Tune Your Brown Clustering, Please

Leon Derczynski  
University of Sheffield  
leon@dcs.shef.ac.uk

Sean Chester  
Aarhus University  
schester@cs.au.dk

Kenneth S. Bøgh  
Aarhus University  
ksb@cs.au.dk

Abstract

Brown clustering, an unsupervised hierarchical clustering technique based on n-gram mutual information, has proven useful in many NLP applications. However, most uses of Brown clustering employ the same default configuration; the appropriateness of this configuration has gone predominantly unexplored. Accordingly, we present information for practitioners on the behaviour of Brown clustering in order to assist hyper-parametre tuning, in the form of a theoretical model of Brown clustering utility. This model is then evaluated empirically in two sequence labelling tasks over two text types. We explore the dynamic between the input corpus size, chosen number of classes, and quality of the resulting clusters, which has an impact for any approach using Brown clustering. In every scenario that we examine, our results reveal that the values most commonly used for the clustering are sub-optimal.

1 Introduction

Brown clustering (Brown et al., 1992) uses distributional information to group similar words. Unsupervised, it induces a hierarchical clustering over words to form a binary tree (e.g. Figure 1). This hierarchical clustering has recently been used in thousands of computational linguistics papers, often for feature generation. However, no work exists describing the behaviour and hyper-parametre tuning effects of Brown clustering; even the original paper concentrates on implementation rather than its behaviour. Except for a few forays off the beaten track (e.g. Christodoulopoulos et al. (2010), Owoputi et al. (2012), Derczynski et al. (2015a)), default parametres dominate; either 800 or 1000 Brown clusters are generated in nearly every published use. Few experiments use other configurations, and we are not aware of any prior work on hyper-parametre tuning for Brown clustering.

This paper addresses this information gap, providing practitioners with principled insights into the algorithm. We provide an analysis of how Brown clustering adds information over input, and, based on this, describe models for the effect that corpus size and cluster count have on the quality of results. These models are then tested in two sequence labeling tasks, cf. Qu et al. (2015). Finally, we compare the initial analysis to observations, leading to concrete advice for practitioners.

2 Background

Brown clustering uses mutual information to determine distributional similarity, placing similar words in the same cluster and similar clusters nearby in the binary tree. This is an unsupervised learned representation of language from the input corpus (Bengio et al., 2013). In the main implementation of Brown clustering (Liang, 2005), mutual information is measured at the bigram level. The resulting structure of word types can be used as feature representations in many NLP tasks, leading to quick, solid performance increases (Turian et al., 2010). In fact, as well as producing effective discriminative features, unsupervised hierarchical clusterings like Brown often lead to better taggers than models developed 20 years later (Blunsom and Cohn (2011), Owoputi et al. (2013)).
<table>
<thead>
<tr>
<th>Bit path</th>
<th>Word types</th>
</tr>
</thead>
<tbody>
<tr>
<td>00111001</td>
<td>can cn can caan canan shall ccan caaann ckaan</td>
</tr>
<tr>
<td>001011111001</td>
<td>ii id ion iv II iii ud wd uma ul idnt provoking hed 1+1 ididnt hast ine 2+2 idw #thingsblackpeopledo iii #onlywhitepeople dost doan uon apt-get</td>
</tr>
</tbody>
</table>

Table 1: Sample Brown clusters over English tweets. Each set of terms is a leaf in the hierarchy.

In practice, Brown clustering takes an input corpus $T$ and number of classes $c$, and uses mutual information to assign each term in the corpus vocabulary $V$ to one of the $c$ classes. Ideally, each class contains highly semantically-related words, by virtue of words being distributed according to their meaning (Wittgenstein, 1953). Each class is a leaf on an unbalanced binary tree. The path from the root to each leaf can be described as a bit string, where the $i$'th bit is 0 iff the path branches left at depth $i$ (e.g., you, I is on the path 01 in Figure 1). Brown clustering posits that leaves with longer common path prefixes are more semantically related. For example, in Figure 1, the cats, dog and you, I classes are more similar than either is to the love, pet class.

### 3 A Model for Brown Clustering

Here we outline our model for the behaviour of Brown clustering under various situations. Our goal is to describe how the number of classes, $c$, affects the quality of the resulting clustering.

Initial values for $c$ might not be appropriate for a given task or data set. Large values of $c$ risk forcing similar words into different classes, under-representing their similarity. Conversely, a small $c$ may cluster too coarsely, thereby reducing the discriminative power of resulting representations.

Brown clustering adds two forms of information: the agglomeration of terms into similar groups and the hierarchy connecting semantically similar groups of terms. At extreme values of $c$, little is added: if $c = |V|$, each word has its own class and only the hierarchy is added; if $c = 1$, one cluster contains all terms and information is gained from neither clustering nor a hierarchy. So, the information added by clustering increases with $c > 1$, peaks, and then declines towards $|V|$.

However, the information added solely by the hierarchy increases with $c$ and peaks when every word type has its own cluster, i.e., when $c = |V|$, as this gives the maximum number to the tree; we cannot add more leaves than there are word types (given a single root).

Also, a too-small $c$ may produce classes of unequal quality. Table 1 lists two classes derived in Owoputi et al. (2012), with $c = 1000$, on a large social media corpus. The first cluster agglomerates a set of semantically close lexemes, but the second cluster is internally semantically disparate, conflating many different concepts. This could indicate an inadequate value for $c$ that forces many concepts into a too-confined number of classes.

A $c$ exceeding the number of word types is also problematic: each word type should have only one class. This can arise in small datasets and when the vocabulary is particularly formalised (e.g., in a controlled natural language) (Wyner et al., 2010). Indeed, the size of the input dataset not only affects the number of eventual word types (Montemurro, 2001), but also quality of the classes.

For a fixed task and corpus genre, we hypothesise that each corpus size has an optimal $c$ and each $c$ has an optimal corpus size. When increasing a corpus size, new word types and further distributional information is revealed. The new distributional information leads to better-informed assignment of terms to classes, thereby improving the cluster quality. Eventually, however, the profusion of word types outgrows $c$ and semantically dissimilar words will be placed in the same class. Overall, we expect clustering performance to scale as shown in Figure 2: quality increases with $c$ to an optimal value, then dips slightly and levels off with some stochastic variance.

In fact, this behaviour has been observed (but not explained nor analysed) before. Owoputi et al. (2012) comment on the performance of a PoS tagger that for “different amounts of unlabeled tweets, keeping the number of clusters constant at 800 […] initially there was a logarithmic rela-
tion between number of tweets and accuracy, but from from 750 thousand to 56 million tweets, the tagging accuracy remained relatively constant.”

4 Method

Datasets We evaluate Brown tuning using two text types. For newswire, we use the Reuters RCV1 dataset (Rose et al., 2002). For social media, we draw randomly from a 10% sampling of tweets collected from 2009–2015, filtered for English using langid.py (Lui and Baldwin, 2012).

Preprocessing Drawing upon previous work (Turian et al., 2009; Owoputi et al., 2012), input data is preprocessed:

- Newswire data is cleaned per Liang (2005);
- Tabs, newlines and carriage returns are replaced with spaces;
- URLs are replaced with a <URL> marker;
- @Mentions are replaced by <Mention>; and
- Social media data has end-of-sequence markers <EOS> between tweets (see below).

Social media text was tokenised using the twokenize tool (O’Connor et al., 2010); newswire, with the Stanford tokenizer (Manning et al., 2014). The cleaning is the removal of any sentence where less than 90% of the characters are lowercase letters (excluding whitespace). This was not applied to tweets, as non-alphabet characters are markedly more frequent in social media text and an equivalent threshold is unclear. Cleaning has a notable effect on the RCV1 dataset, which has much potentially misleading non-text data such as numeric tables. Ultimately, $|T| = 1008.6c$ for 72.1M social media tweets. For newswire, $|T| = 114.8M$.

Terminals We note that Brown et al. (1992) assume a corpus long enough ($T \rightarrow \infty$) that the final term in Equation 1 tends to 1, and so $Pr(c_1|c_2)$ tends to the relative frequency of consecutive classes $c_1c_2$.

$$Pr(c_1|c_2) = \frac{C(c_1c_2)}{T} \times \frac{C(c_1)}{\sum_c C(c_1c)} \quad (1)$$

When corpora are composed of long, structured documents, bigrams are unlikely to cross the boundaries of unrelated sentences. However, in social media corpora there is little running discourse: each document is ≤ 140 characters and usually just one sentence. Running discourse only occurs when consecutive messages are from the same user and temporally ordered (or perhaps relate to a hashtag or conversation, which may be non-linear). Given the uniformity of Twitter sampling (Kergl et al., 2014), this continuity is unlikely. Therefore, we introduce an <EOS> marker after each tweet to break bigrams. This also captures some sentence position information.

5 Evaluation

The effect of class count ($c$) and corpus size (number of tokens, $|T|$) is measured extrinsically in two scenarios. Firstly, the generated clusters are used as a plug-in to the CMU Twitter part-of-speech tagger, replacing the supplied clusters and paths. This evaluation only covers social media. Secondly, the clusters are used to support feature generation in named entity recognition. This covers newswire and social media. The scenarios and corresponding evaluation measures are described below. Clusters are generated from all word types, even those that occur only once in the corpus.

Note critically that we aim to observe the performance sensitivity to input parametres, and to gain insights for tuning Brown clustering. Achieving new top scores in any task is not the goal.

5.1 Part of Speech Tagging

Owoputi et al. (2013) present a PoS tagger for tweets which relies on (among other features) Brown clusters. A reference clustering (and two evaluation datasets) is provided with the tagger, which we substitute with newly generated clusters. To observe the impact of tuning Brown clustering, we vary input parametres to produce new clusters and measure the tagger’s resultant tagging accuracy at token level. The “oct27” training and test splits are used.

5.2 Named Entity Recognition

We simplify NER to isolate the impact of $c$ and $|T|$. A CRF (Okazaki, 2007) is used to train and classify NER models. The only features are Brown cluster path prefixes of length [4,6,10,20] for newswire, as per Ratinov and Roth (2009), and [2,4,8,16] for newswire, as per Plank et al. (2014).

For newswire, we train and evaluate on the CoNLL data (Tjong Kim Sang and De Meulder, 2003) taking RCV clusters as input. For social media, we use the CRF with passive-aggressive updates to overcome some social media noise (Derczynski and Bontcheva, 2014), and train and eval-
As expected, extrinsic performance increases as number of classes \( c \) rises for a given corpus size \( |T| \), and also as \( |T| \) rises for a given \( c \), supporting our hypothesis that performance improves as \( c \) grows from 1. As \( c \) continues rising, word types are distributed more thinly across classes. Results show that performance levels off, and even begins to decrease (Table 2). In this experiment, we used an 8M token corpus and up to 10240 classes.

While this shows the effect of cluster quality decreasing when there are both too many and too few clusters, it does not approach the extreme value of \( c \) where there is one class for each word type. Thus, we ran another experiment varying cluster size but on a smaller corpus, which allowed examination of performance nearer to \( c = |V| \). For this, we took 62500 tokens of cleaned RCV1, which contained 9229 word types, and kept the same range of \( c \) values. The news genre (NW) was selected for two reasons: the larger evaluation set provides better resolution in results, and the reduced lexical variation means lower word type proliferation, giving more distributional information for the same data. Results are given in Table 2. The plateauing behaviour matches the predicted idealised performance curve in Figure 2 reasonably well. Note that the NER extrinsic evaluation relies more on hierarchical information than clustering, and so the drop in quality may be less pronounced than in other tasks.

For the social media data (SM), we observe unstable quality for large \( |T| \) (Table 3). This shows the point where too much data has been added and the classes have become noisy. Additional data for some \( |T| \) values is shown in Figure 3. As the noise is balanced by the addition of distributional information, we do not expect cluster quality to plummet rapidly, but rather hover; the data reflects this.

For PoS tagging, we see that there is a peak performance with \( c = 640 \), after which accuracy drops unstably (Table 4). This matches our expectations. In fact, the performance for \( c = 1000 \) (the value used to generate the original clusters for the CMU tagger) is a local minimum in our test. The PoS task involves a lot of other factors, and so is not as close an estimate of clustering quality as the NER task is, but it does make use of both the clusters and the hierarchy. No clear result came from varying \( |T| \) with a fixed \( c \) (Table 5), unlike in NER, where increasing \( |T| \) had a strong impact.

Some low values of \( c \) are particularly bad, espe-
specialized in the social media NER task, as in Table 2: with 40 or fewer classes, performance was consistently very low. This may be due to the smaller size of the SM evaluation set and high lexical variation in tweets, compared to newswire, where performance is also low but increases (sluggishly).

As expected, we see (for SM) that larger input corpora benefit from higher $c$. The default value gave sub-optimal results in every case.

The separation of cluster-path and class information (Tables 6, 7; Figures 4, 5) was revealing. In both cases, low values of $c$ give not static but worsening performance as $|T|$ rises (see e.g. the low-performance region in the lower back right of Figure 6). This is likely due to the effect $c$ has on determining the number of items considered for a merge at any point; as the input corpus grows, this “window” comprises an ever-decreasing proportion of available word types. Also, performance is more sensitive to increases in $c$ when $|T|$ is large, whereas increases under smaller $|T|$ are milder.

With the class-only experiment, performance peaks and then declines as $c \rightarrow |V|$, as expected (Section 3). The extreme class-only case, $c = |V|$, is one class per word, equivalent to a one-hot representation.

2 We do not use a minimum token frequency cutoff; if one

<table>
<thead>
<tr>
<th>Classes $c$</th>
<th>Oct27 TA</th>
<th></th>
<th>Classes $c$</th>
<th>Oct27 TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>62.9</td>
<td>640</td>
<td>80.0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>66.3</td>
<td>1'000</td>
<td>76.9</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>66.3</td>
<td>1'280</td>
<td>78.2</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>76.7</td>
<td>2'560</td>
<td>75.0</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>76.4</td>
<td>5'120</td>
<td>76.5</td>
<td></td>
</tr>
<tr>
<td>320</td>
<td>72.3</td>
<td>10'240</td>
<td>79.4</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: PoS token accuracy (TA), varying $c$ (8M tokens).

<table>
<thead>
<tr>
<th># tokens ($T$)</th>
<th>Oct27 TA</th>
<th># tokens ($T$)</th>
<th>Oct27 TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>8K</td>
<td>78.7</td>
<td>2M</td>
<td>77.1</td>
</tr>
<tr>
<td>16K</td>
<td>80.6</td>
<td>4M</td>
<td>79.1</td>
</tr>
<tr>
<td>32K</td>
<td>76.9</td>
<td>8M</td>
<td>76.9</td>
</tr>
<tr>
<td>62.5K</td>
<td>79.5</td>
<td>16M</td>
<td>68.7</td>
</tr>
<tr>
<td>125K</td>
<td>79.5</td>
<td>32M</td>
<td>74.6</td>
</tr>
<tr>
<td>250K</td>
<td>76.4</td>
<td>64M</td>
<td>77.0</td>
</tr>
<tr>
<td>500K</td>
<td>76.1</td>
<td>125M</td>
<td>73.8</td>
</tr>
<tr>
<td>1M</td>
<td>72.6</td>
<td>250M</td>
<td>74.2</td>
</tr>
</tbody>
</table>

Table 5: PoS token accuracy, varying corpus size ($c = 1000$).

---

Figure 4: Using decomposed class prefixes, without cluster ID, for paths-only features.

Figure 5: Using Brown class / cluster ID as sole feature. A 3D plot of these data points and others is shown in Figure 6.
Figure 6: 3D plot of F1 using only cluster information, varying $T$ and $c$. Interactive version at http://derczynski.com/sheffield/brown-tuning/.

Performance increases with both $|T|$ and $c$. The advice here is that if $|T|$ is easier to increase than waiting for a large $c$, then get the big corpus first.

The best possible $c$ behaves oppositely with class-only and path-only information. For class-only, with small corpora, $c$ should be high (or set to $|V|$); as the corpora grow, so the best $c$ levels off (Table 7). Conversely, for path-only, small corpora benefit from lower $c$, whereas larger corpora do better with high values of $c$ (Table 6). This is because as $c \to |V|$, more path information is added, whereas clustering information decreases, as suggested in Section 3.

To exploit high values of $c$ when $|T|$ is substantial, path features are required. Further, it may be more efficient to try a lower $c$ and a larger $|T|$. In scenarios where the clusters are more important than hierarchical information, choosing too high a value for $c$ is both expensive and risky.

Default values of $c$ are unlikely to perform well, and are often even local minima in performance. Note that performance does not increase monotonically with either $|T|$ or $c$; this is likely due to poor decisions being made by the algorithm based on the information available at the time under those parameters. As a different tree is generated for every different corpus and class count, and these tree vary almost chaotically across text types and corpus sizes, and also as performance depends on how features are extracted, it is unlikely that a universal formula for selecting $c$ exists. Ceteris paribus, it is reasonable to start finding $c$ through random search (Bergstra and Bengio, 2012) beta-weighted against high $c$ to reduce computation costs (Micenková et al., 2015) and against very low $c$ where extrinsic performance is poor; e.g., something like $c \sim B(\alpha = 1.5, \beta = 5)c_{\text{max}}$, with $c_{\text{max}}$ in the order of $10^5$, based on $|T|$ and our results in both text types.

Supplementary to this paper, we provide many clusters and paths for the two common text types investigated, to help researchers start exploring Brown parameter space for their problem for some values of $c$, thus deferring the initial large computational costs of running this algorithm.

7 Conclusion

As a community, if Brown clustering is to continue its adoption in so many NLP tasks, we need methods to choose appropriate values for its hyper-parameters. We presented our model of how Brown clustering quality changes depending on its input and tuning. This model was supported in an empirical evaluation.

The target number of classes $c$ has an impact on the utility of the classes. The corpus size $|T|$ also has an impact.

Setting $c$ too low clusters too coarsely; setting it too high forces similar words to be split across clusters. Similarly, a preset $c$ will not be optimal for ever-increasing corpus sizes: just adding more data will eventually make no difference or even reduce cluster quality. We therefore strongly recommend avoiding the default value of $c = 1000$, and instead finding values which fully activate this powerful hierarchical clustering technique.

Acknowledgments

This research has received funding from the EU’s seventh Framework Programme for research, technological development and demonstration under grant agreement No. 611233, PHEME, and from the WALLViz project, supported by the Danish Council for Strategic Research, grant 10-092316.

References


4http://www.pheme.eu/
5http://wallviz.dk/