# Korean morphology

Seong-Hwan Jun

# Morphology

- Morpheme: smallest grammatical unit
- Word is composed of one or more morphemes
- Example: *Unbreakable* is made up of
    1. Un-: **bound** morpheme, cannot stand on its own
    2. break: **free** morpheme (**lexeme**)
    3. -able: free morpheme

- Derivational morpheme: changes the part-of-speech as well as semantic meaning:

  1. un-: changes the meaning

  2. -able: changes the part-of-speech

- Inflectional morpheme: does **not** change the part-of-speech nor semantic meaning:

  1. -s: pluralization

  2. -ed: past participle

# Computational morphology

- Field of morphology: studies everything about morphemes
- Computational morphology is focused on two tasks:
  1. **Morphological analysis**
  2. **Morphological disambiguation**

- Morphological analyzer: produce all possible analysis of a word in terms of part-of-speech and inflections
- Morphological disambiguation: choose the most plausible analysis
- Example: breaks
  1. V+3SG
  2. N+PL
- He took too many **breaks** during work hours!

  * N+PL

# Computational morphology: Korean

- Morphemes add on to the main lexeme (agglutination)
- Example: 강가에서 (from riverbank)
    1. lexeme: 강가 (riverbank)
    2. bound morpheme: 에서 (...from)
- Previous approaches: dictionary-based, rule-based (extracted from corpus-based)

# Problems

- Unknown words due to finite size of dictionary and corpus
- Unknown words are tagged as common noun by default
- Rule-based approach...

- Suppose you observe word, *kicked* (assume that you have never seen the word *kick* before)
- What is your guess at the part-of-speech of this word by observing *-ed*?
- Rule-based approach is only a heuristic and the accuracy depends on the size of the corpus from which the rule was extracted from
- **Main idea: learn the rules**

# Clustering I

- To learn the rules, more data, the better
- Cannot possibly expect to annotate/label
- Idea: group the words that are "similar"
- Words belonging to similar groups can be used for learning rules that are frequently occurring for that group

# String alignment

- Numerous ways to measure similarities between two strings $w_i$ and $w_j$
  1. Levenshtein distance
  2. Probabilistic model over strings
- Probabilistic model, sums over all possible alignments of $w_i$ and $w_j$:

$$
\begin{aligned}
p(w_i, w_j) &= \sum_{A \in \mathbf{A}_{w_i, w_j}} p(w_i, w_j, A) \\
&\propto \sum_{A \in \mathbf{A}_{w_i, w_j}} \exp\{\theta' f(A)\}
\end{aligned}
$$

- Log-linear model: features are defined on the alignments.
- An example of a feature is how many times a character is aligned with another character.
- Example: raining and rainier can be aligned as,

  raining

  raini--er

  f=(0, ..., 0, 1, 1, 2, 1, 0, ..., 0) because *r* is aligned with *r* once, *a* is aligned with *a* once, *i* aligned with *i* twice and so on

- If $p(w_i, w_j) > p(w_i, w_k)$, then we can conclude that $w_i$ and $w_j$ fit better together.

# Clustering II: DPMM

- Analogy: Chinese Restaurant Process
- Customer *i* (word) enters the restaurant, chooses to seat at a table *l* with probability proportional to the number of customers (words) already seated at the table
- Alternatively, customer may choose to seat at a new table with probability proportional to $\alpha_0$ (a parameter to be trained)

# Inference

- Once the table is chosen, we can assess the similarity of the customer *i* (word) with the other customers (words) already seated at the table using the probabilistic model over strings
- Inference method: Gibbs sampling method, which iteratively re-assess...
  - the cluster of the words (CRP)
  - the part-of-speech tag (tri-gram model)
  - the inflection tag (log-linear model)

# Training

- Once the grouping of the words become stable, we train the parameters based on the groups by grabbing features from the words
- Parameters:

1. $\theta$: probabilistic model over the strings
2. $\tau$: trigram part-of-speech tagger
3. $\phi$: inflection tagging model

# POS tagging

Trigram model

$$
\begin{aligned}
t_i &= \text{argmax}_t p(t_i|\mathbf{l}, \mathbf{s}, \mathbf{w}, \mathbf{t}_{-i}) \\
&= \text{argmax}_t \frac{p(l_i, s_i|t_i, \mathbf{t}_{-i}, \mathbf{l}_{-i}, \mathbf{s}_{-i}, \mathbf{w}) p(t_i|\mathbf{t}_{-i})}{p(l_i, s_i|\mathbf{t}_{-i}, \mathbf{l}_{-i}, \mathbf{s}_{-i}, \mathbf{w})} \\
&= \text{argmax}_t p(l_i, s_i|t_i, \mathbf{t}_{-i}, \mathbf{l}_{-i}, \mathbf{s}_{-i}, \mathbf{w}) p(t_i|\mathbf{t}_{-i})
\end{aligned}
$$

$$p(l_i|t_i, \mathbf{t}_{-i}, \mathbf{l}_{-i}, \mathbf{w}) \qquad p(s_i|t_i, \mathbf{t}_{-i}, l_i, \mathbf{l}_{-i}, \mathbf{s}_{-i}, \mathbf{w})$$

DPMM

Inflection model: log-linear

Note: Does not depend on word counts -- solves the unknown words problem

# Reflection

1. Parts of the code are implemented, not able to put everything together
2. Hence, no experiments and not able to fully explore the models (no model tweaking)
3. Research-based project, too much time spent on learning... in order to put together a paper
4. Learned many new methods, re-learned already known methods really well
5. Getting Korean font installed for MikTex distribution of LaTeX is hard.