Sequence models

Micha Elsner

January 31, 2014
Readings and stuff

We’re back in Charniak (sections 3.1 to 3.6.1)

▶ We’ll be covering the rest of chapter 3 much later

I’ll also point to:

▶ Jason Eisner’s HMM lecture
▶ Reading materials:
  http://www.cs.jhu.edu/~jason/papers
    ▶ Look for “An interactive spreadsheet for teaching the forward-backward algorithm”
▶ Video: http://videolectures.net/hltss2010_eisner_plm/video/2/
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

- So far, discussed morphosyntactic features
- Context also a strong cue to POS tag
- DT ADJ often followed by NOUN
- NOUN often followed by end of sentence
Context words

Consider adding context to our feature-based model:

- 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
- Previous word is “dreaded”
- Next word is “.”

Some of you may have tried this out—this simple change improves things a lot!

- From about 93% to over 96%
- In about three lines of code
Context tags

But context words aren’t the whole story:

- Adding all these words expands the feature set a lot
  - Problems with speed/overfitting
- And the context words themselves may be rare or ambiguous

How did you tag “the mome raths outgrabe”? 
Mome raths outgrabe

- DT (JJ/NN) NNS VBD . “the (angry/pig) farmers shouted .”
  - “mome raths” are a kind of creature
  - “outgr(ibing)” is an activity
- DT NN VB NN . “the pig likes salad .”
  - “mome” is a kind of creature
  - “rath(ing)” is an activity which you do to something
  - “outgrabe” is a thing you can “rath”
- These sequences are not interchangeable!
  - DT JJ VB VBD “the angry likes shouted” is awful
- Knowing the previous word is “raths” isn’t helpful on its own
  - The tags we assign to “raths” and “outgrabe” depend on one another
More examples

Ambiguous context words

- If a can can can, a can can can a can, can’t it?
- Buffalo buffalo Buffalo buffalo buffalo buffalo buffalo buffalo Buffalo buffalo

Once we’ve figured out tags for the first few words, the next word is easy to tag
Context tags as features

Add the previous and next tag as features:

- ... 
- Previous tag is “JJ” 
- Next tag is “.”

This works even if we don’t know the word “dreaded”

- But how do we implement it?
- Circular reasoning: in order to tag “dreaded” as JJ, we need to run the tagger
  - To run the tagger, need the tag of the next word
  - To tag the next word, need to run the tagger...

Oracle system

A system that uses information from the true test set answers (that wouldn’t be available in real life)

- Can demonstrate how much it would help to have certain information
- In this case actually only a bit better than prev/next word
Improving the model

- Our current classifiers treat each word/tag as independent
- Instead, let's model the words and tags of the entire sentence jointly

\[ P(W_{1:n}, T_{1:n}) \]

We'll need to make this tractable by making independence assumptions

- ...Just like we always do
Markov modeling

Distribution over words and tags:

\[ P(W_1:n, T_1:n) \]

Part one: bigram language model for POS tags:

\[ P(T_1:n) = \prod_{i=0}^{n+1} P(T_i|T_{i-1}) \]

Just like our other LM:

▶ \( P(\text{The}|<s>) \) \( P(\text{debates}|\text{The}) \) \( P(\text{of}|\text{debates}) \)...

▶ \( P(\text{DT}|<s>) \) \( P(\text{NN}|\text{DT}) \) \( P(\text{IN}|\text{NN}) \)...

This is a Markov assumption:
Naive-Bayes style model of the words

\[ P(W_{1:n} | T_{1:n}) = \prod_i P(W_i | T_i) \]

- P(The|DT) P(debates|NN) P(of|IN)
Putting things together

The joint probability $P(W, T)$ is:

$$
P(W_{1:n}, T_{1:n}) = P(W_{1:n} | T_{1:n}) P(T_{1:n})
$$

$$
= \prod_{i=0}^{n+1} P_{\text{trans}}(T_i | T_{i-1}) P_{\text{emit}}(W_i | T_i)
$$

- Tag to tag probabilities called *transitions*
  - From bigram model
- Tag to word probabilities called *emissions*
  - From Naive Bayes model
- Pretend words $W_0 = \langle s \rangle$, $W_{n+1} = \langle /s \rangle$
This is a Hidden Markov Model (HMM):

- It’s *hidden* because (at test time) we don’t know the tags
  - Unlike the language model project, where we knew all the words
- It’s basically a big Naive Bayes classifier
  - $P_{\text{trans}}(T_i | T_{i-1})$ and $P_{\text{trans}}(T_{i+1} | T_i)$ work like $P(T)$ in the Naive Bayes model
  - Notice that information propagates from both the left and the right
- “DT JJ VB” dispreferred because $P(VB|JJ)$ is low
The conditional

If we want to predict tag sequences (like classification):

\[
P(T_{1:n} \mid W_{1:n}) = \frac{P(W_{1:n} \mid T_{1:n})P(T_{1:n})}{P(W_{1:n})}
\]

Our HMM equation for the joint goes into the numerator:

\[
P(T_{1:n} \mid W_{1:n}) = \prod_{i=0}^{n+1} P_{\text{trans}}(T_i \mid T_{i-1})P_{\text{emit}}(W_i \mid T_i)
\]

The denominator is the sum over all the numerators:

\[
P(T_{1:n} \mid W_{1:n}) = \frac{\prod_{i=0}^{n+1} P_{\text{trans}}(T_i \mid T_{i-1})P_{\text{emit}}(W_i \mid T_i)}{\sum_{T'_{1:n}} P(T'_{1:n}, W_{1:n})}
\]
Parts of a CL project (redux)

- Data collection, problem design
  - Part of speech tagging with context features
- Constructing a formal model
  - Let’s use an HMM
- Estimation: using training data to learn model parameters
  - Next couple slides (really easy!)
- Inference: using parameters to make decisions
  - Somewhat difficult; we’ll proceed slowly
- Evaluation and analysis
  - Same as last project
Estimation (emissions)

The tag-to-word transitions $P_{\text{emit}}(W|T)$

- You just finished doing this!
- You should know how to do it by now
Estimation (transitions)

The tag-to-tag transitions:

| $t$  | $P_{trans}(T_i = t| T_{i-1} = DT)$ |
|------|-----------------------------------|
| <s>  | 9.7e-5                            |
| IN   | 0.0096                            |
| DT   | 0.0015                            |
| NNP  | 0.11                              |
| NN   | 0.47                              |
| NNS  | 0.07                              |
| JJ   | 0.21                              |

Estimate: same as in language model:

\[
\hat{P}(T_i = a| T_{i-1} = b) = \frac{\#(ab) + \lambda}{\#(a) + \lambda|T|}
\]

- $|T|$: the number of tag types $t$
- (Can use more sophisticated smoothing, of course!)
Inference for Naive Bayes

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

Goal of inference: given test item with features \( F \), what is best tag \( T \)?
Evaluate probability for each tag \( t \)
  - Pick the best one

46 tags
  - We evaluate \( P(F|T = t)P(T = t) \) 46 times per test item

No big deal!
Basic inference for the HMM

\[ P(T_{1:n} \mid W_{1:n}) = \frac{P(W_{1:n} \mid T_{1:n})P(T_{1:n})}{P(W_{1:n})} \]

Goal of inference: given test sentence with words \( W_1 \ldots W_n \), what is best sequence of tags \( T_1 \ldots T_n \)?

- Evaluate probability for each sequence of possible tags \( T_1 \ldots T_n \)
- Pick the best one

46 tags for each word; average sentence is 25 words long

- 46 possible tags for \( w_1 \), 46 for \( w_2 \ldots \)
- \( 46^{25} \) sequences
  \( (370634456879779497815637488681697333477376) \)
- Bad algorithm: try each of these, pick the best
  - Your assignment will not finish by the due date...