Training and evaluating models for these NLP tasks requires large corpora annotated with the desired representations.

Annotation at scale is expensive, so a few existing corpora and their annotations and annotation schemes (tag sets, etc.) often become the de facto standard for the field.

It is difficult to know what the ‘right' annotation scheme should be for any particular task

How difficult is it to achieve high accuracy for that annotation?
How useful is this annotation scheme for downstream tasks in the pipeline?
⇒ We often can’t know the answer until we’ve annotated a lot of data…
Word classes

Open classes:
- Nouns, Verbs, Adjectives, Adverbs

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

(see Appendix for details)

Defining a tag set

Tag sets have different granularities:
- Brown corpus (Francis and Kucera 1982): 87 tags
- Penn Treebank (Marcus et al. 1993): 45 tags
- Simplified version of Brown tag set (de facto standard for English now)
- NN: common noun (singular or mass): water, book
- NNS: common noun (plural): books

Prague Dependency Treebank (Czech): 4452 tags
- Complete morphological analysis:
  - AAFP3----3N----: nejnezajímavější
  - Adjective Regular Feminine Plural Dative…Superlative
  - [Hajic 2006, VMC tutorial]

How much ambiguity is there?

Most word types are unambiguous:

Number of tags per word type:

<table>
<thead>
<tr>
<th></th>
<th>87-tag Original Brown</th>
<th>45-tag Treebank Brown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unambiguous</td>
<td>44,019</td>
<td>38,857</td>
</tr>
<tr>
<td>Ambiguous</td>
<td>5,490</td>
<td>8844</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 (t, half, back, a)</td>
<td>3 (that, more, in)</td>
</tr>
<tr>
<td>9 tags</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NB: These numbers are based on word/tag combinations in the corpus. Many combinations that don’t occur in the corpus are equally correct.

But a large fraction of word tokens are ambiguous

Original Brown corpus: 40% of tokens are ambiguous

Evaluating POS taggers
Evaluating POS taggers

Evaluation setup:
Split data into separate training, (development) and test sets.

Better setup: n-fold cross validation:
Split data into \( n \) sets of equal size
Run \( n \) experiments, using set \( i \) to test and remainder to train

This gives average, maximal and minimal accuracies

When comparing two taggers:
Use the same test and training data with the same tag set

Evaluation metric: test accuracy

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

Compare your model against a baseline
Standard: assign to each word its most likely tag
(use training corpus to estimate \( P(t|w) \) )
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 a word with tag \( i \) mistagged as tag \( j \):

<table>
<thead>
<tr>
<th>Correct Tags</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>3.3</td>
<td>4.1</td>
<td>2.2</td>
<td>2.8</td>
<td>2.6</td>
</tr>
<tr>
<td>JJ</td>
<td>2</td>
<td>2</td>
<td>2.2</td>
<td>2.0</td>
<td>2.2</td>
<td>3.5</td>
<td>3.5</td>
</tr>
<tr>
<td>NN</td>
<td>3.3</td>
<td>2.2</td>
<td>3.3</td>
<td>4.1</td>
<td>2.2</td>
<td>2.8</td>
<td>2.6</td>
</tr>
<tr>
<td>NNP</td>
<td>2.1</td>
<td>2.8</td>
<td>4.4</td>
<td>2.6</td>
<td>3.5</td>
<td>2.8</td>
<td>2.8</td>
</tr>
<tr>
<td>RB</td>
<td>1.7</td>
<td>1.7</td>
<td>2.6</td>
<td>2.6</td>
<td>4.4</td>
<td>3.5</td>
<td>3.5</td>
</tr>
<tr>
<td>VBD</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3.5</td>
<td>3.5</td>
</tr>
<tr>
<td>VBN</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3.5</td>
</tr>
</tbody>
</table>

% of errors caused by mistagging VBN as JJ

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

Is POS-tagging a solved task?

Penn Treebank POS-tagging accuracy
\( \approx \) human ceiling

Yes, but:
Other languages with more complex morphology
need much larger tag sets 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

POS tagging with generative models

$$\text{argmax}_t P(t|w) = \text{argmax}_t \frac{P(t,w)}{P(w)}$$

$$= \text{argmax}_t P(t,w)$$

$$= \text{argmax}_t P(t)P(w|t)$$

$P(t,w)$: the joint distribution of the labels we want to predict ($t$) and the observed data ($w$).

We decompose $P(t,w)$ into $P(t)$ and $P(w|t)$ since these distributions are easier to estimate.

Models based on joint distributions of labels and observed data are called generative models: think of $P(t)P(w|t)$ as a stochastic process that first generates the labels, and then generates the data we see, based on these labels.

Statistical POS tagging

```
She promised to back the bill
```

What is the most likely sequence of tags $t = t^{(1)} \ldots t^{(N)}$ for the given sequence of words $w = w^{(1)} \ldots w^{(N)}$?

$$t^* = \text{argmax}_t P(t | w)$$

Hidden Markov Models (HMMs)

HMMs are the most commonly used generative models for POS tagging (and other tasks, e.g. in speech recognition).

HMMs make specific independence assumptions when defining $P(t)$ and $P(w | t)$:

- $P(t)$ is an n-gram model over tags:
  - Bigram HMM: $P(t) = P(t^{(1)})P(t^{(2)} | t^{(1)})P(t^{(3)} | t^{(2)}) \ldots P(t^{(N)} | t^{(N-1)})$
  - Trigram HMM: $P(t) = P(t^{(1)})P(t^{(2)} | t^{(1)})P(t^{(3)} | t^{(2)}, t^{(1)}) \ldots P(t^{(N)} | t^{(N-1)}, t^{(N-2)})$

- $P(t)$ or $P(t_i \mid t_j, t_k)$ are called transition probabilities

In $P(w | t)$ each word is generated by its tag:

$$P(w \mid t) = P(w^{(1)} \mid t^{(1)})P(w^{(2)} \mid t^{(2)}) \ldots P(w^{(N)} \mid t^{(N)})$$

$P(w \mid t)$ are called emission probabilities.
HMMs as probabilistic automata

An HMM defines
Transition probabilities:
\( P(t_i | t_j) \)
Emission probabilities:
\( P(w_i | t_i) \)

Trigram model:
\( q_0 \)
\( a \)
\( 0.1 \)
\( 0.1 \)
\( 0.003 \)
\( 0.6 \)
\( NN_DT \)

\( v_0 \)
\( a \)
\( 0.2 \)
\( 0.5 \)
\( 0.009 \)
\( 0.7 \)
\( zealous \)

\( v_0 \)
\( a \)
\( 0.1 \)
\( 0.3 \)
\( 0.00024 \)
\( abandonment \)

\( v_0 \)
\( a \)
\( 0.1 \)
\( 0.002 \)
\( 0.55 \)
\( acts \)

\( v_0 \)
\( a \)
\( 0.1 \)
\( 0.001 \)
\( 0.001 \)
\( yields \)

\( v_0 \)
\( a \)
\( 0.1 \)
\( 0.001 \)
\( 0.45 \)

How would the automaton for a trigram HMM with transition probabilities \( P(t_i | t_j t_k) \) look like?

What about unigrams or n-grams?

HMM definition

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

- a set of \( N \) states \( Q = \{q_1,...,q_N\} \)
  - 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 \( V = \{v_1,...,v_m\} \)

- an \( N \times N \) state transition probability matrix \( A \)
  - with \( a_{ij} \) the probability of moving from \( q_i \) to \( q_j \).
  - \( (\sum_{j=1}^{N} a_{ij} = 1 \forall i; \quad 0 \leq a_{ij} \leq 1 \forall i, j) \)

- an \( N \times M \) symbol emission probability matrix \( B \)
  - with \( b_{ij} \) the probability of emitting symbol \( v_j \) in state \( q_i \).
  - \( (\sum_{i=1}^{N} b_{ij} = 1 \forall i; \quad 0 \leq b_{ij} \leq 1 \forall i, j) \)

- an initial state distribution vector \( \pi = (\pi_1,...,\pi_N) \)
  - with \( \pi_i \) the probability of being in state \( q_i \) at time \( t = 1 \).
  - \( (\sum_{i=1}^{N} \pi_i = 1 \quad 0 \leq \pi_i \leq 1 \forall i) \)
Building an HMM Tagger

To build an HMM tagger, we have to:

- **Train** the model, i.e. estimate its parameters (the transition and emission probabilities)
- **Easy case**: we have a corpus labeled with POS tags (supervised learning)

  - Define and implement a **tagging algorithm** that finds the best tag sequence \( t^* \) for each input sentence \( w \):
    \[
    t^* = \arg\max_t P(t)P(w \mid t)
    \]

Learning an HMM from labeled data

We count how often we see \( t_it_j \) and \( w_jt_i \) etc. in the data (use relative frequency estimates):

Learning the transition probabilities:

\[
P(t_j \mid t_i) = \frac{C(t_it_j)}{C(t_i)}
\]

Learning the emission probabilities:

\[
P(w_j \mid t_i) = \frac{C(w_jt_i)}{C(t_i)}
\]

Learning an HMM from unlabeled data

We can’t count anymore. We have to guess how often we’d expect to see \( t_it_j \) etc. in our data set. Call this expected count \( \langle C(.) \rangle \)

- Our estimate for the transition probabilities:
  \[
  \hat{P}(t_j \mid t_i) = \frac{\langle C(t_it_j) \rangle}{\langle C(t_i) \rangle}
  \]

- Our estimate for the emission probabilities:
  \[
  \hat{P}(w_j \mid t_i) = \frac{\langle C(w_jt_i) \rangle}{\langle C(t_i) \rangle}
  \]

- We will talk about how to obtain these counts on Friday
Finding the best tag sequence

The number of possible tag sequences is exponential in the length of the input sentence:

Each word can have up to T tags.
There are N words.
There are up to $N^T$ possible tag sequences.

We cannot enumerate all $N^T$ possible tag sequences.

But we can exploit the independence assumptions in the HMM to define an efficient algorithm that returns the tag sequence with the highest probability.

### Computing $P(t, w)$ for one tag sequence

Let $t = t_1 t_2 \ldots t_i$ be a tag sequence for the sentence $w = w_1 w_2 \ldots w_i$.

We can compute the probability $P(t, w)$ recursively:

$$P(t, w) = P(t_1) \cdot P(w_1 | t_1) \cdot \prod_{i=2}^{n} P(t_i | t_{i-1}) \cdot P(w_i | t_i)$$

### Using the trellis to find $t^*$

Let $t^*$ be the best tag sequence.

We can recursively compute $\text{trellis}[i][j]$ from the entries in the previous column $\text{trellis}[i-1][j]$.

$$\text{trellis}[i][j] = \max_{t_t} P(w_1 \ldots w_i, t_1 \ldots t_i = t_t)$$

At the end of the sentence, we pick the highest scoring entry in the last column of the trellis.
Retrieving $t^* = \text{argmax}_t P(t, w)$

By keeping one backpointer from each cell to the cell in the previous column that yields the highest probability, we can retrieve the most likely tag sequence when we’re done.

More about this on Friday…

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
(Full) 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
VBP: non-3rd person present tense
VBZ: 3rd person singular present tense

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

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
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’)

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)

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
- PRPS possessive
- WP: wh-pronoun
Determiners

Determiners precede noun phrases:
the/that/a/every book

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

Penn Treebank tags:
DT: determiner