Lecture 14
Sequence tagging, and social media NLP

Intro to NLP, CS585, Fall 2014
http://people.cs.umass.edu/~brenocon/inlp2014/
Brendan O’Connor
• Added after lecture, will review on Thursday:
  • The three main structured models are: (1) struct perceptron, (2) crf’s, (3) structsvm’s. All of them work the same at test time (decoding via the viterbi algorithm, by maximizing a linear goodness score). Only at training time are they different.
  • Averaged perceptron is probably the simplest to implement and use. Lots of practitioners in NLP who don’t care about fancy machine learning often use it. I actually like CRF’s myself because of they have a probabilistic interpretation, but that doesn’t always matter. Training CRF’s is slightly more complicated than struct perceptrons (not that much more complicated, but like a lecture’s worth of material), so I figured we could skip it in this class.
  • Instead of averaging, you can also do early stopping: keep a development set and evaluate accuracy on it every iteration through the data. Choose the theta that did best. I don’t know which method is better (different researchers may prefer different methods). Averaging has the advantage that there aren’t really any hyperparameters to tune (well, the learning rate to a certain extent).
  • Why does averaging work? Theta is bouncing a lot around the space, because the perceptron doesn’t know how to prefer solutions according to the magnitude of the errors it makes. The value of theta will be overfitted towards doing well on the most recent examples it’s seen. If you average, you average away some of the noise. Averaging is used in other areas of machine learning too. It’s a form of regularization.
  • Perceptron learning is actually a form of gradient descent. It’s not on the logistic regression log-likelihood, but instead the gradients of a different function (the “1-0” loss).
  • The Collins 2002 paper that introduced the structured perceptron is still great to read for more details: http://www.cs.columbia.edu/~mcollins/papers/tagperc.pdf
  • More on the classification perceptron: see Hal Daume’s book chapter draft, http://ciml.info/dl/v0_9/ciml-v0_9-ch03.pdf
• Is perceptron learning a form of gradient descent? Yes!

• Stochastic gradient descent (ascent) algorithm: on every training example, increment gradient

\[
\theta := \theta + \eta g_i(x_i, y_i)
\]

gradient for just one example

• Log-linear gradient for SGD

log-likelihood: how good model is at predicting gold \( y_i \)

\[
g_i = \frac{\partial}{\partial \theta_j} \left[ \theta^T f(x_i, y_i) - \log \sum_{y' \in \mathcal{Y}} \exp \theta^T f(x_i, y') \right]
\]

\[
= f_j(x_i, y_i) - \sum_{y'} p(y'|x) f_j(x_i, y')
\]

feature’s expected value, under model’s prediction distribution

• Perceptron as SGD

perceptron neg-loss: different measure of evaluating model predictions

\[
y^* = \arg \max_y \theta^T f(x, y)
\]

\[
g_i = \frac{\partial}{\partial \theta_j} \left[ \theta^T f(x_i, y_i) - \theta^T f(x_i, y^*) \right]
\]

\[
= f_j(x_i, y_i) - f_j(x_i, y^*)
\]

In both cases: loss gradients want to make gold-standard features match predicted-structure features. Either you care about a distribution over all outputs ... or just the best output
Log-linear gradient for SGD

log-likelihood: how good model is at predicting gold \( y_i \)

\[
g_i = \frac{\partial}{\partial \theta_j} \left[ \theta^T f(x_i, y_i) - \log \sum_{y' \in Y} \exp \theta^T f(x_i, y') \right]
\]

\[
g_i = f_j(x_i, y_i) - \sum_{y'} p(y'|x) f_j(x_i, y')
\]

feature’s expected value, under model’s prediction distribution

Perceptron as SGD

\[
y^* = \arg \max_y \theta^T f(x, y)
\]

perceptron neg-loss: different measure of evaluating model predictions

\[
g_i = \frac{\partial}{\partial \theta_j} \left[ \theta^T f(x_i, y_i) - \theta^T f(x_i, y^*) \right]
\]

\[
g_i = f_j(x_i, y_i) - f_j(x_i, y^*)
\]

In both cases: loss gradients want to make gold-standard features match predicted-structure features. Either you care about a distribution over all outputs ... or just the best output.
<table>
<thead>
<tr>
<th>approach</th>
<th>loss($x, y; h$)</th>
<th>training expense</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>generative models (3.3)</td>
<td>$-\log p_w(x, y)$</td>
<td>if multinomial-based, easy to train</td>
<td>can answer “many questions,” but the model must explain all evidence</td>
</tr>
<tr>
<td>globally normalized conditional models (3.5)</td>
<td>$-\log p_w(y</td>
<td>x) = -w^T g(x, y) + z_w(x)$</td>
<td>require inference for feature expectations and $z_w$</td>
</tr>
<tr>
<td>perceptron (3.6.2)</td>
<td>$-w^T g(x, y) + \max_{y' \in \mathcal{Y}_x} w^T g(x, y')$</td>
<td>only requires a decoder</td>
<td>no probabilistic interpretation or explicit regularization</td>
</tr>
<tr>
<td>large margin models (3.6)</td>
<td>$-w^T g(x, y) + \max_{y' \in \mathcal{Y}_x} w^T g(x, y') + \text{cost}(x, y', y)$</td>
<td>only require a cost-augmented decoder</td>
<td>incorporate cost function; no probabilistic interpretation</td>
</tr>
</tbody>
</table>

**Figure 3.3:** A comparison of the main learning methods discussed in this chapter. The form of the predictor $h(x)$ is assumed to be a linear decoder, $\arg\max_{y \in \mathcal{Y}_x} w^T g(x, y)$. 

Noah Smith, *Linguistic Structure Prediction*, page 107

link on course webpage
Averaging vs. early stopping

- Why does the perceptron keep flip flopping?
- This induces overfitting: cares too much about whatever it last saw
- Solution #1: early stopping
- Solution #2: averaging (or voting...)
  - Averaging seems to be the most popular: no fiddly hyperparameters to tune.
  - Perceptrons don’t allow a regularization term ... averaging is an alternate form of anti-overfitting control
- Avg. perceptron seems to be the most popular supervised struct. pred. algorithm for people who don’t care about machine learning and just want to do NLP. (“code to usefulness ratio”...)

Thursday, October 23, 14
Figure 1.2  Diagram of the relationship between naive Bayes, logistic regression, HMMs, linear-chain CRFs, generative models, and general CRFs.

From Sutton and McCallum tutorial on CRFs
Applications of sequence tagging
Document segmentation

38 files belonging to 7 UseNet FAQs
Tagging decisions are at the *line* level

Example:

```
<head> X-NNTP-Poster: NewsHound v1.33
<head> Archive-name: acorn/faq/part2
<head> Frequency: monthly
<head>
<question>2.6) What configuration of serial cable should I use?
<answer>
Here follows a diagram of the necessary connection
<answer> programs to work properly. They are as far as I know
<answer> agreed upon by commercial comms software developers fo
<answer>
<answer> Pins 1, 4, and 8 must be connected together inside
<answer> is to avoid the well known serial port chip bugs. The
```
Features in Experiments

begins-with-number
begins-with-ordinal
begins-with-punctuation
begins-with-question-word
begins-with-subject
blank
contains-alphanum
contains-bracketed-number
contains-http
contains-non-space
contains-number
contains-pipe
contains-question-mark
contains-question-word
ends-with-question-mark
first-alpha-is-capitalized
indented
indented-1-to-4
indented-5-to-10
more-than-one-third-space
only-punctuation
prev-is-blank
prev-begins-with-ordinal
shorter-than-30
# Results for FAQ segmentation

<table>
<thead>
<tr>
<th>Learner</th>
<th>Segmentation precision</th>
<th>Segmentation recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log. Reg.</td>
<td>0.038</td>
<td>0.362</td>
</tr>
<tr>
<td>TokenHMM</td>
<td>0.276</td>
<td>0.140</td>
</tr>
<tr>
<td>FeatureHMM</td>
<td>0.413</td>
<td>0.529</td>
</tr>
<tr>
<td>MEMM</td>
<td>0.867</td>
<td>0.681</td>
</tr>
</tbody>
</table>

Context important!

Features important!
Named Entity Recognition

The task is usually defined as:
identify segments in text that are names,
and some coarse types for them

SOCCER - [PER BLINKER] BAN LIFTED.
[LOC LONDON] 1996-12-06 [MISC Dutch] forward
[PER Reggie Blinker] had his indefinite suspension
lifted by [ORG FIFA] on Friday and was set to make
his [ORG Sheffield Wednesday] comeback against
[ORG Liverpool] on Saturday. [PER Blinker] missed
his club’s last two games after [ORG FIFA] slapped a
worldwide ban on him for appearing to sign contracts for
both [ORG Wednesday] and [ORG Udinese] while he was
playing for [ORG Feyenoord].

Figure 1: Example illustrating challenges in NER.
After meeting with the Denver Post Editorial Board, Virginia Lake traveled to Gile State Forest New Hampshire where she went camping with her daughter Anne’s Girl Scout Troop.

- State-of-the-art performance reported in the range 85 to 94% F-score (avg of prec/rec), depending on the annotated dataset http://www.aclweb.org/aclwiki/index.php?title=Named_Entity_Recognition_(State_of_the_art)
- But keep in mind your training data and features... http://nlp.stanford.edu:8080/corenlp/process

Who is seeing Skrillex today?
Application: Social media NLP

- Sequence models for online conversational text
- Why is online conversational text interesting or hard?

- Some material borrowed from Jacob Eisenstein
- Useful resource: Alan Ritter’s social media NLP course http://aritter.github.io/courses/5539.html
# A partial taxonomy of Twitter messages

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Official announcements</td>
<td>BritishMonarchy TheBritishMonarchy</td>
</tr>
<tr>
<td></td>
<td>On 6 Jan: Changing the Guard at Buckingham Palace - Starts at approx 11am <a href="http://www.royal.gov.uk/G">http://www.royal.gov.uk/G</a></td>
</tr>
<tr>
<td>Business advertising</td>
<td>bigdogcoffee bigdogcoffee</td>
</tr>
<tr>
<td></td>
<td>Back to normal hours beginning tomorrow...........Monday-Friday 6am-10pm Sat/Sun 7:30am-10pm</td>
</tr>
<tr>
<td>Links to blog and web content</td>
<td>crampell Catherine Rampell</td>
</tr>
<tr>
<td></td>
<td>Casey B. Mulligan: Assessing the Housing Sector - <a href="http://nyti.ms/hcUKK9">http://nyti.ms/hcUKK9</a></td>
</tr>
<tr>
<td>Celebrity self-promotion</td>
<td>THE_REAL_SHAQ THE_REAL_SHAQ</td>
</tr>
<tr>
<td></td>
<td>fill in da blank, my new years shaqalution is ____________</td>
</tr>
<tr>
<td>Status messages</td>
<td>emax electronic max</td>
</tr>
<tr>
<td></td>
<td>1.1.11 - britons and americans can agree on the date for once. happy binary day!</td>
</tr>
<tr>
<td>Group conversation</td>
<td>_siddx3 Evelyn Santana</td>
</tr>
<tr>
<td></td>
<td>RT @ LusciousVee: #EveryoneShouldKnow Ima Finally Be 18 This Year ^^^</td>
</tr>
<tr>
<td>Personal conversation</td>
<td>xoxoJuicyCee CeeCee ^</td>
</tr>
<tr>
<td></td>
<td>@txkmCelly aha kayy goodnight (:</td>
</tr>
</tbody>
</table>
Isn’t this “bad language”? 

- Text in computer-mediated communication (SMS, social media, IRC....) has shortenings, abbreviations, and grammar that’s very different than standard written English.
- Is it “bad language”?
- Why is it so different?
Are users illiterate?

- ... No.
Length limits?

• ... No.

<table>
<thead>
<tr>
<th>standard</th>
<th>length</th>
<th>alternative</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>your</td>
<td>85.1 ± 0.4</td>
<td>ur</td>
<td>81.9 ± 0.6</td>
</tr>
<tr>
<td>you’re</td>
<td>90.0 ± 0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>with</td>
<td>87.9 ± 0.3</td>
<td>wit</td>
<td>78.8 ± 0.7</td>
</tr>
<tr>
<td>going</td>
<td>82.7 ± 0.5</td>
<td>goin</td>
<td>72.2 ± 1.0</td>
</tr>
<tr>
<td>know</td>
<td>86.1 ± 0.4</td>
<td>kno</td>
<td>78.4 ± 1.0</td>
</tr>
<tr>
<td>about</td>
<td>88.9 ± 0.4</td>
<td>bout</td>
<td>74.5 ± 0.7</td>
</tr>
</tbody>
</table>

Table 1: Average length of messages containing standard forms and their shortenings
What do you see in conversations?

- Language use is socially contingent
  - Individual (every person has a dialect?)
  - Social groups
  - Gender
  - Socioeconomic background
  - Ethnicity
  - Geographic region....
Minority dialects/languages

• “A language is a dialect with an army and navy”
• Are minority languages/dialects “incorrect”?
  What accommodations are given to minority languages?
  • Ukrainian vs. Russian ...
  • African American dialects vs standard American English ...
• Descriptive linguistics vs. prescriptive grammarians
Social contingency of language
Social contingency of language
Social contingency of language
Social contingency of language
Social contingency of language
Social contingency of language

Figure 5: Geolocations for messages containing the words af (as fuck), ard (alright), ion (id o n ' t), lbvs (laughing but very serious), ctfu (cracking the fuck up), and the emoticon - (ambivalence or annoyance).
Social contingency of language

weeks 1–50

weeks 51–100

weeks 101–150

af

ikr
Alternate spellings

More remotely, _ard_ is an alternative spelling for _alright_, as in:

(4) @name ard let me kno
(5) lol (*laugh out loud*) u’ll be ard

Similarly, _brib_ is an alternative spelling for _crib_, which in turn signifies _home_.

(6) bbq (*barbecue*) at my fams (*family’s*) brib
(7) in da brib, just took a shower

• Nationally, _brib_ appears at a rate of once per 22,000 messages, which is roughly 5% as often as _crib_. But in the New York City area, _brib_ appears at a rate of once per 3,000 messages.
Final consonant dropping

Avg. Census demographics of counties in which users of each term live

Figure 1: Average demographics of the counties in which users of each term live, with 95% confidence intervals.
AAVE stressed 'been' on Twitter

From
From
The Second Great Migration, 1940–1970

Major Migration Corridors:
- South West to Midwest & Far West
- South Central to Midwest
- Southeast to Northeast

Map by Michael Siegel
Rutgers Cartography 2005


From
One interesting, but unexpected, finding is that the mobile phone (i.e., iPhone and Blackberry) clients have fewer out-of-vocabulary terms, on average, than the Web-based client. This suggests that either the users of the clients are less likely to misspell words or use slang terminology or that the clients may have better or more intuitive spell checking capabilities. A more thorough analysis is necessary to better understand the root cause of this phenomenon.

At the other end of the spectrum are the UberTwitter and Snaptu clients, which exhibit a substantially larger number of out-of-vocabulary terms. These clients are also typically used on mobile devices. As with our previous analysis, it is difficult to pinpoint the exact cause of such behavior, but we hypothesize that it is a function of user demographics and difficulties associated with inputting text on mobile devices.

<table>
<thead>
<tr>
<th>Client</th>
<th>% In-Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>88%</td>
</tr>
<tr>
<td>Twitter for iPhone</td>
<td>84%</td>
</tr>
<tr>
<td>Twitter for Blackberry</td>
<td>83%</td>
</tr>
<tr>
<td>Web</td>
<td>82%</td>
</tr>
<tr>
<td>UberTwitter</td>
<td>78%</td>
</tr>
<tr>
<td>Snaptu</td>
<td>73%</td>
</tr>
<tr>
<td>Overall</td>
<td>81%</td>
</tr>
</tbody>
</table>

Table 5: Percentage of in-vocabulary found in large English lexicon for different Twitter clients.
• Any NLP, starting with POS tagging, is going to require different models/resources than traditional written English

<table>
<thead>
<tr>
<th>ikr</th>
<th>smh</th>
<th>he</th>
<th>asked</th>
<th>fir</th>
<th>yo</th>
<th>last</th>
<th>name</th>
<th>so</th>
<th>he</th>
<th>can</th>
<th>add</th>
<th>u</th>
<th>on</th>
</tr>
</thead>
<tbody>
<tr>
<td>fb</td>
<td>lololol</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

On a PTB-trained tagging model:
http://nlp.stanford.edu:8080/corenlp/process
• How to make a new POS tagger?
  • Annotate some data
  • Train a supervised sequence tagger
  • Have good features
  • Use semi-supervised learning to leverage unlabeled data

• Two examples: POS for Twitter
  • Ritter et al. 2011 (UW Twitter NLP)
  • Gimpel et al. 2011, Owoputi et al. 2013 (ARK TweetNLP)
Just a little annotated data

<table>
<thead>
<tr>
<th></th>
<th>#Msg.</th>
<th>#Tok.</th>
<th>Tagset</th>
<th>Dates</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OCT27</strong></td>
<td>1,827</td>
<td>26,594</td>
<td>App. A</td>
<td>Oct 27-28, 2010</td>
</tr>
<tr>
<td><strong>NPSCHAT</strong></td>
<td>10,578</td>
<td>44,997</td>
<td>PTB-like</td>
<td>Oct–Nov 2006</td>
</tr>
<tr>
<td>(w/o sys. msg.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RITTERTw</strong></td>
<td>789</td>
<td>15,185</td>
<td>PTB-like</td>
<td>unknown</td>
</tr>
</tbody>
</table>

Table 1: Annotated datasets: number of messages, tokens, tagset, and date range. More information in §5, §6.3, and §6.2.
Features (MEMM tagger)

- Direct representations
  - Lexical identity
  - Shape features
  - Character n-gram prefix/suffix of word
- Regex detectors
  - Regex-based emoticon detectors
  - Regexes for hashtags, @-mentions
- Dictionary lookups
  - Traditional POS dictionary
  - Word clusters (next few slides)
- ...All of these at next/prev positions

- Does the algorithm matter?
  - First-order MEMM
  - Greedy decoding has same performance as Viterbi
    - Greedy decoding is 3 times faster, at least for us
  - CRF has slightly better performance (0.3% or so?)
private void initializeFeatureExtractors() throws IOException {
    allFeatureExtractors = new ArrayList<FeatureExtractorInterface>();

    allFeatureExtractors.add(new WordClusterPaths());
    allFeatureExtractors.add(new WordListFeatures.POSTagDict());
    allFeatureExtractors.add(new WordListFeatures.MetaphonePOSDict());

    allFeatureExtractors.add(new MiscFeatures.NgramSuffix(20));
    allFeatureExtractors.add(new MiscFeatures.NgramPrefix(20));
    allFeatureExtractors.add(new MiscFeatures.PrevWord());
    allFeatureExtractors.add(new MiscFeatures.NextWord());
    allFeatureExtractors.add(new MiscFeatures.WordformFeatures());

    allFeatureExtractors.add(new MiscFeatures.CapitalizationFeatures());
    allFeatureExtractors.add(new MiscFeatures.SimpleOrthFeatures());
    allFeatureExtractors.add(new MiscFeatures.PrevNext());

    allFeatureExtractors.add(new WordListFeatures.Listofnames("proper_names"));
    allFeatureExtractors.add(new WordListFeatures.Listofnames("celebs")); // 2012-c
    allFeatureExtractors.add(new WordListFeatures.Listofnames("videogame")); // jui
    allFeatureExtractors.add(new WordListFeatures.Listofnames("mobyplaces")); //
    allFeatureExtractors.add(new WordListFeatures.Listofnames("family"));
    allFeatureExtractors.add(new WordListFeatures.Listofnames("male"));
    allFeatureExtractors.add(new WordListFeatures.Listofnames("female"));

    allFeatureExtractors.add(new MiscFeatures.Positions());

    //allFeatureExtractors.add(new Prev2Words());
    //allFeatureExtractors.add(new Next2Words());
    //allFeatureExtractors.add(new MiscFeatures.URLFeatures());
}
Word clustering

- Unsupervised HMM to induce word classes. ("Brown clustering")
- Train on lots of unlabeled data
  - 56 M tweets, 847 M tokens
  - Compare to annotated data: 3000 tweets, 30k tokens
Word clustering

<table>
<thead>
<tr>
<th>Binary path</th>
<th>Top words (by frequency)</th>
</tr>
</thead>
<tbody>
<tr>
<td>111010100010</td>
<td>lmao lmfao lmaoo lmaooh haahahahaha lool ctfu rofl loool lmfaooo lmao0oo lmaoooo lmaooh lmaooh lmaooh haahahahaha hehe hahahahaha aha hehehe hahahahh aha hahaha hahahaha kk hahaa ah</td>
</tr>
<tr>
<td>111010100011</td>
<td>yes yep yup nope yess yesss yessss ofcourse yeap likewise yepp yesh yw yuup yu yeah yea nah naw yeahh nooo yeh noo noooo yeaa ikr nvm yeahhh nahn nooooo</td>
</tr>
<tr>
<td>111010100100</td>
<td>smh jk #fail #random #fact smfh #smh #winning #realtalk smdh #dead #justsaying</td>
</tr>
<tr>
<td>11101010111000</td>
<td>u yu yuh yhu uu yuu yew y0u yuhh youh yhuu iget hoy yooh yuo yue juu yue juu dya</td>
</tr>
<tr>
<td>11100101111001</td>
<td>w fo fa fr fro ov fer fir whit abou aft serie fore fah fuh w/her w/that fron isn agains</td>
</tr>
<tr>
<td>111101011000</td>
<td>facebook fb itunes myspace skype ebay tumblr bbm flickr aim msn netflix pandora</td>
</tr>
<tr>
<td>0011001</td>
<td>tryna gon finna bouta trynna boutta gne fina gonn tryina fenna qone trynaa qon</td>
</tr>
<tr>
<td>0011000</td>
<td>gonna gunna gona gna guna gnna ganna qonna gonna qon</td>
</tr>
<tr>
<td>0110110111</td>
<td>soo sooo sooooo soooooo sooooooooo soooooooooo sooooooooooo sooooooooooc</td>
</tr>
<tr>
<td>11101011001010</td>
<td>;) :p :-) xd ;-) ;d (; ;3 ;p =p :-p =)) ;] xdd #gno xddd &gt;:) ;-p &gt;:d 8-) ;-d</td>
</tr>
<tr>
<td>11101011001011</td>
<td>:) (:=) ;)) [: ♡ :-] ^<em>^ :))]) ^</em>[: ;]) (:( ^<em>^ (= ^</em>- :))))</td>
</tr>
<tr>
<td>11101011001111</td>
<td>:( :/ -<em>- -</em>- -:-:( :'( d: :] :s -<strong>- =(: /= &gt;.&lt; -</strong><em>- :-/ &lt;3 -_____ -;(; /: ;(:( &gt;</em>&lt; =</td>
</tr>
<tr>
<td>111010110001</td>
<td>&lt;3 ♥ xoxo &lt;33 xo &lt;333 ♥ ♥ #love s2 &lt;URL-twitition.com&gt; #neversaynever</td>
</tr>
</tbody>
</table>

http://www.ark.cs.cmu.edu/TweetNLP/cluster_viewer.html
Word clusters as features

- **smh**  #jk  #fail  #random  #fact  #smfh  #smh  #winning  #realtalk  #smdh  #dead  #justsaying

- w fo fa fr fro ov fer **fir** whit abou aft serie fore fah fuh w/her w/that fron isn agains

- yeah yea nah naw yeahh nooo yeh noo noooo yeaa **ikr** nvm yeahhh nahh nooooo

- **facebook**  **fb**  itunes  myspace  skype  ebay  tumblr  bbm  flickr  aim  msn  netflix  pandora

---

<table>
<thead>
<tr>
<th>ikr</th>
<th>smh</th>
<th>he</th>
<th>asked</th>
<th>fir</th>
<th>yo</th>
<th>last</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>O</td>
<td>V</td>
<td>P</td>
<td>D</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>so</th>
<th>he</th>
<th>can</th>
<th>add</th>
<th>u</th>
<th>on</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>P</td>
<td>O</td>
<td>V</td>
<td>V</td>
<td>O</td>
<td>P</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fb</th>
<th>lololol</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>!</td>
</tr>
</tbody>
</table>
Clusters help a lot

<table>
<thead>
<tr>
<th>Feature set</th>
<th>OCT27TEST</th>
<th>DAILY547</th>
<th>NPSCHATTEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>All features</td>
<td>91.60</td>
<td>92.80</td>
<td>91.19</td>
</tr>
<tr>
<td>with clusters; without tagdicts, namelists</td>
<td>91.15</td>
<td>92.38</td>
<td>90.66</td>
</tr>
<tr>
<td>without clusters; with tagdicts, namelists</td>
<td>89.81</td>
<td>90.81</td>
<td>90.00</td>
</tr>
<tr>
<td>only clusters (and transitions)</td>
<td>89.50</td>
<td>90.54</td>
<td>89.55</td>
</tr>
<tr>
<td>without clusters, tagdicts, namelists</td>
<td>86.86</td>
<td>88.30</td>
<td>88.26</td>
</tr>
<tr>
<td>Gimpel et al. (2011) version 0.2</td>
<td>88.89</td>
<td>89.17</td>
<td></td>
</tr>
<tr>
<td>Inter-annotator agreement (Gimpel et al., 2011)</td>
<td>92.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model trained on all OCT27</td>
<td></td>
<td></td>
<td>93.2</td>
</tr>
</tbody>
</table>

[Ablation tests: remove a feature class, check performance]

Figure 3: OCT27 development set accuracy using only clusters as features.
Clusters help for nonstandard terms

Table 3: DAILY547 accuracies (%) for tokens in and out of a traditional dictionary, for models reported in rows 1 and 3 of Table 2.