CS 6120/CS 4120: Natural Language Processing

Instructor: Prof. Lu Wang
College of Computer and Information Science
Northeastern University
Webpage: www.ccs.neu.edu/home/luwang

Logistics

- Reminder for late day usage:
  - "Each student has a budget of 5 days throughout the semester before a late penalty is applied."
  - No need to inform TAs about late submission.
  - For assignments, we will start grading one week after the deadline. Let us know on piazza if you plan to submit later than that.
  - Grace period of one hour is given.
- NO CLASS next Tuesday (instructor out of town for academic meetings). Quiz will be on next **Friday**.
  - See schedule at http://www.ccs.neu.edu/home/luwang/courses/cs6120_sp2019/cs6120_sp2019.html

Brown Clusters

Brown Clusters -- Unsupervised

- Goal
  - To learn about regularities in words
  - By clustering words into groups

Example Clusters

- 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
- liberal conservative parliamentary royal progressive
- Tory provisional separatist federalist PQ

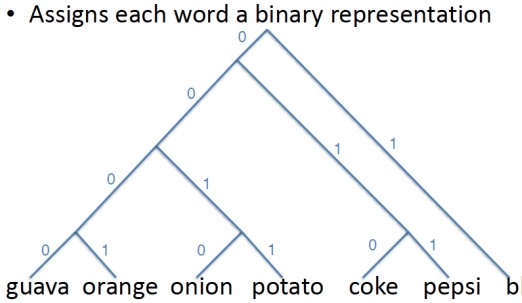
Brown Clustering Algorithm

- Input: a (large) corpus of words
- Output 1: a partition of words into word clusters
- Output 2 (generalization of 1): a hierarchical word clustering

- Assigns each word a binary representation

guava orange onion potato coke pepsi blue

- Assigns each word a binary representation



guava orange onion potato coke pepsi blue

- onion: 0010

- Different prefix lengths: different abstractions

111111110110000	slapped
111111110110000	shattered
111111110110000	commissioned
111111110110000	drafted
111111110110000	authorized
111111110110000	authorised
111111110110000	imposed
111111110110000	established
111111110110000	developed

111111111100110	officer
111111111100110	acquaintance
111111111100110	policymaker
111111111100110	instructor
111111111100110	investigator
111111111100110	advisor
111111111100110	aide
111111111100110	expert
111111111100110	adviser

111110100	Clinton
111110100	Aleman
111110100	Zeroual
111110100	Sampras
111110100	Barzani
111110100	Cardoso
111110100	Kim
111110100	King
111110100	Saddam
111110100	Netanyahu
111110100	Dole

111111100	Bill
111111100	Boris
111111100	Warren
111111100	Fidel
111111100	Yasser
111111100	Kenneth
111111100	Viktor
111111100	Benjamin
111111100	Jacques
111111100	Bob
111111100	Alexander

Intuition

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

eat	juice
grow	seeds
...	...

guava

eat	juice
grow	seeds
...	...

orange

Formulation

- *V* is the set of all words seen in the corpus
- Say C: V → {1, 2,...k} is a partition of the vocabulary into k classes (k ~ 1000)
- The model: ($C(w_0)$ is a special <s> state)

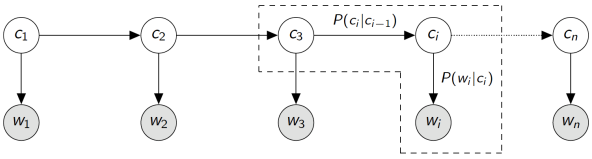
$$p(w_1, w_2, ..., w_N)_{\text{Corpus}} = \prod_{t=1}^{N} e(w_t \mid C(w_t)) q(C(w_t) \mid C(w_{t-1}))$$

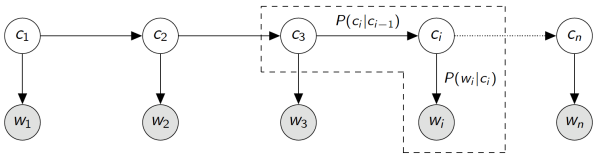
Formulation

- *V* is the set of all words seen in the corpus
- Say C: V → {1, 2,...k} is a partition of the vocabulary into k classes (k ~ 1000)
- The model: ($C(w_0)$ is a special <s> state)

$$p(w_1, w_2, ..., w_N) = \prod_{t=1}^{N} e(w_t \mid C(w_t)) q(C(w_t) \mid C(w_{t-1}))$$

Corpus — *Emission Probability!* — *Transition Probability!*





Difference from HMM: each word can be labeled as *only* one class!

C(I)=1, C(ate)=C(drank)=2
C(guava)=C(pepsi)=3, C(and)=4

e(I|1)=1, e(ate|2)= e(drank|2)= 0.3
e(guava|3)=e(pepsi|3)=0.1, e(and|4)=1
q(1|0)=0.2, q(2|1)=0.4, q(3|2)=0.3, q(4|3)=0.1, q(2|4)=0.2

C(I)=1, C(ate)=C(drank)=2
C(guava)=C(pepsi)=3, C(and)=4

e(I|1)=1, e(ate|2)= e(drank|2)= 0.3
e(guava|3)=e(pepsi|3)=0.1, e(and|4)=1
q(1|0)=0.2, q(2|1)=0.4, q(3|2)=0.3, q(4|3)=0.1, q(2|4)=0.2

P(I ate guava and drank pepsi) =

0.2*1*0.4*0.3*0.3*0.1*0.1*1*0.2*0.3*0.3*0.1

The Model

- Vocabulary *V*
- A function C: *V* → {1..k}
  - partitioning of vocabulary into k classes
- Emission probabilities e(w|C(w))
- Transition probability q(c'|c)

Scoring a Partition: **Quality (C)**

N is the number of words in the corpus

$$\frac{1}{N}\sum_{t=1}^{N}\log\left(e(w_t \mid C(w_t))\,q(C(w_t) \mid C(w_{t-1}))\right)$$

c and c' are in {1, 2, ..., k}

$$= \sum_{c,c'} p(c,c')\log\left(\frac{p(c,c')}{p(c)p(c')}\right) + \sum_{w} p(w)\log p(w)$$

n(c): #occurences of c in corpus under function C
n(c,c'): #occurences of (c,c') in corpus under function C

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

Scoring a Partition: **Quality (C)**

N is the number of words in the corpus

$$\frac{1}{N}\sum_{t=1}^{N}\log\left(e(w_t \mid C(w_t))\,q(C(w_t) \mid C(w_{t-1}))\right)$$

$$= \sum_{c,c'} p(c,c')\log\left(\frac{p(c,c')}{p(c)p(c')}\right) + \sum_{w} p(w)\log p(w)$$

Mutual information (MI) **constant**

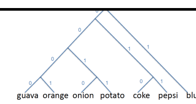
n(c): #occurences of c in corpus under function C
n(c,c'): #occurences of (c,c') in corpus under function C

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

Proof (not required)

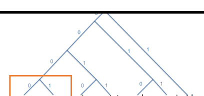
$$\begin{aligned}
\text{Quality}(C) &= \frac{1}{n}\sum_{i=1}^{n}\log P(C(w_i)|C(w_{i-1}))P(w_i|C(w_i)) \\
&= \sum_{w,w'}\frac{n(w,w')}{n}\log P(C(w')|C(w))P(w'|C(w')) \\
&= \sum_{w,w'}\frac{n(w,w')}{n}\log\frac{n(C(w),C(w'))}{n(C(w))}\frac{n(w')}{n(C(w'))} \\
&= \sum_{w,w'}\frac{n(w,w')}{n}\log\frac{n(C(w),C(w'))n}{n(C(w))n(C(w'))} + \sum_{w,w'}\frac{n(w,w')}{n}\log\frac{n(w')}{n} \\
&= \sum_{c,c'}\frac{n(c,c')}{n}\log\frac{n(c,c')n}{n(c)n(c')} + \sum_{w'}\frac{n(w')}{n}\log\frac{n(w')}{n}
\end{aligned}$$

A First (Naïve) Algorithm



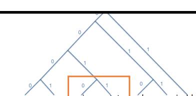
- Start with |*V*| clusters: each word gets its own cluster
- Our aim is to find k final clusters
- We run |*V*| - k merge steps:
  - At each merge step we pick two clusters c_i and c_j, and merge them into a single cluster
  - We greedily pick merges such that Quality(C) for the clustering C after the merge step is maximized at each stage

A First (Naïve) Algorithm



- Start with |*V*| clusters: each word gets its own cluster
- Our aim is to find k final clusters
- We run |*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 Quality(C) for the clustering C after the merge step is maximized at each stage

A First (Naïve) Algorithm



- Start with |*V*| clusters: each word gets its own cluster
- Our aim is to find k final clusters
- We run |*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 Quality(C) for the clustering C after the merge step is maximized at each stage

A First (Naïve) Algorithm

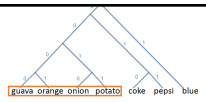
- Cost?
  - Naive = $O(|V|^5)$. *Calculate everything on-the-fly!*
  - Improved algorithm gives $O(|V|^3)$ *Store word transitions!*

A First (Naïve) Algorithm

- Cost?
  - Naive = $O(|V|^5)$. *Calculate everything on-the-fly!*
  - Improved algorithm gives $O(|V|^3)$ *Store word transitions!*

Too slow!

A Second Algorithm

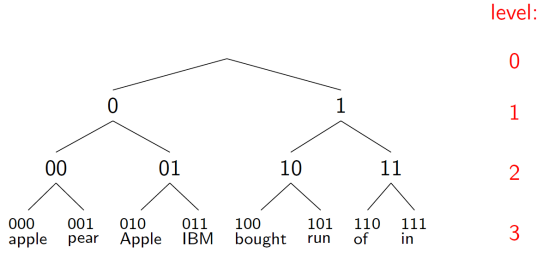


- New parameter: m (e.g., m = 1000)
- Take the top m most frequent words, put each into its own cluster, $c_1, c_2, \ldots c_m$
- For i = (m + 1) ... |V|
  - Create a new cluster, c_{m+1} , for the i'th most frequent word. We now have m + 1 clusters
- Choose two clusters from $c_1 \ldots c_{m+1}$ to 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

A Second Algorithm

- Running time: $O(|V|m^2 + n)$ where n is corpus length

A perfect balanced binary tree



In reality: