Brown Clustering

Alla Rozovskaya

Some slides are taken from Julia Hockenmaier and Michael Collins

Word representations

- Unsupervised clustering of similar words from a large corpus

The Brown et al. Word Clustering

- The Brown clustering algorithm
  - An unsupervised learning algorithm
    - Take as input unlabeled corpus of text
  - Outputs very useful representations of individual words

The Brown Clustering Algorithm

- Input: a large corpus of words
- Output 1: a partition of words into word clusters
  - Similar words appear in similar clusters
- Output 2: 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 custodian

Brown Clustering

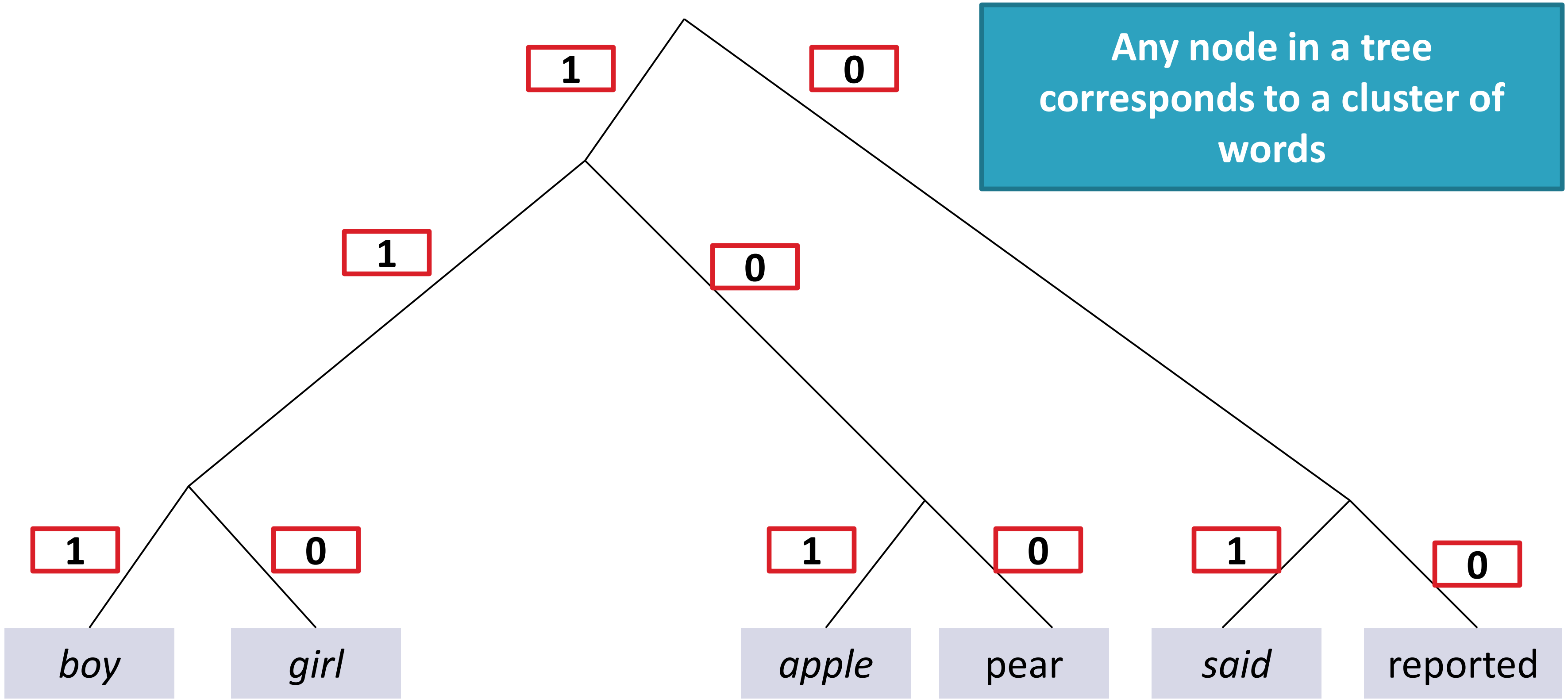
- This kind of word clusters can be very useful in a wide range of NLP applications
- Now for every word we have additional knowledge about what class of words it falls into

Brown Clustering representations

- In addition to word clusters, Brown Clustering can produce hierarchical representations

Tree representation

- [boy,girl,apple,pear,said,reported]



Clusters and bitstrings

- Boy 111
- Girl 110
- Apple 101
- Pear 100
- Said 01
- Reported 00

❑ **Common prefixes** correspond to clusters

A Sample Hierarchy (Miller et al. ‘04)

lawyer	1000001101000
newspaperman	100000110100100
stewardess	100000110100101
toxicologist	10000011010011
slang	1000001101010
babysitter	100000110101100
conspirator	1000001101011010
womanizer	1000001101011011
mailman	10000011010111
salesman	100000110110000
bookkeeper	1000001101100010
troubleshooter	10000011011000110
bouncer	10000011011000111
technician	1000001101100100
janitor	1000001101100101
saleswoman	1000001101100110
...	
Nike	1011011100100101011100
Maytag	10110111001001010111010
Generali	10110111001001010111011
Gap	1011011100100101011110
Harley-Davidson	10110111001001010111110
Enfield	101101110010010101111110
genus	101101110010010101111111
Microsoft	10110111001001011000
Ventritex	101101110010010110010
Tractebel	1011011100100101100110
Synopsys	1011011100100101100111
WordPerfect	1011011100100101101000

- Brown clustering can produce a **hierarchy;**
- Allows clustering with different levels of **granularity**
- Each word in the vocabulary can be assigned a **bitstring**
- All words with the same bitstring prefix belong to the same cluster

Can be especially useful when dealing with **rare words** (e.g. name cluster for NER)

The Brown Clustering Algorithm

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

Intuition

- Similar words appear in similar contexts
  - Similar words have similar distributions of words to their immediate left and right
- E.g. ‘the’, ‘a’
  - Left contexts: in, of, …
  - Right contexts: nouns (dog, cat, table, etc.)
- E.g. ‘Monday’, ‘Tuesday’
  - Left contexts: last, on, etc.
  - Right contexts: …

The formulation

- V is the set of all words in the corpus w_1, w_2, …w_T
- Output is a function C: V → {1,2,…k} is a *partition* of the vocabulary into *k* classes (clusters)

The formulation

- The model

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

- Two types of parameters (similar to a bigram HMM):
  - **Emission parameters** $e(w_i \mid C(w_i))$
  - **Transition parameters** $q(C(w_i) \mid C(w_{i-1}))$

- So the model consists of function C that maps every word in the vocabulary to the set {1,2,…k} and the emission and transition parameters

- It's very similar to HMM but **C is deterministic**:
  - Each word is mapped to exactly one state

The formulation

- Given the parameters, we can express the probability of our corpus as follows:

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

- This is a bigram model $p(w_i \mid w_{i-1})$

$$p(w_i \mid w_{i-1}) = e(w_i \mid C(w_i)q(C(w_i) \mid C(w_{i-1}))$$

The formulation

- The model

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

$$\log p(w_1, w_2, \ldots w_T) = \sum_{i=1}^{n} \log e(w_i \mid C(w_i))q(C(w_i) \mid C(w_{i-1}))$$

An example

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

V={cat,dog,saw,the}

C(the)=1, C(dog)=C(cat)=2, C(saw)=3

e(the|1)=1, e(cat|2)=e(dog|2)=0.5, e(saw|3)=1

q(1|0)=0.2, q(2|1)=0.4, q(3|2)=0.3, q(1|3)=0.6

Find p(the dog saw the cat)

Question

- Say we're given a deterministic class function C: C(dog)=1 C(man)=2 C(woman)=2 C(walk)=3
- Which of the following are possible definitions of e for a model with this class function?
  - (1) e(dog|1)=1, e(man|2)=0.5, e(woman|2)=0.5, e(walk|3)=1
  - (2) e(dog|1)=1, e(man|2)=1,e(woman|3)=0.5, e(walk|3)=0.5
  - (3) e(dog|1)=1,e(man|2)=0.5,e(woman|2)=0.5,e(man|3)=0.5, e(walk|3)=0.5
  - (4) e(dog|1)=1,e(man|2)=0.9,e(woman|2)=0.1,e(walk|3)=1.0

Answer: (1) and (4) are valid

The Brown Clustering Model

- A Brown clustering model consists of:
  - A vocabulary V
  - A function C: V→ {1,2,…k} defining a partition of the vocabulary into k classes
  - A parameter e(v|c) for every $v \in V$, $c \in \{1 \ldots k\}$
  - A parameter q(c'|c) for every c', $c \in \{1 \ldots k\}$
- How do take a training corpus as input and produce C, e(v|c), and q(c'|c) as output?

Measuring the Quality of C

- Given our training corpus w_1, w_2, … w_n, **Quality(C)** is a measure of how well partition C fits our training corpus:

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

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

where G is a constant

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

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

Finding partition C: Algorithm 1

- Start with |V| clusters: each word gets its own cluster
- Our goal is to find *k* clusters
- Run |V|-k merge steps:
  - At each step, pick 2 clusters c_i and c_j and merge them into one cluster
  - Greedily pick merges such that Quality(C) for the clustering C after the merge is maximized
- This is inefficient

Finding partition C: Algorithm 2

- Parameter m (e.g. m=1000)
- Take the top m most frequent words, put each into its own cluster, c_1, c_2, … c_m
- For i = (m+1) … |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 …c_{m+1} to be merged: pick the merge that gives a maximum value for Quality(C). We now have m clusters
- Carry out (m-1) final merges, to create a full hierarchy

Brown Clustering and Named Entity Recognition

Name Tagging with Word Clusters and Discriminative Training

Scott Miller, Jethran Guinness, Alex Zamanian
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
szmiller@bbn.com

Named Entity recognition

INPUT:

Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:

Profits soared at [ORG Boeing Co.] , easily topping forecasts on [LOC Wall Street], as their CEO [PER Alan Mulally] announced first quarter results.

Named Entity recognition

- Can be mapped to a tagging problem

Profits/NA soared/NA at/NA Boeing/S-ORG Co./C-ORG ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/S-LOC Street/C-LOC ,/NA as/NA their/NA CEO/NA Alan/S-PER Mulally/S-PER announced/NA first/NA quarter/NA results/NA ./NA

Miller et al., NAACL'04

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.

Miller et al., NAACL'04

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.

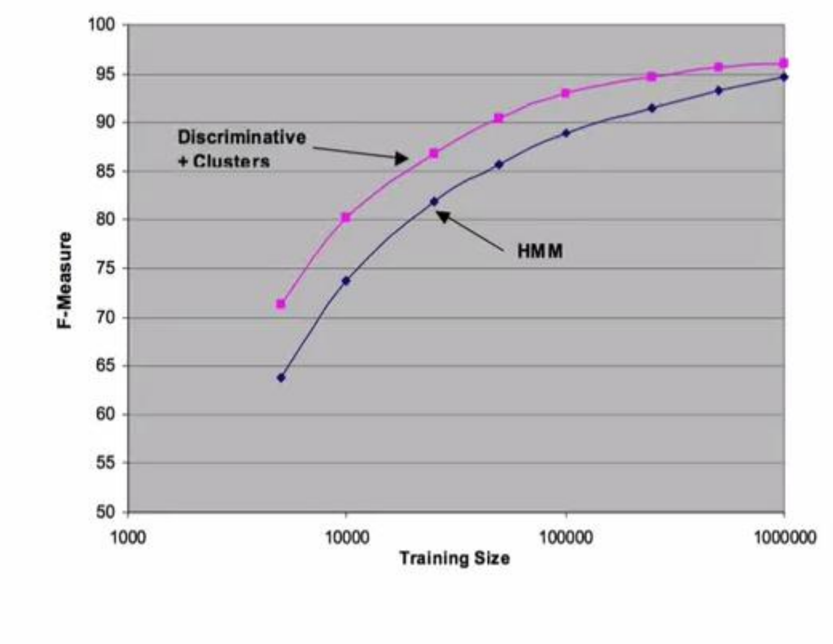
Representation in log-linear tagger: Histories

- A history is a 4-tuple $< t_{-2}, t_{-1}, w_{[1:n]}, i >$
- t_{-2}, t_{-1} are the previous two tags
- $w_{[1:n]}$ are the n words in the input sentence
- i is the index of the word being tagged
- X is the set of all possible histories

Miller et al., NAACL'04

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

Results on NER (Miller et al.)



F-Measure=2*Precision*Recall/(Precision+Recall)

Results on NER (Miller et al.)