Advanced NLP

Lecture 4 - Morphology

Morphological Segmentation

• Basic Task: segment an utterance into a sequence of morphemes (the smallest meaningful linguistic units)
  – Example: unresolved $\rightarrow$ un-resolv-ed

• Extensions:
  – Identify role of each morpheme (stem vs. affix)
  – Identify canonical form of the morpheme (e.g., the root of “unresolved” is “resolve”, the root of “took” is “take”)


Related Problem: Word Segmentation

• Task: divide text into a sequence of words
  “Word is a string of contiguous alphanumeric characters with space on either side; may include hyphens and apostrophes but no other punctuation marks” (Kucera and Francis)

• The problem is relative easy for English
  ‘`Wash. vs wash’”
  ‘`won’t”, “John’s”
  ‘`pro-Arab”, ”the idea of a child-as-required-yuppie-possession ...”

• Hard for other languages (Chinese, Arabic, ...)
  – Words are not separated by white spaces

Morphological Segmentation: Cross-Lingual Perspective

• The distinction between the notion of word and morpheme is vague across languages
  – In English, “in” is a word while it is a prefix in Hebrew
  – In English, passive is realized using an auxiliary (“have”), while it is part of the stem in Hebrew

• Languages vary greatly in how morphemes are combined to produce words
Morphological Structures

Two classes of morphemes:
• **Stems** -- the main morpheme of the word that carries its semantic meaning
• **Affixes** -- an auxiliary morpheme that carries additional semantic and grammatical functions
  – Prefix: precedes the stem (English: “unresolved”)
  – Suffix: follows the stem (English: “unresolved”)
  – Infix: inside the stem (Tagalog: “humingi”)
  – Circumfix: combines prefix and suffix (German: “gesagt”)

Morphological Compounding

• Inflectional: grammatical transformations within the same grammatical category
  Example: computer + s = computers
• Derivational: production of words in a different class
  Example: computer + ation = computerization
• Compounding: combination of multiple word stems together
  Example: dog + house = doghouse
• Cliticization: combination of a stem with clitic
  Example: I + ‘ve = I’ve
Human Morphological Processing

How human store morphological variants?
- Full words are stored as units
- Stem/affixes stored separately

Experimental Methods:
- Reading Time: measure reading time for each word
  - Findings: reading time depends on the size of morphological family
- Priming: measure change in recognition time when morphologically related words are repeated
  - Findings: regularly inflected forms are not distinct in the lexicon from their stems
- Analysis of Speech Errors: analyze speech errors (slips of tongue)
  - Findings: inflectional and derivational suffixes appear separately from their stems
How Children Learn Morphology?

Saffran, Newport & Aslin (1996):
• Children estimate the probability of each syllable in the language conditioned on its predecessor
• Children segment utterances at low points of transitional probability

Computational Approaches to Morphological Segmentation

Harris (1954): the successor of letters within words will tend to be more constrained than the successors of letters at the ends of words

Example: compare possible fillings for the two strings “dog ?” vs “zeb?”

Idea: 1. compute “suprisingness” of each letter
2. place boundaries at local maxima of these values
Learning of Word Segmentation: Non-probabilistic Approach


- Identifies word boundaries in Japanese
- Doesn’t assume the presence of lexicon (aka knowledge-lean)
- Uses simple N-gram statistics to place boundaries
  - Optimization criteria inspired by Harris
- Outperforms lexicon and grammar-based morphological analyzers

Word Segmentation

Key idea: for each candidate boundary, compare the frequency of the n-grams adjacent to the proposed boundary with the frequency of the n-grams that straddle it.

For $N = 4$, consider the 6 questions of the form: ”Is $\#(s_i) \geq \#(t_j)$?”, where $\#(x)$ is the number of occurrences of $x$

Example: Is “TING” more frequent in the corpus than ”INGE”? 
Example

\[
\begin{aligned}
\text{A} & \quad \text{B} & \quad \text{C} & \quad \text{D} & \quad \text{W} & \quad \text{X} & \quad \text{Y} & \quad \text{Z} \\
\underline{s_1} & \quad ? & \quad \underline{s_2} & \quad t_1 & \quad t_2 & \quad t_3
\end{aligned}
\]

Figure 2: Collecting evidence for a word boundary
– are the non-straddling \( n \)-grams \( s_1 \) and \( s_2 \) more frequent than the straddling \( n \)-grams \( t_1, t_2, \) and \( t_3 \)?

Algorithm

\( s_1^n \) non-straddling \( n \)-grams to the left of location \( k \)
\( s_2^n \) non-straddling \( n \)-grams to the right of location \( k \)
\( t_j^n \) straddling \( n \)-gram with \( j \) characters to the right of location \( k \)
\( I_{\geq}(y, z) \) indicator function that is 1 when \( y \geq z \), and 0 otherwise.

1. Calculate the fraction of affirmative answers for each \( n \) in \( N \):

\[
v_n(k) = \frac{1}{2 \times (n - 1)} \sum_{i=1}^{2} \sum_{j=1}^{n-1} I_{\geq}((s_1^n), (t_j^n))
\]

2. Average the contributions of each \( n \)-gram order

\[
v_N(k) = \frac{1}{N} \sum_{n \in N} v_n(k)
\]
Algorithm (Cont.)

Place boundary at all locations $l$ such that either:

- $l$ is a local maximum: $v_N(l) > v_N(l - 1)$ and $v_N(l) > v_N(l + 1)$
- $v_N(l) \geq t$, a threshold parameter

---

Experimental Set-Up

- Corpus: 150 megabytes of 1993 Nikkei newswire
- Manual annotations: 50 sequences for development (parameter tuning) and 50 sequences for test data
- Compare against two manually crafted word segmentors (Chasen and Juman)
Evaluation Measures

- **Precision (P):** Percentage of system identified words that are correct.
- **Recall (R):** Percentage of words actually present in the input that were correctly identified by the system.
- **F-Measure (F):** \[ F = \frac{2PR}{P + R} \]

Results

![Graph showing word accuracy](image)

**Figure 4:** Word accuracy. The three rightmost groups represent our algorithm with parameters tuned for different optimization criteria.
Learning of Morphology: Probabilistic Approach


• Identifies morphemic boundaries in Finnish. Successfully applied for many other languages
• Doesn’t assume the repository of morphemes is known apriori
• Objective: find a concise morpheme repository that yields concise representation of data
  – Formulated in Bayesian Framework
• Delivers state-of-the-art performance for several languages

Model Structure

Notations:
• D – a corpus of words $w_1...w_n$ (morphologically unsegmented)
• S – segmentation over D
• Lex – a lexicon which lists a set of allowed morphemes $m$ along with their probabilities $\theta(m)$

Goal: Find lexicon and segmentation

$\text{Lex}^*, S^* = \text{argmax}_{\text{Lex}, S} \ P(\text{Lex}, S | D)$
(Note this is a MAP estimate)

$\text{argmax}_{\text{Lex}, S} \ P(\text{Lex}, S | D) = \text{argmax}_{\text{Lex}, S} \ P(D | \text{Lex}, S) \times P(\text{Lex}, S)$

$= \text{argmax}_{\text{Lex}, S} \ P(\text{Lex}, S)$

$= \text{argmax}_{\text{Lex}, S} \ P(\text{Lex}) \times P(S | \text{lex})$

We assume that $P(D | \text{Lex}, S) = 1$ if segmentation $S$ is consistent with corpus $D$
The model: Estimating $P(S|\text{Lex})$

$D = w_1...w_n$, where $w_i = m_{i1}...m_{il_i}$

$\theta(m)$ - probability of morpheme $m$ specified by Lex

The likelihood of corpus $D$ with segmentation $S$ given Lex:

$$P(S|\text{Lex}) = \prod_{j=1}^{l} \prod_{i=1}^{n} \Theta(m_{ij})$$

The Model: Estimating $P(\text{Lex})$

Prior $P(\text{Lex})$ incorporates our belief about the form of the lexicon (its size, the length and letter composition of a morpheme, the frequency distribution of morphemes in text)

- The prior of our model encodes:
  - lexicon size is distributed uniformly
  - letters in morphemes are selected based on their frequency in text
  - morpheme length follows Gamma distribution
  - morpheme frequency follows Zipfian distribution
The Model: Estimating P(Lex)

Assuming lexicon of length M:

\[ P(\text{Lex}) = M! \cdot P(M, N) \cdot \prod_{i=1}^{M} P(l_i) \cdot \prod_{j=1}^{l_j} P(c_{ij}) \cdot P(\Theta_i | N) \]

- \( M! \) – accounts for different orders in which morphemes in the lexicon could be generated
- \( P(M, N) \) – probability that the number of morpheme types in Lex is M and the number of morpheme tokens is N
  - Assume that \( P(M, N) \) is constant for all reasonable M and N

P(l) – probability that morpheme \( m \) has length \( l \)

- Modeled using Gamma distribution with \( \alpha \) and \( \beta \) as hyperparameters

\[ P(l) = \frac{l^{\alpha-1}e^{-\frac{l}{\beta}}}{\Gamma(\alpha)\beta^\alpha} \]

- The Gamma distribution peaks at \((\alpha - 1)\), and controls skewness of the distribution.
  - If the most frequent morpheme length is 4, then we set \( \alpha = 5 \)
  - We set \( \beta = 1 \)
The Model: Estimating $P(\text{Lex})$

Assuming lexicon of length $M$:

$$P(\text{Lex}) = M! \cdot P(M, N) \cdot \prod_{i=1}^{M} P(l_i) \cdot \prod_{j=1}^{l_i} P(c_j) \cdot P(\Theta_i \mid N)$$

- Probability of a character $c$ in a morpheme
  $$p(c) = \frac{\text{count } c}{\text{count of all characters}}$$

- Morpheme probability is computed using unigram LM
The Model: Estimating $P(\text{Lex})$

Assuming lexicon of length $M$:

$$P(\text{Lex}) = M! \cdot P(M, N) \cdot \prod_{i=1}^{M} P(l_i) \cdot \prod_{j=1}^{b_j} P(c_{ij}) \cdot P(\Theta_j | N)$$

- Prior on the probability of morpheme occurrence (this distribution ensures Zipfian behaviour)

$$P(\Theta | N) = (\Theta \cdot N)^{\log_2(1-h)} - (\Theta \cdot N + 1)^{\log_2(1-h)}$$

$h$ is a probability that a morph type will be expected to occur only once in the corpus.

Search

- Start with a segmentation where each word corresponds to a single morpheme
- Consider all possible splits for the $i$-th word in the corpus:
  - Select the split with the highest probability $P(\text{Lex}, S | D)$ across all possible splits or no split
  - In the case of split, continue recursively to process the two fragments
  - Compute MLE lexicon for given segmentation
- Repeat the previous step until convergence

This is a greedy search with no theoretical guarantees
In few lectures, we will study more effective search strategies
Results: Finnish

![Graph comparing Finnish results]

Figure 1: Expectation of the percentage of recognized morphemes for Finnish data.

Results: English

![Graph comparing English results]

Figure 2: Expectation of the percentage of recognized morphemes for English data.
Projection: Stem Prediction

David Yarowsky, Grace Ngai, Richard Wicentowski

“Inducing Multilingual Text Analysis Tools via Robust Projection across Aligned Corpora”, 2001

- **Task:** find a root of the word given its inflected form
  - defies -> defy
  - skipped -> skip
  - took -> take
- **Input:** parallel text in two languages annotated with part-of-speech tags
  - Tags discriminate between roots and inflections
  - Lemmatizer that connects roots and inflections for one language

Direct-bridge French inflection/root alignment

Inflection “croyant” and root “croire” are connected via believing (their English translation)
(this approach is limited since typically translation preserves tenses)
Multi-bridge French inflection-root alignment

- Use English lemmatizer to compute a multi-step transitive association:
  croyaient → believed → believe → croire
- We can build similar chains for other translations of the word of interest
  croyaient → thought → think → croire

Notations:
- \( E_{\text{lem}} \) -- all English lemmas (belived, belive, believing)
- \( F_{\text{inf}} \) -- foreign inflection (croyaient)
- \( F_{\text{root}} \) -- foreign root (croire)

\[
P_{mp}(F_{\text{root}}|F_{\text{inf}}) = \sum_i P_a(F_{\text{root}}|E_{\text{lem},i}) P_a(E_{\text{lem},i}|F_{\text{inf}})
\]

Example:

\[
P_{mp}(\text{croire}|\text{croyaient}) =
\]
\[
P_a(\text{croire}|\text{BELIEVE}) P_a(\text{BELIEVE}|\text{croyaient}) +
P_a(\text{croire}|\text{THINK}) P_a(\text{THINK}|\text{croyaient}) + ...
\]
Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Precisions</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>typ</td>
<td>tok</td>
</tr>
<tr>
<td><strong>FRENCH Verbal Morphology Induction</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French Hansard (12M words):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MProj only</td>
<td>.952</td>
<td>.999</td>
</tr>
<tr>
<td>MProj+MTrig</td>
<td>.998</td>
<td>.999</td>
</tr>
<tr>
<td>MProj+MTrig+BM</td>
<td>.994</td>
<td>.999</td>
</tr>
<tr>
<td>French Hansard (12M words):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MProj only</td>
<td>.985</td>
<td>.988</td>
</tr>
<tr>
<td>MProj+MTrig</td>
<td>.995</td>
<td>.999</td>
</tr>
<tr>
<td>MProj+MTrig+BM</td>
<td>.979</td>
<td>.998</td>
</tr>
<tr>
<td>French Bible (100K words):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MProj only</td>
<td>.962</td>
<td>.951</td>
</tr>
<tr>
<td>MProj+MTrig</td>
<td>.984</td>
<td>.991</td>
</tr>
<tr>
<td>MProj+MTrig+BM</td>
<td>.932</td>
<td>.989</td>
</tr>
<tr>
<td>French Bible (100K words):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MProj only</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>MProj+MTrig</td>
<td>.991</td>
<td>.998</td>
</tr>
<tr>
<td>MProj+MTrig+BM</td>
<td>.984</td>
<td>.998</td>
</tr>
<tr>
<td>Czech Reader's Digest (500K words):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MProj only</td>
<td>.928</td>
<td>.975</td>
</tr>
<tr>
<td>MProj+MTrig</td>
<td>.981</td>
<td>.991</td>
</tr>
<tr>
<td>MProj+MTrig+BM</td>
<td>.964</td>
<td>.991</td>
</tr>
</tbody>
</table>

Adding More Monolingual Parallel Data
Supervised: Stem Prediction

- Assume manually annotated data for stem prediction (e.g., 250 verbs and their inflections)
- We predict stems by considering probabilities of different transformations:
  \[ P(\text{root}|\text{inflection}) = P(\delta|\delta \alpha) = P(\alpha \rightarrow \beta|\delta \alpha) = \sum \lambda_i P(\alpha \rightarrow \beta|h_i) \quad \text{for } h_i = \text{suffix}(i, \delta \alpha) \]

  Example:
  \[ P(\text{commencer}|\text{commença}) = \lambda_0 P(\text{ça} \rightarrow \text{cer}) + \lambda_1 P(\text{ça} \rightarrow \text{cer}(a)) + \lambda_2 P(\text{ça} \rightarrow \text{cer}(ça)) + \lambda_3 P(\text{ça} \rightarrow \text{cer}(ança)) + \lambda_4 P(\text{ça} \rightarrow \text{cer}(ença)) + \ldots \]
  \[ P(\text{ployer}|\text{plœie}) = P(\text{ie} \rightarrow \text{yer}|\text{plœie}) = \lambda_0 P(\text{ie} \rightarrow \text{yer}) + \lambda_1 P(\text{ie} \rightarrow \text{yer}(e)) + \lambda_2 P(\text{ie} \rightarrow \text{yer}|\text{ie}) + \lambda_3 P(\text{ie} \rightarrow \text{yer}|\text{œie}) + \alpha_4 P(\text{ie} \rightarrow \text{yer}|\text{loie}) + \ldots \]

Summary

- Unsupervised algorithms for morphological analysis capitalize on the difference in recurrence patterns within and across morphemes.
- Probabilistic methods provide effective means for incorporating our prior beliefs about the structure of morphological dictionary.
- The performance of unsupervised methods varies greatly across languages.