Brown Clusters

Many slides from Michael Collins
The Brown Clustering Algorithm

- Input: a (large) corpus of words
- Output 1: a partition of words into *word clusters*
- Output 2 (generalization of 1): a hierarchical word clustering
Example Clusters (from Brown et al, 1992)

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
June March July April January December October November September August
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
down backwards ashore sideways southward northward overboard aloft downwards adrift
water gas coal liquid acid sand carbon steam shale iron
great big vast sudden mere sheer gigantic lifelong scant colossal
man woman boy girl lawyer doctor guy farmer teacher citizen
American Indian European Japanese German African Catholic Israeli Italian Arab
pressure temperature permeability density porosity stress velocity viscosity gravity tension
mother wife father son husband brother daughter sister boss uncle
machine device controller processor CPU printer spindle subsystem compiler plotter
John George James Bob Robert Paul William Jim David Mike
anyone someone anybody somebody
feet miles pounds degrees inches barrels tons acres meters bytes
director chief professor commissioner commander treasurer founder superintendent dean cus-
todian
A Sample Hierarchy (from Miller et al., NAACL 2004)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Bit String</th>
</tr>
</thead>
<tbody>
<tr>
<td>lawyer</td>
<td>10000011010000</td>
</tr>
<tr>
<td>newspaperman</td>
<td>1000001101001000</td>
</tr>
<tr>
<td>stewardess</td>
<td>1000001101001010</td>
</tr>
<tr>
<td>toxicologist</td>
<td>1000001101000101</td>
</tr>
<tr>
<td>slang</td>
<td>1000001101001010</td>
</tr>
<tr>
<td>babysitter</td>
<td>1000001101011010</td>
</tr>
<tr>
<td>conspirator</td>
<td>1000001101011011</td>
</tr>
<tr>
<td>womanizer</td>
<td>1000001101011011</td>
</tr>
<tr>
<td>mailman</td>
<td>10000011010111</td>
</tr>
<tr>
<td>salesmen</td>
<td>1000001101011000</td>
</tr>
<tr>
<td>bookkeeper</td>
<td>10000011010110000</td>
</tr>
<tr>
<td>troubleshooter</td>
<td>10000011010110000</td>
</tr>
<tr>
<td>bouncer</td>
<td>10000011010110001</td>
</tr>
<tr>
<td>technician</td>
<td>10000011010110000</td>
</tr>
<tr>
<td>janitor</td>
<td>10000011010110010</td>
</tr>
<tr>
<td>saleswoman</td>
<td>1000001101011010</td>
</tr>
</tbody>
</table>

Table 1: Sample bit strings

Table 2: Feature Set

1. Tag + PrevTag
2. Tag + CurWord
3. Tag + CapAndNumFeatureOfCurWord
4. ReducedTag + CurWord
5. Tag + PrevWord
6. Tag + NextWord
7. Tag + DownCaseCurWord
8. Tag + Pref8ofCurrWord
9. Tag + Pref12ofCurrWord
10. Tag + Pref16ofCurrWord
11. Tag + Pref20ofCurrWord
12. Tag + Pref8ofPrevWord
13. Tag + Pref12ofPrevWord
14. Tag + Pref16ofPrevWord
15. Tag + Pref20ofPrevWord
16. Tag + Pref8ofNextWord
17. Tag + Pref12ofNextWord
18. Tag + Pref16ofNextWord
19. Tag + Pref20ofNextWord
The Intuition

- Similar words appear in similar contexts
- More precisely: similar words have similar distributions of words to their immediate left and right
The Formulation

- $\mathcal{V}$ is the set of all words seen in the corpus $w_1, w_2, \ldots w_n$
- Say $C : \mathcal{V} \rightarrow \{1, 2, \ldots k\}$ is a *partition* of the vocabulary into $k$ classes
- The model:

$$p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i | C(w_i))q(C(w_i) | C(w_{i-1}))$$

(note: $C(w_0)$ is a special start state)
An Example

\[ p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1})) \]
An Example

\[ p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1})) \]

\[ C(\text{the}) = 1, \quad C(\text{dog}) = C(\text{cat}) = 2, \quad C(\text{saw}) = 3 \]
An Example

\[ p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i | C(w_i)) q(C(w_i) | C(w_{i-1})) \]

\[ C(\text{the}) = 1, \quad C(\text{dog}) = C(\text{cat}) = 2, \quad C(\text{saw}) = 3 \]

\[ e(\text{the}|1) = 1, \quad e(\text{cat}|2) = e(\text{dog}|2) = 0.5, \quad e(\text{saw}|3) = 1 \]
An Example

\[ p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1})) \]

\[ C(\text{the}) = 1, \quad C(\text{dog}) = C(\text{cat}) = 2, \quad C(\text{saw}) = 3 \]

\[ e(\text{the}|1) = 1, \quad e(\text{cat}|2) = e(\text{dog}|2) = 0.5, \quad e(\text{saw}|3) = 1 \]

\[ q(1|0) = 0.2, \quad q(2|1) = 0.4, \quad q(3|2) = 0.3, \quad q(1|3) = 0.6 \]
An Example

\[ p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1})) \]

\[ C(\text{the}) = 1, \quad C(\text{dog}) = C(\text{cat}) = 2, \quad C(\text{saw}) = 3 \]

\[ e(\text{the}|1) = 1, \quad e(\text{cat}|2) = e(\text{dog}|2) = 0.5, \quad e(\text{saw}|3) = 1 \]

\[ q(1|0) = 0.2, \quad q(2|1) = 0.4, \quad q(3|2) = 0.3, \quad q(1|3) = 0.6 \]

\[ p(\text{the dog saw the cat}) = \]
The Brown Clustering Model

A Brown clustering model consists of:

- A vocabulary $\mathcal{V}$
- A function $C : \mathcal{V} \rightarrow \{1, 2, \ldots k\}$ defining a partition of the vocabulary into $k$ classes
- A parameter $e(v|c)$ for every $v \in \mathcal{V}$, $c \in \{1 \ldots k\}$
- A parameter $q(c'|c)$ for every $c'$, $c \in \{1 \ldots k\}$
Measuring the Quality of $C$

How do we measure the quality of a partition $C$?

$$\text{Quality}(C) = \sum_{i=1}^{n} \log e(w_i|C(w_i))q(C(w_i)|C(w_{i-1}))$$

$$= \sum_{c=1}^{k} \sum_{c'=1}^{k} p(c, c') \log \frac{p(c, c')}{p(c)p(c')} + G$$

where $G$ is a constant

Here

$$p(c, c') = \frac{n(c, c')}{\sum_{c,c'} n(c, c')} \quad p(c) = \frac{n(c)}{\sum_{c} n(c)}$$

where $n(c)$ is the number of times class $c$ occurs in the corpus, $n(c, c')$ is the number of times $c'$ is seen following $c$, under the function $C$. 

A First Algorithm

- We start with $|\mathcal{V}|$ clusters: each word gets its own cluster
- Our aim is to find $k$ final clusters
- We run $|\mathcal{V}| - k$ merge steps:
  - At each merge step we pick two clusters $c_i$ and $c_j$, and merge them into a single cluster
  - We greedily pick merges such that $\text{Quality}(C)$ for the clustering $C$ after the merge step is maximized at each stage
- Cost? Naive $= O(|\mathcal{V}|^5)$. Improved algorithm gives $O(|\mathcal{V}|^3)$: still too slow for realistic values of $|\mathcal{V}|$
A Second Algorithm

- Parameter of the approach is \( m \) (e.g., \( m = 1000 \))
- Take the top \( m \) most frequent words, put each into its own cluster, \( c_1, c_2, \ldots c_m \)
- For \( i = (m + 1) \ldots |V| \)
  - Create a new cluster, \( c_{m+1} \), for the \( i \)'th most frequent word. We now have \( m + 1 \) clusters
  - Choose two clusters from \( c_1 \ldots c_{m+1} \) to be merged: pick the merge that gives a maximum value for Quality\((C')\). We’re now back to \( m \) clusters
- Carry out \((m - 1)\) final merges, to create a full hierarchy

**Running time**: \( O(|V|m^2 + n) \) where \( n \) is corpus length
Abstract

We present a technique for augmenting annotated training data with hierarchical word clusters that are automatically derived from a large unannotated corpus. Cluster membership is encoded in features that are incorporated in a discriminatively trained tagging model. Active learning is used to select training examples. We evaluate the technique for named-entity tagging. Compared with a state-of-the-art HMM-based name finder, the presented technique requires only 13% as much annotated data to achieve the same level of performance. Given a large annotated training set of 1,000,000 words, the technique achieves a 25% reduction in error over the state-of-the-art HMM trained on the same material.

1 Introduction

At a recent meeting, we presented name-tagging technology to a potential user. The technology had performed well in formal evaluations, had been applied successfully by several research groups, and required only annotated training examples to configure for new name classes. Nevertheless, it did not meet the user's needs.

To achieve reasonable performance, the HMM-based technology we presented required roughly 150,000 words of annotated examples, and over a million words to achieve peak accuracy. Given a typical annotation rate of 5,000 words per hour, we estimated that setting up a name finder for a new problem would take four person days of annotation work – a period we considered reasonable. However, this user's problems were too dynamic for that much setup time. To be useful, the system would have to be trainable in minutes or hours, not days or weeks.

We left the meeting thinking about ways to reduce training requirements to no more than a few hours. It seemed that three existing ideas could be combined in a way that might reduce training requirements sufficiently to achieve the objective.

First were techniques for producing word clusters from large unannotated corpora (Brown et al., 1990; Pereira et al., 1993; Lee and Pereira, 1999). The resulting clusters appeared to contain a great deal of implicit semantic information. This implicit information, we believed, could serve to augment a small amount of annotated data. Particularly promising were techniques for producing hierarchical clusters at various scales, from small and highly specific to large and more general. To benefit from such information, however, we would need an automatic learning mechanism that could effectively exploit it.

Fortunately, a second line of recent research provided a potential solution. Recent work in discriminative methods (Lafferty et al., 2001; Sha and Pereira, 2003, Collins 2002) suggested a framework for exploiting large numbers of arbitrary input features. These methods seemed to have exactly the right characteristics for incorporating the statistically-correlated hierarchical word clusters we wished to exploit.

Combining these two methods, we suspected, would be sufficient to drastically reduce the number of annotated examples required. However, we also hoped that a third technique, active learning (Cohn et al., 1996;
At a recent meeting, we presented name-tagging technology to a potential user. The technology had performed well in formal evaluations, had been applied successfully by several research groups, and required only annotated training examples to configure for new name classes. Nevertheless, it did not meet the user's needs.
To achieve reasonable performance, the HMM-based technology we presented required roughly 150,000 words of annotated examples, and over a million words to achieve peak accuracy. Given a typical annotation rate of 5,000 words per hour, we estimated that setting up a name finder for a new problem would take four person days of annotation work – a period we considered reasonable. However, this user's problems were too dynamic for that much setup time. To be useful, the system would have to be trainable in minutes or hours, not days or weeks.
To implement discriminative training, we followed the averaged perceptron approach of (Collins, 2002). Our decision was based on three criteria. First, the method performed nearly as well as the currently best global discriminative model (Sha and Pereira, 2003), as evaluated on one of the few tasks for which there are any published results (noun phrase chunking). Second, convergence rates appeared favorable, which would facilitate multiple experiments. Finally, and most important, the method appeared far simpler to implement than any of the alternatives.

We implemented the averaged perceptron training algorithm exactly as described by Collins. However, we did not implement cross-validation to determine when to stop training. Instead, we simply iterated for 5 epochs in all cases, regardless of the training set size or number of features used. Furthermore, we did not implement features that occurred in no training instances, as was done in (Sha and Pereira, 2003). We suspect that these simplifications may have cost several tenths of a point in performance.

A set of 16 tags was used to tag 8 name classes (the seven MUC classes plus the additional null class). Two tags were required per class to account for adjacent elements of the same type. For example, the string Betty Mary and Bobby Lou would be tagged as PERSON-START PERSON-START NULL-START PERSON-START PERSON-CONTINUE.

Our model uses a total of 19 classes of features. The first seven of these correspond closely to features used in a typical HMM name tagger. The remaining twelve encode cluster membership. Clusters of various granularity are specified by prefixes of the bit strings. Short prefixes specify short paths from the root node and therefore large clusters. Long prefixes specify long paths and small clusters. We used 4 different prefix lengths: 8 bit, 12 bit, 16 bit, and 20 bit. Thus, the clusters decrease in size by about a factor of 16 at each level. The complete set of features is given in Table 2.

### Table 2: Feature Set

1. Tag + PrevTag
2. Tag + CurWord
3. Tag + CapAndNumFeatureOfCurWord
4. ReducedTag + CurWord
   //collapse start and continue tags
5. Tag + PrevWord
6. Tag + NextWord
7. Tag + DownCaseCurWord
8. Tag + Pref8ofCurrWord
9. Tag + Pref12ofCurrWord
10. Tag + Pref16ofCurrWord
11. Tag + Pref20ofCurrWord
12. Tag + Pref8ofPrevWord
13. Tag + Pref12ofPrevWord
14. Tag + Pref16ofPrevWord
15. Tag + Pref20ofPrevWord
16. Tag + Pref8ofNextWord
17. Tag + Pref12ofNextWord
18. Tag + Pref16ofNextWord
19. Tag + Pref20ofNextWord
Third, we consider the impact of active learning. Figure 3 shows (a) discriminative tagger performance without cluster features, (b) the same tagger using active learning, (c) the discriminative tagger with cluster features, and (d) the discriminative tagger with cluster features using active learning. Both with and without clusters, active learning exhibits a noticeable increase in learning rates. However, the increase in learning rate is significantly more pronounced when cluster features are introduced. We attribute this increase to better confidence measures provided by word clusters – the system is no longer restricted to whether or not it knows a word; it now can know something about the clusters to which a word belongs, even if it does not know the word.

Finally, Figure 4 shows the impact of consolidating the gains from both cluster features and active learning compared to the baseline HMM. This final combination achieves an F-score of 90 with less than 20,000 words of training – a quantity that can be annotated in about 4 person hours – compared to 150,000 words for the HMM – a quantity requiring nearly 4 person days to annotate. At 1,000,000 word of training, the final combination continues to exhibit a 25% reduction in error over the baseline system (because of limitations in the experimental framework discussed earlier, active learning can provide no additional gain at this operating point).

6 Discussion

The work presented here extends a substantial body of previous work (Blum and Mitchell, 1998; Riloff and Jones, 1999; Lin et al., 2003; Boschee et al, 2002; Collins and Singer, 1999; Yarowsky, 1995) that all focuses on reducing annotation requirements through a combination of (a) seed examples, (b) large un-annotated corpora, and (c) training example selection. Moreover, our work is based largely on existing techniques for word clustering (Brown et al., 1990), discriminative training (Collins 2002), and active learning. The synthesis of these techniques, nevertheless, proved highly effective in achieving our primary objective of reducing the need for annotated data. Much work remains to be done. In an effort to move rapidly toward our primary objective, we investigated only one type of discriminative training (averaged perceptron), only one type of clustering (bigram mutual information), and only one simple confidence measure for active learning. It seems likely that some additional gains could be realized by alternative discriminative methods (e.g. conditional random fields estimated with conjugate-gradient training). Similarly, alternative clustering techniques, perhaps based on different contextual features or different distance measures,
Third, we consider the impact of active learning. Figure 3 shows (a) discriminative tagger performance without cluster features, (b) the same tagger using active learning, (c) the discriminative tagger with cluster features, and (d) the discriminative tagger with cluster features using active learning. Both with and without clusters, active learning exhibits a noticeable increase in learning rates. However, the increase in learning rate is significantly more pronounced when cluster features are introduced. We attribute this increase to better confidence measures provided by word clusters – the system is no longer restricted to whether or not it knows a word; it now can know something about the clusters to which a word belongs, even if it does not know the word.

Finally, Figure 4 shows the impact of consolidating the gains from both cluster features and active learning compared to the baseline HMM. This final combination achieves an F-score of 90 with less than 20,000 words of training – a quantity that can be annotated in about 4 person hours – compared to 150,000 words for the HMM – a quantity requiring nearly 4 person days to annotate. At 1,000,000 word of training, the final combination continues to exhibit a 25% reduction in error over the baseline system (because of limitations in the experimental framework discussed earlier, active learning can provide no additional gain at this operating point).

Discussion

The work presented here extends a substantial body of previous work (Blum and Mitchell, 1998; Riloff and Jones, 1999; Lin et al., 2003; Boschee et al, 2002; Collins and Singer, 1999; Yarowsky, 1995) that all focuses on reducing annotation requirements through a combination of (a) seed examples, (b) large unannotated corpora, and (c) training example selection. Moreover, our work is based largely on existing techniques for word clustering (Brown et al., 1990), discriminative training (Collins 2002), and active learning.

The synthesis of these techniques, nevertheless, proved highly effective in achieving our primary objective of reducing the need for annotated data. Much work remains to be done. In an effort to move rapidly toward our primary objective, we investigated only one type of discriminative training (averaged perceptron), only one type of clustering (bigram mutual information), and only one simple confidence measure for active learning. It seems likely that some additional gains could be realized by alternative discriminative methods (e.g. conditional random fields estimated with conjugate-gradient training). Similarly, alternative clustering techniques, perhaps based on different contextual features or different distance measures,
Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters

Olutobi Owoputi*  Brendan O’Connor*  Chris Dyer*
Kevin Gimpel†  Nathan Schneider*  Noah A. Smith*

*School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
†Toyota Technological Institute at Chicago, Chicago, IL 60637, USA
Corresponding author: brenocon@cs.cmu.edu
### Binary path

| A1    | 111010100010   | Imao Imaoo Imaooo hahahahahahool clfu rofl loool Imfaoo Imfaooo Imaoooo Imbo lololol |
| A2    | 111010100011   | hahahahahhehe hahahahahahah ahah hehehehe ahaha hahahahahah ahahhahah kk hahaa ahah |
| A3    | 111010100100   | yes yep yep nope yess yesss yessss ofcourse likewise yepp yesss yy yuup yus |
| A4    | 111010101010    | yeah yea nah naw yeahh nooo yeh noo nooo yeeaa ikr rvm yeahhh nahnh nooooo |
| A5    | 11101011011100 | smh jk #fai# #random #fact smfh #smh #winning #realtalk smdh #dead #justsaying |
| B     | 0111010111 | u yu yuh yhu uu yuu yew y0u yuhh yhou yhuu iget yoy y0oh yuo ᴬ yue juu ᴻ dya youz you |
| C     | 11100101111100 | w fo fa fr fro ov fer fir whit abou aft serie fore faf fuh w/her w/that from isn again |
| D     | 1111010110000 | facebook fb itunes myspace skype ebay tumblr bbm flickr aim msn netflix pandora |
| E1    | 00110010 | tryna gon finna bouta trynna boutta gne fina gonn tryina fenn aone trynaa qon |
| E2    | 00110000 | gonna gunna gona gna guna gnnna ganna gonna gonnna gana qunn gonne goona |
| F     | 0110111011 | soo soo soo soo soooooo soooooooo soooooooos sooooooooos soooooooosos soooooooososso |
| G1    | 11101011001010 | ;( ;p :: xd ;-) ;d (; :3 ;p =p ;p =l) ;) xdd #gno xddd >;) ;p >:d 8-) ;d |
| G2    | 1110101100111 | ;( (: == )) ;j ] ( ⊘ :') =] ^_^ ;) ) ^ ^ [: ] :)) ) ( ( : ^_ ^ ( = ^_^ :) )) |
| G3    | 1110101100111 | :| '/ - - - - :-( ;'( d: ;l ;s -__ - =/ >,< - ____ :/ <3 ^-____ ;/: (( >_< =[ ; :fml |
| G4    | 1110101110001 | <3 ♥ xoxo <33 xo <333 ♥ <3 love s2 <URL-twitition.com> #neversaynever <3333 |
Owoputi et. al. ACL 2013
Owoputi et. al. NAACL 2013

<table>
<thead>
<tr>
<th>Feature set</th>
<th>OCT27 Test</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>90.00</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>93.2</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Tagging accuracies (%) in ablation experiments. OCT27 Test and Daily547 95% confidence intervals are roughly ±0.7%. Our final tagger uses all features and also trains on OCT27 Test, achieving 93.2% on Daily547.
Software for Brown Clusters

• Percy Liang’s code: https://github.com/percyliang/brown-cluster

• “A bit of progress in language modeling” Joshua Goodman
  • More sophisticated smoothing
  • JCLUSTER: http://research.microsoft.com/en-us/downloads/0183a49d-c86c-4d80-aa0d-53c97ba7350a/default.aspx