Feature-based classification

Micha Elsner

January 17, 2014
Reading material

We are done with Chapter 1 of Charniak; for reference in this section, I have posted some very detailed slides by Klein and Manning.
Linguistic idea: classes of words that can appear in specific contexts

- Example from Carnie: context for nouns
  - The *man* loved peanut butter cookies
  - The *puppy* loved peanut butter cookies
  - The *king* loved peanut butter cookies
  - *The in* loved peanut butter cookies

- Not coincidentally, these words can also fit here:
  - Kim hit a *man*
  - Kim hit a *puppy*
  - Kim hit a *king*
  - *Kim hit a(n) in*

If a set of words can appear in (many) of the same contexts, we call it a Part of Speech
Coarse-grained and fine-grained

There are contexts in which we can’t use all nouns. What might these be?
Coarse-grained and fine-grained

There are contexts in which we can’t use all nouns. What might these be?

This suggests there are subtypes of noun:

- Singular/plural: Kim hit twelve puppies/*puppy
- Common/proper: That man is a linguist/*Chomsky
- Mass/count: In the warehouse is a pile of steel/*wrench
- Animate/inanimate: The king/*wrench who is over there
- Masculine/feminine/neuter: The king/queen/demon bit himself/herself/itself

Parts of speech are often subdivided along some of these lines

- Although subdividing too much creates data sparsity problems
- There isn’t necessarily a principled distinction between POS-level distinctions and lower-level ones
- Good tagsets differ in different languages
Why study part of speech?

- Initial stage of parsing
- Unknown word handling in LMs
- Corpus diving: find instances of “…make PRONOUN VERB the…”
- Theory of morphophonology—learn markers for categories
- Measure reading difficulty (simple Wikipedia, etc)
The Penn tagset

Penn treebank: tagged, parsed data (about 4.5 mi words of American English)

- Mitch Marcus, Beatrice Santorini, others
- Wall Street Journal articles, about 1 mi words
- 98732 articles
- Annotated by UPenn grad students, using initial automatic tagger and hand-correcting

Tagging guide is available in Readings on Carmen

- Tags are multi-letter codes
- Each letter is more specific
  - Initial J: adjective
  - JJR: adjective, comparative
- Final letters tend to symbolize common suffixes
  - R = “er”, VBD = “ed”, etc
Part of speech tagging

On the other side of the earth poets and artists had begun to dream of a strange, dank cyclopean city whilst a young sculptor had moulded in his sleep the form of the dreaded Cthulhu. –H P Lovecraft

What syntactic category do we apply to “Cthulhu”?  
- Evidence from context...
- And morphophonology...
- For now, let’s take a look at the form of the word (in isolation)
Morphosyntactic features

- Strange words are often some kind of noun or verb
  - ...these are open classes
- Words beginning with capitals are often proper nouns
- Words with phonologically bad letter combinations are often proper nouns from a foreign language ("Zhao", "Rekjavik")
- "hu" isn’t a very good verbal ending

“Cthulhu” is probably a proper noun
Let’s build a model of part of speech tag $T$ given these features $F_{1:n}$?
Maximum likelihood

What is the simplest way to model this?
What is the simplest way to model this? Just count feature vectors:

\[ P(T|F_{1:n}) = \frac{\#(T, F_{1:n})}{\#(F_{1:n})} \]

When will this work? When won’t it?
Classification

This set of lectures/project will introduce several methods for putting objects in *categories* based on *features*

- The case study (project out today) is POS tagging

So far, we’ve seen two approaches to combining information from multiple sources:

- Interpolation
- Independence assumptions
  - ...which is where we’re headed today
The Naive Bayes classifier for morphosyntax

Making this work for morphosyntax:

▶ Since I don’t have any particular beliefs about dependence between features
▶ I’ll assume all are independent
  ▶ In classification, this is called the Naive Bayes assumption
▶ This is just like the language model...

\[
P(F_{1:k} | T) = \prod_{i=1}^{k} P(f_i | T)
\]
Digression: reading the diagram

Graphical model
(Also called “Bayes net”)
- Encodes a joint probability distribution as a graph
- Nodes are random variables
- Arrows indicate conditional dependences
- Distribution of a node conditioned on all its parents
- Node’s value affects all its children
Independence assumptions

Independence assumptions in the LM

- The information: all the words \( w_1, w_2 \ldots w_n \)
- The task: distinguish translation from original
- The approach: make independence assumptions to simplify

\[
P(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n+1} P(w_i|w_{i-1})
\]

Independence in NB

- We throw away information about relationships between features
Estimating Naive Bayes parameters

And we estimate:

$$\hat{P}(capital = false|T = noun) = \frac{ #(capital = false, noun) + \lambda }{ #(noun) + 2\lambda }$$

(Or some more complex smoother perhaps)

- $2\lambda$ because the domain of $capital$ is $true$, $false$

Do we even need $\lambda$?
Making decisions with Naive Bayes

To build a Naive Bayes classifier:

- Build a model $P(F \mid T = t)$ for all $t$
  - A noun model, a verb model, etc
- Now we use Bayes’ rule to derive $P(T \mid F)$
Making decisions with Naive Bayes

To build a Naive Bayes classifier:

- Build a model $P(F|T = t)$ for all $t$
  - A noun model, a verb model, etc
- Now we use Bayes’ rule to derive $P(T|F)$

\[
P(T = t|F) = \frac{P(F|T = t)P(T = t)}{\sum_{t' \in T} P(F|T = t')}
\]

- This is the form of Bayes’ rule with the denominator $P(F)$ expanded

Now pick the tag $t$ for which $P(T = t|F)$ is highest
We compute $P(T = t | F)$ for all $t \in T$

- The Penn tagset, which we will look at soon, has about 40 finer-grained POS tags
  - So this is a loop over 40 alternatives
- We compute all the numerators, and sum them together to get the denominator
  - Or if we just want the best tag, we don’t even need the denominator!
Thought experiment:

- Classify: “are the gods favorable or not?”
- Feature one, the Oracle of Delphi, is 60% accurate
  - If the gods are really favorable, the Oracle replies “yes” 60% of the time
  - If not, the Oracle replies “no” 60% of the time
- Features 2-k, ordinary peasants, are 51% accurate

Suppose the gods are unfavorable this time, but the Oracle says “yes”

- How many peasants does it take to outvote the oracle?
- Is this reasonable?
The independence assumptions can hurt us...
A fairly artificial case: we design the following classifier:

- Two classes, *noun* and *adj*
- Feature 1: word begins with “h”
- Feature 2: word begins with a fricative
Hawaiian

Apply this classifier to Hawaiian!

- Hawaiian has only one fricative, which happens to be “h”

```
<table>
<thead>
<tr>
<th></th>
<th>noun</th>
<th>adj</th>
</tr>
</thead>
<tbody>
<tr>
<td>prior</td>
<td>.75</td>
<td>.25</td>
</tr>
<tr>
<td>P(&quot;h&quot;</td>
<td>t)</td>
<td>.2</td>
</tr>
<tr>
<td>P(fric</td>
<td>t)</td>
<td>.2</td>
</tr>
</tbody>
</table>
```

- Suppose (made up):
  - Classify “Hawai’i”
    - Features: “h”: true, fric:true
    - \( P(noun|true, true) \propto .75 \times .2 \times .2 = .324 \)
    - \( P(adj|true, true) \propto .25 \times .5 \times .5 = .675 \)

But really, these are the same feature! True probabilities are:

- \( P(noun|true) \propto .75 \times .2 = .54 \)
- \( P(adj|true) \propto .25 \times .5 = .45 \)
Independence assumptions

- Making a decision costs you probability...
- Making the same decision twice costs you probability squared
- A problem with any correlated features
  - Not just disguised identical features as in the example
  - And the correlations aren’t always obvious
- Correlated features increase our confidence in an answer
  - Because we think they represent independent sources of evidence

In the next couple of lectures, we will look at a framework that deals with the correlated feature problem