Computational Linguistics Seminar
LING-696G

Week 9
IBM Models 4 and 5

• Model 3:
  1. translation of words (T-table), t(e | f)
  2. reordering (distortion),
  3. insertion of words (NULL insertion),
  4. dropping of words (words with fertility 0), and one-to-many translation (fertility).

– alignment (distortion) search space too big
  • uses sampling with hill-climbing
IBM Models 4 and 5

- Uses POS tagging (word classes):
  - need a tagger for foreign words (or some clustering method)
- Uses a relative distortion model:
  - **cept**:
    - foreign word mapping to at least one English word
  - center of a cept:
    - ceiling(average output position)
  - offset from center of previous cept

![Alignment Diagram]

![Foreign words and cepts table]
IBM Models 4 and 5

- Relative distortion:
  - first word of a cept $i$
    - $d_1(j - \pi i - 1)$
    - Example: relative distortion for 1st word of $\pi_3 = -1$ since $\pi 2 = 4$
  - next words of a cept $i$
    - $\pi_{i,k} = \text{position of the } k\text{th word in the } i\text{th cept}$
    - $d_{\geq 1}(j - \pi_{i,k-1})$
    - Example: position of to is $\pi_{4,0} = 5$, and the position of the is $j = 6$

<table>
<thead>
<tr>
<th>Foreign words and cepts</th>
<th>English words $e_j$ and distortion</th>
</tr>
</thead>
<tbody>
<tr>
<td>cept $\pi_i$</td>
<td>$\pi_1$  $\pi_2$  $\pi_3$  $\pi_4$  $\pi_5$</td>
</tr>
<tr>
<td>foreign position $[i]$</td>
<td>1        2        4        5        6</td>
</tr>
<tr>
<td>foreign word $f_i$</td>
<td>ich      gehe     nicht    zum      haus</td>
</tr>
<tr>
<td>English words $e_j$</td>
<td>I        go       not      to, the   house</td>
</tr>
<tr>
<td>English positions $j$</td>
<td>1        4        3        5, 6      7</td>
</tr>
<tr>
<td>center of cept $\ominus_j$</td>
<td>1        4        3        6        7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$e_j$</th>
<th>1</th>
<th>do</th>
<th>not</th>
<th>go</th>
<th>to</th>
<th>the</th>
<th>house</th>
</tr>
</thead>
<tbody>
<tr>
<td>in cept $\pi_{i,k}$</td>
<td>$\pi_{1,0}$</td>
<td>$\pi_{0,0}$</td>
<td>$\pi_{3,0}$</td>
<td>$\pi_{2,0}$</td>
<td>$\pi_{4,0}$</td>
<td>$\pi_{4,1}$</td>
<td>$\pi_{5,0}$</td>
</tr>
<tr>
<td>$\ominus i = 1$</td>
<td>0</td>
<td>-</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>$j - \ominus i = 1$</td>
<td>+1</td>
<td>-</td>
<td>-1</td>
<td>+3</td>
<td>+2</td>
<td>-</td>
<td>+1</td>
</tr>
<tr>
<td>distortion</td>
<td>$d_1(+1)$</td>
<td>1</td>
<td>$d_1(-1)$</td>
<td>$d_1(+3)$</td>
<td>$d_1(+2)$</td>
<td>$d_{\geq 1}(+1)$</td>
<td>$d_1(+1)$</td>
</tr>
</tbody>
</table>
IBM Models 4 and 5

• Relative distortion:
  – conditioned on POS tags
  – for initial word in cept:
    • \( d_1(j − \circ_{i−1} | A(f_{i−1}), B(e_j)) \)
  – for additional words:
    • \( d_{>1}(j − \pi_{i,k−1} | B(e_j)) \)

– Two functions \( A(f) \) and \( B(e) \) that map words to their word classes
– the operator \([i]\) maps cept \( i \) back to its corresponding foreign input word position.
NLTK IBM Model 4

- class nltk.translate.ibm4.IBMModel4(sentence_aligned_corpus, iterations, source_word_classes, target_word_classes, probability_tables=None)

```python
text = []
text.append(AlignedSent(['klein', 'ist', 'das', 'haus'], ['the', 'house', 'is', 'small']))
text.append(AlignedSent(['das', 'haus', 'war', 'ja', 'groß'], ['the', 'house', 'was', 'big']))
text.append(AlignedSent(['das', 'buch', 'ist', 'ja', 'klein'], ['the', 'book', 'is', 'small']))
text.append(AlignedSent(['ein', 'haus', 'ist', 'klein'], ['a', 'house', 'is', 'small']))
text.append(AlignedSent(['das', 'haus'], ['the', 'house']))
text.append(AlignedSent(['das', 'buch'], ['the', 'book']))
text.append(AlignedSent(['ich', 'fasse', 'das', 'buch', 'zusammen'], ['i', 'summarize', 'the', 'Book']))
text.append(AlignedSent(['fasse', 'zusammen'], ['summarize']))
src_classes = {'the': 0, 'a': 0, 'small': 1, 'big': 1, 'house': 2, 'book': 2, 'is': 3, 'was': 3, 'i': 4, 'summarize': 5}
trg_classes = {'das': 0, 'ein': 0, 'haus': 1, 'buch': 1, 'klein': 2, 'groß': 2, 'ist': 3, 'war': 3, 'ja': 4, 'ich': 5, 'fasse': 6, 'zusammen': 6}

ibm4 = IBMModel4(text, 5, src_classes, trg_classes)
```
IBM Model 5

• Newer distortion model to prevent deficiency
  – With Models 3 and 4, nothing prohibits the placement of an output word into a position that has already been filled.

• Let $v_j$ the number of vacancies in the English output interval $[1; j]$
  – $d_1(v_j \mid B(e_j), v_{\circ i-1}, \text{vmax})$    \quad A(f_{i-1}) \text{ gone!}
  
  – $d_{>1}(v_j - v_{\pi i, k-1} \mid B(e_j), \text{vmax}')$
• `class nltk.translate.ibm5.IBMMModel5(sentenceAlignedCorpus, iterations, source_word_classes, target_word_classes, probability_tables=None)`

• `ibm5 = IBMMModel5(bitext, 5, src_classes, trg_classes)`
Alignment Matrix

- English: rows; Foreign language: columns
Alignment Matrix

- English: rows; Foreign language: columns

= "bit into the grass"

phrasal alignment
ibm3a.py

- ibm1 training data:
  1. the a book house
  2. das ein Buch Haus
  3. the house
  4. das Haus
  5. the book
  6. das Buch
  7. a book
  8. ein Buch

- ibm3a training data:
  1. to the house
  2. das ja zum Haus
  3. the house
  4. das ja Haus
  5. 1 0 1
  6. 0 1 1
  7. 1 1 0
  8. 2 0 0
  9. 0 2 0
  10. 0 0 2
  11. to the house
  12. zum Haus
  13. 2 1
  14. 1 2
  15. 3 0
  16. 0 3

fertility counts e.g. line 5
das:1 ja:0 Haus:1
ibm_fertility.py

```python
# command line parameters
# l-f [filename] output_filename
# James O'Donnell, Sandway Fong
import json
import argparse
import itertools
p = argparse.ArgumentParser(description='Machine Translation IBM Model Fertility')
p.add_argument('-f', '--fertility',
                help='maximum fertility 1-4 (default: 2)',
                type=int, nargs='?', default=2, const=1, choices=range(1,5))
p.add_argument('-o', '--output',
                help='output data filename (default: ibm_data2.txt)',
                type=argparse.FileType('r'), default='ibm_data2.txt')
p.add_argument('-t', '--tracing',
                help='print fertility permutations 0-1 (default: 0)',
                type=int, nargs='?', default=0, const=1, choices=range(0,2))

args = p.parse_args()

if args.fertility:
    print('Training data: %s' % format(args.filename.name))
if args.output:
    print('Max fertility output: %s' % format(args.output.name))

for fertility in range(args.fertility):
    print('Max fertility %s' % format(args.fertility))
    for tracing in range(args.tracing):
        max_fertility = args.fertility
        trace = args.tracing
        if tracing:
            e_words = fi.readline()
            fo.write(e_words)
            f_words = fi.readline()
            fo.write(f_words)
        else:
            e_line = fi.readline()
            if not e_line:
                break
            fo.write(e_line)
            f_line = fi.readline()

for i in c:
    if max(j) <= fertility and sum(j) == len(WORDS):
        print('%s' % format(i))
        print('trace: %s' % format(p))
        fertile_set.add(j)
        for k in p:
            if k in fertile_set:
                print('trace: %s' % format(k))
                fertile_set.add(k)
                i+=1

for i in e:
    print('fertility too low for sentence pair')
    break

fo.write(f_lines)
fo.close()
```
ibm3a.py

- **ibm3a training data:**
  1. to the house
  2. das ja zum Haus
  3. the house
  4. das ja Haus
  5. 1 0 1
  6. 0 1 1
  7. 1 1 0
  8. 2 0 0
  9. 0 2 0
  10. 0 0 2
  11. to the house
  12. zum Haus
  13. 2 1
  14. 1 2
  15. 3 0
  16. 0 3

- the house
- das Haus
- the house
- ja Haus
- the house
- das ja
- the house
- das das
- the house
- ja ja
- the house
- Haus Haus
IBM Model 3a

- IBM Model 3
  - Factor in fertility and $t(\text{null} \mid \text{fw}_j)$
  - Problems ...
    - $n(0 \mid \text{fw}_j) = 1$ means $t(\text{null} \mid \text{fw}_j)$
    - How to initialize $t$?
  - Code: given es and fs
    - e.g. *the house* and *das das*
    - $k = t[\text{es}[j]][\text{fs2}[i]] \ast a[l_f][l_e][i][j] \ast n[fs2[i]][f\_lookup[fs2[i]]]/s_{\text{total}}[\text{es}[j]]$
    - where $f\_\text{lookup} = \text{fertility number}$
The Knight's Tale

lines 1-34: About Duke Theseus, lord of Athens, and his achievements

Once on a time, as old stories tell to us,
There was a duke whose name was Theseus:
Of Athens he was lord and governor,
And in his time was such a conqueror
That greater was there not beneath the sun.
Very many rich countries had he won;
What with his wisdom and his chivalry
He gained the realm of Femininity,
That was of old time known as Scythia.
There he married the queen, Hippolyta,
And brought her home with him to his country.
In glory great and with great ceremony,
And, too, her younger sister, Emily.
And thus, in victory and with melody,
Let me this noble duke to Athens ride
With all his armed host marching at his side.

Whilom, as olde stories tellen us,
Ther was a duc that highte Theseus;
Of Athenes he was lord and governour,
And in his tyme swich a conquerour,
That gretter was ther noon under the sonne.
Ful many a riche contree hadde he wolonne,
What with his wyssdom and his chivalrie;
He conqueret al the regne of Femenye,
That whilom was cyled Scythia,

And weddede the queene Ypolita,
And broghte hir hoom with hym in his contree,
With muchel glorie and greet solempnyte,
And eek hir yonge suster Emelye.
And thus with victorie and with melodie

Lete I this noble duc to Atthenes ryde,
And al his hoost, in armes hym bisyde.