Stephen Pulman
Department of Computer Science
Oxford University
HT 2016.

The aim of this series of lectures is to provide an introduction to some of the major topics in computational linguistics.

Some background reading


http://aclweb.org/anthology-new/ contains on-line versions of the Computational Linguistics journal, and proceedings of most of the major relevant conferences: ACL, EACL, COLING, EMNLP etc.
Provisional lecture schedule:

Lecture 1: some linguistics; part of speech tagging; chunking and shallow parsing.
Lecture 2: syntax and parsing; disambiguation
Lecture 3: introduction to semantics and inference for language
Lecture 4: word sense disambiguation; contextual interpretation: pronouns etc.; dialogue systems
Lecture 5: some natural language applications: information extraction, sentiment analysis.
We can describe the sounds of languages in articulatory terms: e.g. voiced vs. unvoiced consonants:

<table>
<thead>
<tr>
<th>Voiced</th>
<th>Unvoiced</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>p</td>
</tr>
<tr>
<td>d</td>
<td>t</td>
</tr>
<tr>
<td>g</td>
<td>k</td>
</tr>
<tr>
<td>z</td>
<td>s</td>
</tr>
</tbody>
</table>

Now consider the pronunciation of the plural marker -s on these words:

*cab/cabs, cup/cups, lid/lids, cat/cats, dog/dogs, brick/bricks*

What determines whether you pronounce -s as ‘s’ or ‘z’?
OK: that’s easy - now what about bridge/bridges, bus/buses, buzz/buzzes, church/churches?
Morphology studies the structure of words. **Inflectional morphology**: endings change according to number, tense, etc. 

*talk, talks, talked, talking etc.*

Easy in English, less so in French, getting more complicated in German, absolutely awful in Finnish, Hungarian, etc! **Derivational morphology**: new words from old.

*private, privatise, deprivatise, deprivatisation, deprivatisational, deprivatisationalist....*

(Eh? What is a deprivatisationalist?)
In Computational Linguistics we can mostly ignore phonetics and phonology, although people doing speech recognition and speech synthesis cannot. But morphology is something we have to deal with:

- fly+s → flies
- but: dog+s → dogs
- church+s → churches
- get+ing → getting
- but: meet+ing → meeting
- rely+able → reliable
- fly+er → flier
- but: fly+ing → flying

**Spelling changes:**

Norway and Switzerland have no **Blairesque** delusions of being “in the heart of Europe”...

“My Lords, does the noble Baroness agree that it is high time to **deprivatise** the whole of this clamping racket?”

NB: Only stems are listed in dictionaries, and new words are created all the time: deprivatise, deprivatisation, Blairesque, Thatcherize etc. are not in the OED.
Levels of description: Parts of speech

Noun (N)
- proper: Paris, James, Mr Smith, General Foods Inc.
- common: can appear in frame: the ___ is/are ...
  - countable: man, men, dog, symbol, idea, etc. Appear in plural, following ‘a’, ‘one’, ‘many’, etc.
  - mass: milk, furniture, knowledge etc. Do not typically appear in plural, do appear after ‘much’, but not ‘many’, etc.

Pronoun (Pron)
- personal: he, him, it ...
- possessive: his, yours, mine ...
- reflexive: myself, yourselves, ...
- "wh": who, which, what, ...
<table>
<thead>
<tr>
<th>Determiner (Det)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• articles: a, an, the.</td>
</tr>
<tr>
<td>• quantifiers: all, every, some ...</td>
</tr>
<tr>
<td>• demonstratives: this, that, these ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adjectives (Adj)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• attributive position: an old friend, an expensive book</td>
</tr>
<tr>
<td>• predicative position: a book which is expensive</td>
</tr>
<tr>
<td>• occur after be, seem, etc: ‘that seems/appears expensive</td>
</tr>
<tr>
<td>• comparatives: old+er, rich+er, (but: more intelligent)</td>
</tr>
<tr>
<td>• superlatives: old+est, rich+est, most intelligent</td>
</tr>
</tbody>
</table>
Levels of description: Parts of speech

**Verbs (V)**
Can appear in different inflected forms, e.g. walks, walked, walking. Stem (infinitive) form will appear in frame ‘be V -ing’. There are many different subcategories:

- intransitive: snores, sleeps, walks
- transitive: hits, likes, sees ...
- put the book on the shelf (*put the book, *put on the shelf)
- ‘be’ is often called the ‘copula’
- auxiliary verbs: modals: can, may, might, etc; and various forms of have and be, do etc.

**Prepositions (P)**

- in, on, under, at, beneath, from, of ...
- often called ‘particles’ when linked with verbs, as in ‘look up to’, ‘put up with’, ‘rely on’ etc.
Adverbs (Adv)

Frequently derived from Adj by adding -ly
- ‘S’ type: frankly, obviously, certainly ...
- ‘VP’ type: manner: quickly, deliberately, noisily ..., time: now, then, still ...
- miscellaneous: too, also, even, maybe, not, only...

Adverbs can appear in many places in a sentence, between - not generally inside - phrasal units. Note that many things traditionally called time and place adverbials are syntactically prepositional phrases with adverbial meanings.

Conjunctions (Conj)

- coordinating: and, or, but, if..
- subordinating: although, while, because...
We distinguish **Open** class words = N, V, Adj, Adv (new ones appearing all the time) from **Closed** class words = the rest (very few new ones).

Note also that:

(i) some words can belong to more than one category (e.g. can), or (arguably) none at all: as, so

(ii) base forms can change category by addition of affixes:

\[
\begin{align*}
V + \text{ing} & \rightarrow N \quad \text{the writings of Proust} \\
V + \text{ed} & \rightarrow \text{Adj} \quad \text{the collapsed wall} \\
\text{Adj} + \text{ise} & \rightarrow V \quad \text{privatise} \\
V + \text{ation} & \rightarrow N \quad \text{privatisation}
\end{align*}
\]

etc.
Any natural language processing system needs to associate information with words. The POS information for a word tells us how it fits together with other words to make a sentence, and gives us some limited semantic information.

*Flying planes can be dangerous*

*Flying/ADJ planes/N can/M be/V dangerous/ADJ*

*Flying/V planes/N can/M be/V dangerous/ADJ*

*Time/N flies/V*

*Time/V flies/N*

The most likely POS for an ambiguous word will usually depend on the context:

*He saw the flies on the meat.*

*He flies to Paris every week*

Further reading for this section: Jurafsky and Martin, Ch 8.1-8.2
Taoiseach (prime minister) Enda Kenny has admitted his coalition government has failed to secure a return to office as the Irish election count continues.

From 6am BST today, all services across Europe, the Middle East and Africa, as well as India, have been operating with significant improvement. We continue to monitor the situation 24x7 to ensure ongoing stability.
Part of speech (POS) tagging

What is it?
Automatically assigning a part of speech to the words in a sentence: e.g. the/DET cat/N sat/V on/P the/DET mat/N ./.

Why?
- useful preprocessing step for later parsing, since it reduces ambiguity: How the time/N flies vs. I want you to time/V flies
- enrich a corpus with useful information: ‘find all occurrences of ‘time’ as a verb, followed by a noun’
- can help guess categories of unknown words:
  ’Twas brillig, and the slithy toves
  Did gyre and gimble in the wabe;
  All mimsy were the borogoves,
  And the mome raths outgrabe.
Various different tagsets

- The British National Corpus:
  http://www.natcorp.ox.ac.uk/docs/c5spec.html
  Quite fine-grained, about 150 tags, linguistically well motivated.

- The Penn Treebank:
  http://www.comp.leeds.ac.uk/ccalas/tagsets/upenn.html
  About 50 tags, fails to make some important distinctions.

- Google Universal Tagset:
  http://arxiv.org/abs/1104.2086
  Very small, about 12 tags, aims to provide comparability between languages.
A Markov Model is essentially a finite state machine with probabilities on the transitions:
An HMM is a Markov model which also emits symbols when in a particular state: each symbol having a probability of being emitted:
Hidden Markov Models are so called because in their usual application, we are given a sequence of output symbols and we must decide which sequence of states was most likely to have emitted that sequence. In speech recognition, for example, the ‘symbols’ are representations of sounds, and the states we are trying to recover are the words of which these sounds are pronunciations.

In our case, we have a sequence of words, and we want to know their parts of speech *in that sequence*. We conceptualise this as an HMM problem by regarding each such word sequence as the output of a traversal through a sequence of ‘part of speech’ (POS) states.

Now, if we can build such an HMM, it gives us the probability of emitting a word when in a POS state, and also the probability of making a transition from one POS state to the next. But how do we reverse this - work out the most probable sequence of POS states from the output sequence? We use Bayes’ Theorem.
We want to find \( P(t_1, \ldots, t_n \mid w_1, \ldots, w_n) \). *Bayes’ Theorem* tells us that this is the same as:

\[
P(t_1 \ldots t_N \mid w_1 \ldots w_N) = \frac{P(t_1 \ldots t_N) \times P(w_1 \ldots w_N \mid t_1 \ldots t_N)}{P(w_1 \ldots w_N)}
\]

We want to find the sequence of ts that give us the highest value in this equation. Note \( P(w_1 \ldots w_n) \) will be the same for all sequences if the words are given so we can ignore it. But the top line is still too much to calculate directly, since we would never have enough data to estimate the probabilities from.

The chain rule:

\[
P(A, B) = P(A \mid B) \times P(B)
\]
\[
P(A, B, C) = P(A \mid B, C) \times P(B \mid C) \times P(C)
\]
\[
P(A, B, C, D) = P(A \mid B, C, D) \times P(B \mid C, D) \times P(C \mid D) \times P(D)
\]
\[
P(C_1 \ldots C_n) = P(C_1 \mid C_2 \ldots C_n) \times P(C_2 \mid C_3 \ldots C_n) \times \cdots \times P(C_n)
\]
Conditional Independence Assumptions

Tag transition distribution:

\[ P(t_1 \ldots t_N) = \prod_{i=1}^{N} P(t_i | \ldots t_{i-1} \ldots t_1) \approx \prod_{i=1}^{N} P(t_i | t_{i-1}) \]

Word emission distribution:

\[ P(w_1 \ldots w_N | t_1 \ldots t_N) = \prod_{i=1}^{N} P(w_i | w_{i-1} \ldots w_1, t_1 \ldots t_N) \]

\[ \approx \prod_{i=1}^{N} P(w_i | t_i) \]

NB: we invent a start (and end) of sentence marker or dummy word (e.g. **start**, **end**) so that \( t_{i-1} \) exists even where \( i=1 \).
Use labelled training data for maximum likelihood estimates (MLE).

Example: Part-of-Speech Tagging

- \(t_1 \ldots t_N\) tags correspond to states
- \(w_1 \ldots w_N\) words correspond to observations
- by counting words and tags we can calculate MLEs for the model’s parameters
- Tag transitions:
  \[
p(t_{n+1} = j| t_n = i) = \frac{\text{Count}(t=i, t=j)}{\text{Count}(t=i)}
\]
- Word emissions:
  \[
p(w_n = \text{word}| t_n = k) = \frac{\text{Count}(w=\text{word}, t=k)}{\text{Count}(t=k)}
\]

There are various suitable corpora for this, e.g. the Penn Treebank (UPenn) and the British National Corpus (Oxford Text Archive).
Counts from an corpus annotated with part-of-speech tags (from BNC):

<table>
<thead>
<tr>
<th>First tag</th>
<th>AT</th>
<th>BEZ</th>
<th>IN</th>
<th>NN</th>
<th>VB</th>
<th>PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>48636</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BEZ</td>
<td>1973</td>
<td>0</td>
<td>426</td>
<td>187</td>
<td>0</td>
<td>38</td>
</tr>
<tr>
<td>IN</td>
<td>43322</td>
<td>0</td>
<td>1352</td>
<td>17314</td>
<td>0</td>
<td>185</td>
</tr>
<tr>
<td>NN</td>
<td>1067</td>
<td>3720</td>
<td>42470</td>
<td>11773</td>
<td>614</td>
<td>21392</td>
</tr>
<tr>
<td>VB</td>
<td>6072</td>
<td>42</td>
<td>4758</td>
<td>1476</td>
<td>129</td>
<td>1522</td>
</tr>
<tr>
<td>PERIOD</td>
<td>8016</td>
<td>75</td>
<td>4656</td>
<td>1329</td>
<td>954</td>
<td>0</td>
</tr>
</tbody>
</table>

(AT=article, BEZ=is/was, IN=prep, NN=sing noun, VB=base form of verb...)

<table>
<thead>
<tr>
<th>First tag</th>
<th>AT</th>
<th>BEZ</th>
<th>IN</th>
<th>NN</th>
<th>VB</th>
<th>PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>bear</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>43</td>
<td>0</td>
</tr>
<tr>
<td>is</td>
<td>0</td>
<td>10065</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>move</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>133</td>
<td>0</td>
</tr>
<tr>
<td>on</td>
<td>0</td>
<td>0</td>
<td>5484</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>president</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>382</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>progress</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>108</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
The Viterbi Algorithm

How can we compute the most likely state sequence efficiently, given that for an N word long sentence, with an average of M POS cats per word, there will be $M^N$ possible sequences? The Viterbi algorithm is a dynamic programming algorithm that allows us to compute the best sequence (‘path’), discarding others as we go along.

for each Word $W_1...n$
for each POS category $C_j$
  for each path ending in some $C_k$ for $W_{i-1}$
    compute $P(C_j|C_k)*P(W_i|C_j)*$score of path
    keep a record of best scoring path to $C_j$
Iterate through a lattice:

\[
\text{the } \quad \text{cat} \quad \text{sat} \quad \text{on}
\]
Implementing the Viterbi Algorithm

For a sentence $w_1...w_N$ of $N$ words, assuming a set $t_1...t_K$ of $K$ distinct part of speech tags, create an $K$x$N$ array called ‘Score’, and another $K$x$N$ array called ‘Backpointer’:

**Initialise:**

\[
\text{for } i = 1 \rightarrow K \text{ do} \\
\quad \text{Score}(i,1) = P(w_1|t_i) \times P(t_i|\langle \text{start} \rangle) \\
\text{end for}
\]

**Induction:**

\[
\text{for } j = 2 \rightarrow N \text{ do} \\
\quad \text{for } i = 1 \rightarrow K \text{ do} \\
\quad \quad \text{Score}(i,j) = \max_{k=1...K} \text{Score}(k,j-1) \times P(t_i|t_k) \times P(w_j|t_i) \\
\quad \quad \text{Backpointer}(i,j) = \max \text{ k from previous line} \\
\quad \text{end for} \\
\text{end for}
\]

**Back tracing the best tagging:**

\[
\text{t}_N = \max_i \text{Score}(i,N) \\
\text{for } i = N - 1 \rightarrow 1 \text{ do} \\
\quad \text{t}_i = \text{Backpointer}(t_{i+1},i+1) \\
\text{end for}
\]
( (S
  (NP
    (NP (NNP Pierre) (NNP Vinken) )
    (, ,)
  )
  (ADJP
    (NP (CD 61) (NNS years) )
    (JJ old) )
  )
  (, ,) )
(VP (MD will)
  (VP (VB join)
    (NP (DT the) (NN board) )
    (PP (IN as)
      (NP (DT a) (JJ nonexecutive) (NN director) ))
    (NP (NNP Nov.) (CD 29) )))
( . . ) )
Later, we will discuss parsing techniques that are capable of assigning full trees like this to sentences. But for now, we concentrate on a shallower level of analysis, known as ‘chunking’. We’ll concentrate on two kinds of ‘chunk’:

1. ‘base NPs’: non-recursive NPs i.e. not containing any other NPs.
2. ‘verb groups’ (VG): components of a Verb Phrase, excluding any obligatory complements or optional modifiers (‘adjuncts’).

Why not full Verb Phrases? Because in order to do that accurately, we would need information about the required complements of each verb, in order to know that a sequence like ‘on a horse’ should be part of the verb phrase in:

   He relied on the horse.

but not in:

   He arrived on the horse.

Note that:

   *He relied. (felt to be incomplete)
   He arrived. (OK)
Given a complete parse tree like that shown earlier, we can recover the chunks as below:

[Pierre Vinken]/NP, [61 years]/NP old, [will join]/VG [the board]/NP as [a nonexecutive director]/NP [Nov. 29]/NP.

Note that there is no nesting or overlapping of chunks. We can transform this to an equivalent ‘chunk tag’ representation:

Pierre/B-NP Vinken/I-NP, 61/B-NP years/I-NP old/O, will/B-VG join/I-VG the/B-NP board/I-NP as/O ...

A tag of the form B-X means ‘beginning of an X’, I-X means ‘inside an X’, and O means ‘outside any chunk’. It is easy to see how to reconstruct the actual chunks from this notation (NB sometimes B means something slightly different...)

\[
W1/I \ W2/O \rightarrow W1] \ W2 \\
W1/I \ W2/B \rightarrow W1][W2 \\
W1/O \ W2/I \rightarrow W1 \ [W2
\]
Now we can regard chunking as an extension of tagging, and use HMMs to do it automatically:

- We train an HMM from a POS and chunk-tagged corpus.
- The simplest way to do this is to treat the POS tags as if they were words, and the chunk tags as if they were POS tags.
- Instead of estimating output probabilities like $P(\text{the} \mid \text{DT})$ and transition probabilities like $P(\text{NN} \mid \text{DT})$, we have output probabilities of the form: $P(\text{DT} \mid \text{B-NP})$ and transition probabilities of the form $P(\text{I-NP} \mid \text{B-NP})$.
- Now in order to do chunking we first do ordinary POS tagging of the sentence, and then we tag the POS tags with chunk tags.
- Then we recover the chunks from these tags.
In fact, it turns out that performance can be improved if we enrich the chunk tags by combining them with POS tags and selected words (e.g. all the closed class words). From the training corpus we produce a derived corpus where the tags are:

<table>
<thead>
<tr>
<th>Word</th>
<th>POS tag</th>
<th>Chunk tag</th>
<th>Chunk+POS</th>
<th>Chunk+POS+Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>You</td>
<td>PRP</td>
<td>B-NP</td>
<td>PRP-B-NP</td>
<td>you-PRP-B-NP</td>
</tr>
<tr>
<td>will</td>
<td>MD</td>
<td>B-VG</td>
<td>MD-B-VP</td>
<td>will-MD-B-VG</td>
</tr>
<tr>
<td>start</td>
<td>VB</td>
<td>I-VG</td>
<td>VB-I-VG</td>
<td>VB-I-VG</td>
</tr>
<tr>
<td>the</td>
<td>DT</td>
<td>B-NP</td>
<td>DT-B-NP</td>
<td>the-DT-B-NP</td>
</tr>
<tr>
<td>car</td>
<td>NN</td>
<td>I-NP</td>
<td>NN-I-NP</td>
<td>NN-I-NP</td>
</tr>
</tbody>
</table>

We can vary the words we include by experimenting to minimise error.
Measuring accuracy

For some tasks, like POS tagging, we simply count the number of errors. A typical HMM tagger will achieve about 95% accuracy, so about 1 in 20 POS tags will be wrongly assigned. (The unigram baseline is about 80%). Note that this figure is for a ‘trigram’ tagger, where the transition probabilities are of the form \( P(C_n \mid C_{n-1}, C_{n-2}) \), i.e. we condition on the previous two tags.

ASIDE:

Strictly speaking, trigrams violate the Markov assumption that previous states don’t affect what happens in the current state. A well known way of overcoming this is to make our POS tags more complex, to encode earlier states implicitly:

\[
\texttt{<start>/START the/DT man/NN arrived/VBD <end>/END} \\
\texttt{<start>/START the/START-DT man/DT-NN arrived/NN-VBD <end>/VBD-END}
\]

Now we can just proceed in training and tagging just as for the bigram case, with a simple unpacking of POS labels when we have finished.
We also measure accuracy by RECALL and PRECISION, comparing the system’s output to a ‘gold standard’ reference corpus. For example, for the NP chunking task we would have:

\[
\text{Recall} = \frac{\text{Number of correct NPs found by system}}{\text{Number of NPs in test corpus}}
\]

\[
\text{Precision} = \frac{\text{Number of correct NPs found}}{\text{Number of NPs found by system}}
\]

Recall and precision are often combined in the ‘F’ measure:

\[
F = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}
\]
Accuracy of HMM chunking

As reported by Molina and Pla (2002):

<table>
<thead>
<tr>
<th>Chunk tags</th>
<th>precision</th>
<th>recall</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>84.31</td>
<td>84.35</td>
<td>84.33</td>
</tr>
<tr>
<td>with POS tags</td>
<td>89.58</td>
<td>89.55</td>
<td>89.57</td>
</tr>
<tr>
<td>with POS+words</td>
<td>91.96</td>
<td>92.41</td>
<td>92.19</td>
</tr>
</tbody>
</table>

Trained on sections 15-18 of WSJ Penn Treebank, tested on 20. Results improve by about 1.5% if trained on 0-19.
In HMM tagging we only use two features of the context: the word itself and the tag of the previous word. It would be nice to use a richer set of possibly complex features, e.g.

- previous tag = VB; previous word = ‘be’; current word ends in ‘y’
- current word is ‘TO’ and next tag is VB*; etc.

It is usually convenient to represent these as binary (0/1) functions

\[
f_m(i, w, t_i, t_{i-1}) = \begin{cases} 
1 & \text{if } t_{i-1} = \text{DT} \land t_i = \text{ADJ} \land w_i \text{ ends in } 'y', \\
0 & \text{otherwise.}
\end{cases}
\]

We collect lots of these function into an M dimensional binary feature map or vector:

\[
[f_1(i, w, t_i, t_{i-1}), f_2(i, w, t_i, t_{i-1}), \ldots, f_M(i, w, t_i, t_{i-1})]
\]
Now tagging is a label classification problem, so you can train your favourite classifier. Represent the data as a set of \( \langle \text{feature vector, label} \rangle \) pairs and push the button.

- **Maximum Entropy Model:**
  [http://www.aclweb.org/anthology/W96-0213](http://www.aclweb.org/anthology/W96-0213)

- **Averaged Perceptron:**
  [http://www.aclweb.org/anthology/W02-1001](http://www.aclweb.org/anthology/W02-1001)

- **Conditional Random Fields:** Fei Sha and Fernando Pereira, 2003.
  Shallow Parsing with Conditional Random Fields.
  [http://aclweb.org/anthology/N/N03/N03-1028.pdf](http://aclweb.org/anthology/N/N03/N03-1028.pdf)

- **Neural Networks:**
  Ronan Collobert et al., 2011, Natural Language Processing (Almost) from Scratch