CZ4045 Natural Language Processing

Tutorial 2: Word-Level Processing

Question 1

- Consider the following word segmentation algorithm in the lecture notes:

- Given a lexicon of Chinese, and a string
  1. Start a pointer at the beginning of the string
  2. Find the longest word in dictionary that matches the string starting at pointer
  3. Move the pointer over the word in string
  4. Goto2

- Following the algorithm, you perhaps end up with failing to segment a string, if you cannot find a matching.

Question 1 (cont)

- Example
  - String to segment: thetablesdownthere
  - lexicon: the table down there bled own.
- Discuss how to fix the above problem.

Answer Q1

1. Start a pointer at the beginning of the string
2. Find the longest word in dictionary that matches the string starting at pointer
    1. If matched, move the pointer over the word in string
    2. If no word is matched, **skip to the next letter**
3. Goto2

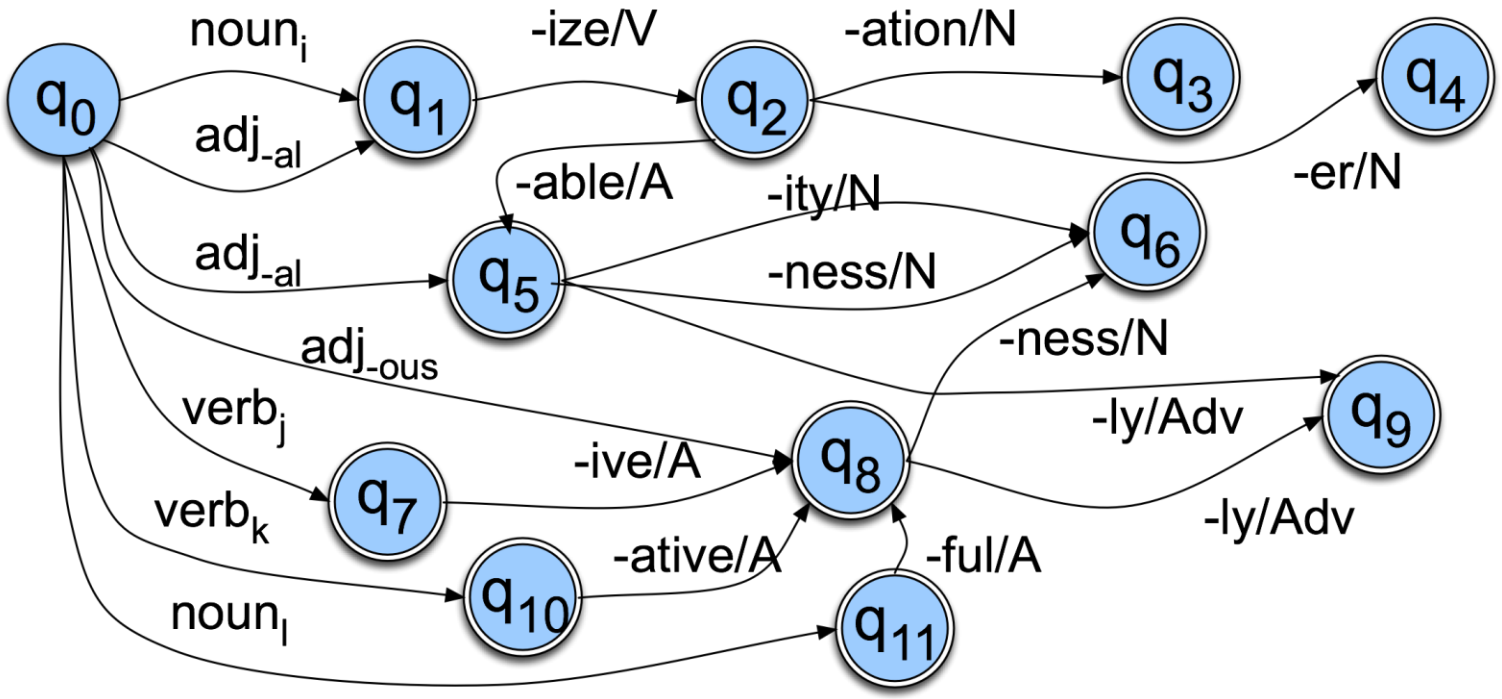
– String to segment: thetablesdownthere
– lexicon: the table down there bled own.

Answer 1 (a bit more processing)

1. Start a pointer at the beginning of the string
2. Find the longest word in dictionary that matches the string starting at pointer
    1. If matched,
        1. Run an FSA for morphological analysis from the beginning of the word
        2. Move the pointer over the FSA-matched part of the string
    2. If no word is matched, skip to the next letter
3. Goto2

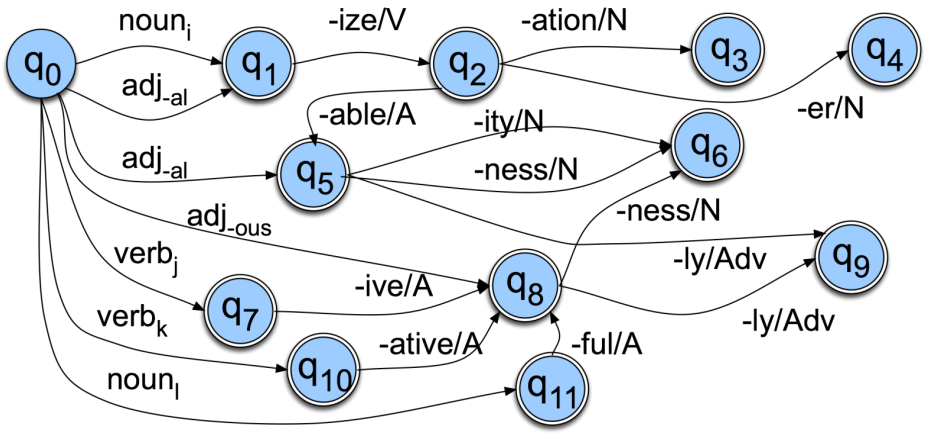
Question 2

- Give examples of each of the noun and verb classes below, and find some **exceptions** to the rules.



Answer 2

- Examples:
  - $noun_i$: fossil
  - $verb_j$: pass
  - $verb_k$: conserve
  - $noun_l$: wonder

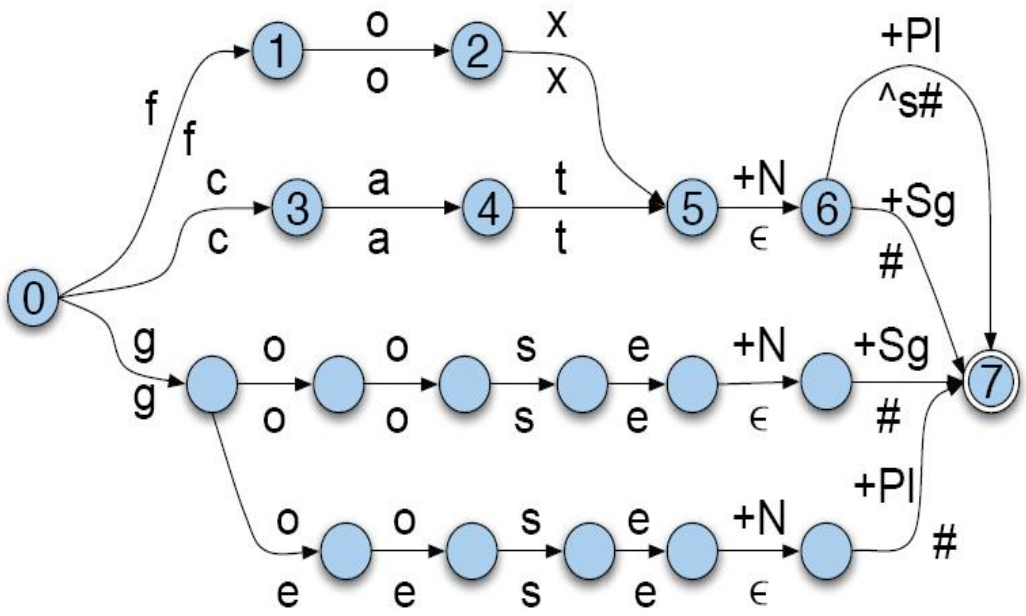


- Exceptions:
  - $noun_i$: **apology** accepts -ize but apologization sounds odd
  - $verb_j$: **detect** accepts -ive but it becomes a noun, not an adjective
  - $verb_k$: **cause** accepts -ative but causitiveness sounds odd
  - $noun_l$: **arm** accepts -ful but it becomes a noun, not an adjective

Question 3: Finite state **transducer** (FST)

- FST is a type of FSA that maps between two sets of symbols.
  - The example figure is an FST that maps between surface level (i.e. actual spelling of words) and lexical level (i.e. concatenation of morphemes making up a word).

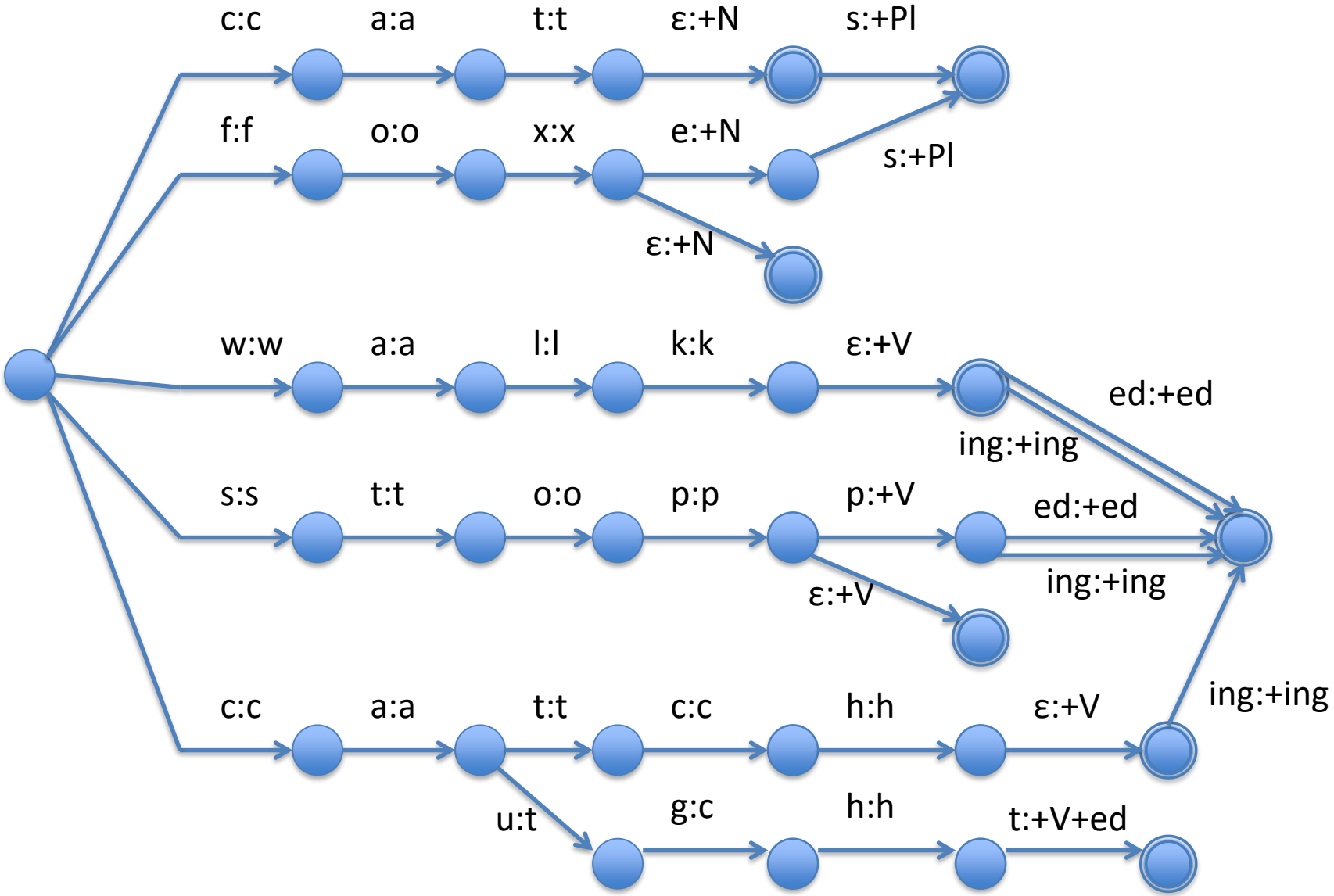
- For example, it can map “cats” and “cat+N+Pl” (Pl: plural).
- Each transition is associated with a pair of two characters
  - Example: e:o, #:+Pl).
- # indicates a word boundary, and ^ a morpheme boundary.



Question 3

- Write a FST for the following mappings:
  - cat – cat+N, cats – cat+N+Pl
  - fox – fox+N, foxes – fox+N+Pl
  - walk – walk+V, walked – walk+V+ed, walking – walk+V+ing
  - stop – stop+V, stopped – stop+V+ed, stopping – stop+V+ing
  - catch – catch+V, caught – catch+V+ed, catching – catch+V+ing

Answer 3


c:c
a:a
t:t
ε:+N
s:+Pl
f:f
o:o
x:x
e:+N
s:+Pl
ε:+N
w:w
a:a
l:l
k:k
ε:+V
ed:+ed
ing:+ing
s:s
t:t
o:o
p:p
p:+V
ed:+ed
ing:+ing
ε:+V
c:c
a:a
t:t
c:c
h:h
ε:+V
ing:+ing
u:t
g:c
h:h
t:+V+ed

Question 4

- Write a program to do the following tasks:
  - **Download** the Web page of a given link and **extract** the text content of the page
  - **Split** the text into sentences and **count** sentences
  - **Split** the text into tokens and **count** token types
  - **Find** lemmas (or stems) of the tokens and **count** lemma types
  - Do **stemming** on the tokens and **count** unique stemmed tokens

Answer 4

- Example URL:
  - https://en.wikipedia.org/wiki/Natural_language_processing
- urllib
  - https://docs.python.org/3/library/urllib.html
- NLTK
  - https://www.nltk.org/
- Beautiful Soup
  - https://www.crummy.com/software/BeautifulSoup/bs4/doc/#porting-code-to-bs4

Alternative libraries:
StanfordNLP
OpenNLP
SpaCy

Sample code

```
import urllib.request
import nltk
from bs4 import BeautifulSoup

with urllib.request.urlopen ('https://en.wikipedia.org/wiki/Natural_language_processing') as response:
    html=response.read()

text = BeautifulSoup(html, "lxml").get_text()
sentences = nltk.tokenize.sent_tokenize(text)
print ('Number of sentences: '+ str(len(sentences)))

tokens= nltk.tokenize.word_tokenize(text)

print ('Number of tokens: '+ str(len(tokens)))

token_types = list(set(tokens))

print ('Number of token types: '+ str(len(token_types)))

wnl=nltk.stem.WordNetLemmatizer()

stemmer = nltk.stem.porter.PorterStemmer()
lemma_types= set()
stemmed_types= set()

for token_type in token_types:
    lemma_types.add(wnl.lemmatize(token_type))
    stemmed_types.add(stemmer.stem(token_type))

print ('Number of lemma types: '+ str(len(lemma_types)))
print ('Number of stemmed types: '+ str(len(stemmed_types)))
```