Improving Data Driven Part-of-Speech Tagging by Morphologic Knowledge Induction

Uwe D. Reichel
Department of Phonetics and Speech Communication
University of Munich
reichelu@phonetik.uni-muenchen.de
Abstract

- **Generalization of a Markov model part-of-speech (POS) tagger:** replacing the $P(w|t)$ emission probabilities of word $w$ given tag $t$ by a linear interpolation of tag emission probabilities given a list of representations of $w$

- **Word Representation:** string suffix of word cut off at a local maximum of backward successor variety

- **What for?** retrieval of linguistically meaningful string suffixes, that may relate to certain POS labels, without the need of linguistic knowledge (language independence, addressing out of vocabulary (OOV) problem)

- **Results:** Basic Markov model POS taggers are significantly outperformed.
Basic Form of a Markov POS Tagger (Jelinek, 1985)

- Estimate for most probable tag sequence \( \hat{T} \) given word sequence \( W \)

\[
\hat{T} = \arg \max_T \left[ P(T|W) \right] 
\]

\[
= \arg \max_T \left[ P(T)P(W|T) \right] \quad \text{(Bayes, } P(W) \text{ constant)}
\]

- Simplifying Assumptions
  - Probability of word \( w_i \) depends only on its tag \( t_i \)
  - Probability of tag \( t_i \) depends only on a limited tag history \( t\text{-history}_i \)

\[
\hat{T} = \arg \max_{t_1...t_n} \left[ \prod_{i=1}^{n} P(t_i|t\text{-history}_i)P(w_i|t_i) \right]
\]

- Retrieval of \( \hat{T} \) using the Viterbi algorithm
Generalisations of the Basic Model

- by linear interpolation
- replacing $P(t_i|\text{t-history}_i)$ by $\sum_j u_j P(t_i|\text{t-history}_{ij})$
- replacing $P(w_i|t_i)$ by $\frac{P(w_i)}{P(t_i)} \sum_k v_k P(t_i|\text{w-representation}_{ik})$
  (reapplication of Bayes formula)

$$\hat{T} = \arg\max_{t_1...t_n} \left[ \prod_{i=1}^{n} \frac{1}{P(t_i)} \sum_j u_j P(t_i|\text{t-history}_{ij}) \sum_k v_k P(t_i|\text{w-representation}_{ik}) \right]$$

(4)

- calculation of interpolation weights $u_j$ and $v_k$ via the EM algorithm
Word Representation: String Suffixes

- **Motivation**
  - In many languages (e.g. German, English) POS information is stored in suffix morphemes, inflectional endings, back parts of compounds (Gelegenheit/NN, Umgehungsstraße/NN, partly/ADV)
  
  - The usage of these entities next to the whole word reduces the OOV problem (see also Suendermann and Ney, 2003)

- **Desideratum:** Retrieve linguistically meaningful string suffixes without prior linguistic knowledge → language independence
Word Representation: Retrieval I

- suffixes are determined by Weighted Backward Successor Variety (SV)

- SV of a string: number of different characters that follow it in given lexicon

- Backward SV: SV’s are calculated from reversed strings in order to separate linguistically meaningful suffixes

- Weighting: SV’s are weighted w.r.t. mean SV at the corresponding string position to eliminate positional effects

- lexicon of reversed words represented in form of a trie (see next sheet)

- SV at given state: number of transitions to other states

- Usage: treat SV peaks as morpheme boundaries (cf. Peak and Plateau algorithm (Nascimento and da Cunha, 1998))
Word Representation: Retrieval II

- Lexicon Trie (reversely) storing the entries *Einigung*, *Kreuzigung* and *Eignung*

- The SV maxima at nodes 3 and 5 correspond to the boundaries of the morphemes *ung* and *ig* respectively
Training and Application

**Training**
- build lexicon trie for training material
- get word representations: word form and two string suffixes derived by SV (e.g. kommenden → ['kommenden', 'en', 'enden'])
- calculate for all word representations $w_{rep}$, tags $t$ and tag histories $t_{hist}$: $P(t|w_{rep})$ and $P(t|t_{hist})$ as well as the interpolation weights for equation 4

**Application**
- transform each word into list of representations (as in training)
- get most probable tag sequence via equation 4 applying Viterbi
Data and Results

• **Data:** 382402 tokens tagged by the IMS Tree Tagger (Schmidt, 1995) and partially hand corrected; 85 % used for training, 15 % for testing (OOV: 38.12 % of types, 11.52 % of tokens)

• **Classes:** 54 different POS tags (Tree Tagger inventory)

<table>
<thead>
<tr>
<th></th>
<th>accuracy</th>
<th>$\kappa$</th>
<th>relative entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline Taggers:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unigram</td>
<td>89.61 %</td>
<td>0.89</td>
<td>1.33</td>
</tr>
<tr>
<td>lin. interpolated Trigram</td>
<td>93.22 %</td>
<td>0.93</td>
<td>0.61</td>
</tr>
<tr>
<td><strong>New Tagger:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trigram, word repr.</td>
<td>95.36 %</td>
<td>0.95</td>
<td>0.45</td>
</tr>
</tbody>
</table>

• This study’s tagger significantly outperforms the baseline taggers (two tailed McNemar test, $p = 0.001$)

• erroneous data probably affects accuracy (e.g. finite vs. infinite verbs)

