Lecture 10: Discriminative Training for MT/ the Brown et al. Word Clustering Algorithm

Michael Collins

April 6, 2011

Discriminative Training for MT

- Our original model:

$$f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta \times \delta(t(p_k), s(p_{k+1}))$$

- A discriminative model for translation (Liang et al., 2006):

$$f(y; \underline{w}, \alpha, \eta) = \alpha \times h(e(y)) + \sum_{k=1}^{L} \underline{w} \cdot \underline{\phi}(p_k) + \sum_{k=1}^{L-1} \eta \times \delta(t(p_k), s(p_{k+1}))$$

  Here $\alpha \in \mathbb{R}$, $\eta \in \mathbb{R}$ and $\underline{w} \in \mathbb{R}^d$ are the parameters of the model

- Crucial idea: $\underline{\phi}(p)$ is a feature-vector representation of a phrase p

The Learning Set-up

- Our training data consists of $(x^{(i)}, e^{(i)})$ pairs, for $i = 1 \dots n$, where $x^{(i)}$ is a source language sentence, and $e^{(i)}$ is a target language sentence

- We use $\mathcal{Y}^{(i)}$ to denote the set of possible derivations for $x^{(i)}$

- A complication: for a given $(x^{(i)}, e^{(i)})$ pair, there may be many derivations $y \in \mathcal{Y}^{(i)}$ such that $e(y) = e^{(i)}$.

A "Bold Updating" Algorithm from Liang et al.

- Initialization: set $\underline{w} = 0$, $\alpha = 1$, $\eta = -1$
- for $t = 1 \ldots T$, for $i = 1 \ldots n$,
  - $y^* = \arg\max_{y \in \mathcal{Y}^{(i)} : e(y) = e^{(i)}} f(y; \underline{w}, \alpha, \eta)$
  - $z^* = \arg\max_{z \in \mathcal{Y}^{(i)}} f(z; \underline{w}, \alpha, \eta)$
  - For any phrase $p \in y^*$, $\underline{w} = \underline{w} + \underline{\phi}(p)$
  - For any phrase $p \in z^*$, $\underline{w} = \underline{w} - \underline{\phi}(p)$
  - Set $\alpha = \alpha + h(e(y^*)) - h(e(z^*))$
  - Set $\eta = \eta + \ldots - \ldots$

A "Local Updating" Algorithm from Liang et al.

- Initialization: set $\underline{w} = 0$, $\alpha = 1$, $\eta = -1$
- for $t = 1 \ldots T$, for $i = 1 \ldots n$,
  - Define N^i to be the k highest scoring translations in $\mathcal{Y}^{(i)}$ under $f(y; \underline{w}, \alpha, \eta)$ (easy to generate N^i using k-best search)
  - y^* is member of N^i that is "closest" to $e^{(i)}$.
  - $z^* = \arg\max_{z \in \mathcal{Y}^{(i)}} f(z; \underline{w}, \alpha, \eta)$
  - For any phrase $p \in y^*$, $\underline{w} = \underline{w} + \underline{\phi}(p)$
  - For any phrase $p \in z^*$, $\underline{w} = \underline{w} - \underline{\phi}(p)$
  - Set $\alpha = \alpha + h(e(y^*)) - h(e(z^*))$
  - Set $\eta = \eta + \ldots - \ldots$

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 hierarchichal 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

liberal conservative parliamentary royal progressive Tory provisional separatist federalist PQ

had hadn't hath would've could've should've must've might've

asking telling wondering instructing informing kidding reminding bothering thanking deposing

that tha theat

head body hands eyes voice arm seat eye hair mouth

A Sample Hierarchy (from Miller et al., NAACL 2004)

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
....	
John	101110010000000000
Consuelo	101110010000000001
Jeffrey	101110010000000010
Kenneth	10111001000000001100
Phillip	101110010000000011010
WILLIAM	101110010000000011011
Timothy	10111001000000001110
Terrence	101110010000000011110
Jerald	101110010000000011111
Harold	101110010000000100
Frederic	101110010000000101
Wendell	10111001000000011

Table 1: Sample bit strings

The Formulation

- $\mathcal{V}$ is the set of all words seen in the corpus $w_1, w_2, \ldots w_T$
- Say $n(w, v)$ is the number of times that word w precedes v in our corpus. $n(w)$ is the number of times we see word w.
- Say $C : \mathcal{V} \rightarrow \{1, 2, \ldots k\}$ is a *partition* of the vocabulary into k classes
- The model:

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

  (note: $C(w_0)$ is a special start state)
- More conveniently:

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

Measuring the Quality of C

- How do we measure the quality of a partition C?
  (Taken from Percy Liang, MENG thesis, MIT, 2005):

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

The Final Equation

- Define

$$P(c,c') = \frac{n(c,c')}{n} \quad P(w) = \frac{n(w)}{n} \quad P(c) = \frac{n(c)}{n}$$

- Then (again from Percy Liang, 2005):

$$\begin{aligned} \text{Quality}(C) &= \sum_{c,c'} P(c,c') \log \frac{P(c,c')}{P(c)P(c')} + \sum_{w} P(w) \log P(w) \\ &= I(C) - H \end{aligned}$$

The first term $I(C)$ is the mutual information between adjacent clusters and the second term H is the entropy of the word distribution. Note that the quality of C can be computed as a sum of mutual information weights between clusters minus the constant H, which does not depend on C. This decomposition allows us to make optimizations.

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 two 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 $i = (m+1) \ldots |\mathcal{V}|$
  - 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: $O(|\mathcal{V}|m^2 + n)$ where n is corpus length

Miller et al, NAACL 2004

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

Miller et al, NAACL 2004

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 2004

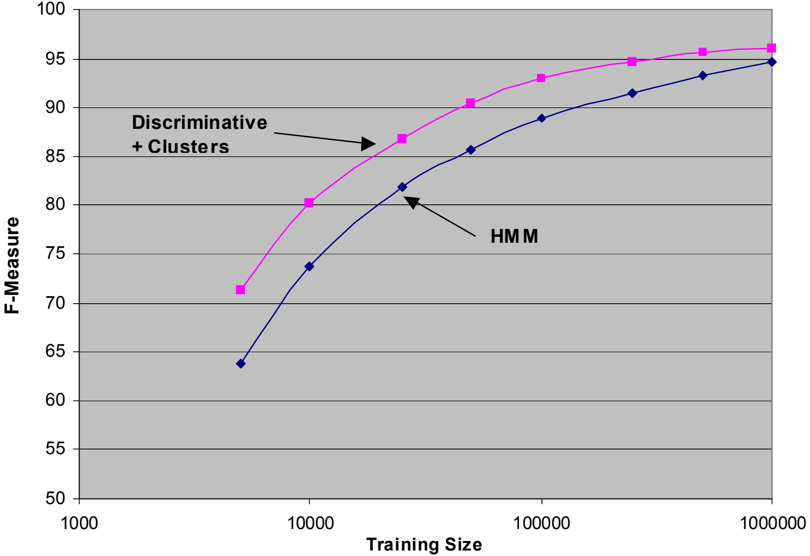
To achieve reasonable performance, the HMM-based technology we presented required roughly 150,000 words of annotated examples, and over a million words to achieve peak accuracy. Given a typical annotation rate of 5,000 words per hour, we estimated that setting up a name finder for a new problem would 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.

Miller et al, NAACL 2004

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

Table 2: Feature Set

Miller et al, NAACL 2004



Miller et al, NAACL 2004