# Computational Linguistics

## 2. Text Normalization, Finite State Transducers, and Morphological Parsing

**Xiaojing Bai**

**Tsinghua University**

https://bxjthu.github.io/CompLing

# At the end of this session you will

- learn that corpora are important linguistic data for NLP;
- know the basic tasks in text processing;
- understand what are finite state automata and finite state transducers;
- understand how a finite state transducer is used in morphological parsing;
- learn Python functions and the functions in NLKT for text normalization.

Over the past 20 years, computational linguistics has grown into both an exciting area of scientific research and a practical technology that is increasingly being incorporated into consumer products (for example, in applications such as Apple's Siri and Skype Translator). **Four key factors** enabled these developments: (i) a vast increase in computing power, (ii) the availability of very large amounts of **linguistic data**, (iii) the development of highly successful machine learning (ML) methods, and (iv) a much richer understanding of the structure of human language and its deployment in social contexts.

Hirschberg, J., & Manning, C. D. (2015).

**Advances in Natural Language Processing**. *Science*, 349(6245), 261-266.

Linguistic Data Consortium







# LDC Catalog

LDC's Catalog contains hundreds of holdings. Use the buttons below to browse, search, and view catalog entries.

| | |
|---|---|
| by year | corpora sorted by release year |
| top ten corpora | the ten most-distributed LDC corpora |
| projects | LDC corpora attributed to project-based research |
| search | search the Catalog by corpus name, catalog number, language, etc. |
| memberships | current LDC memberships |

# Some notes on *Corpus/Corpora*

A corpus is a **collection** of **machine-readable text or speech** produced in a **natural** communicative setting.

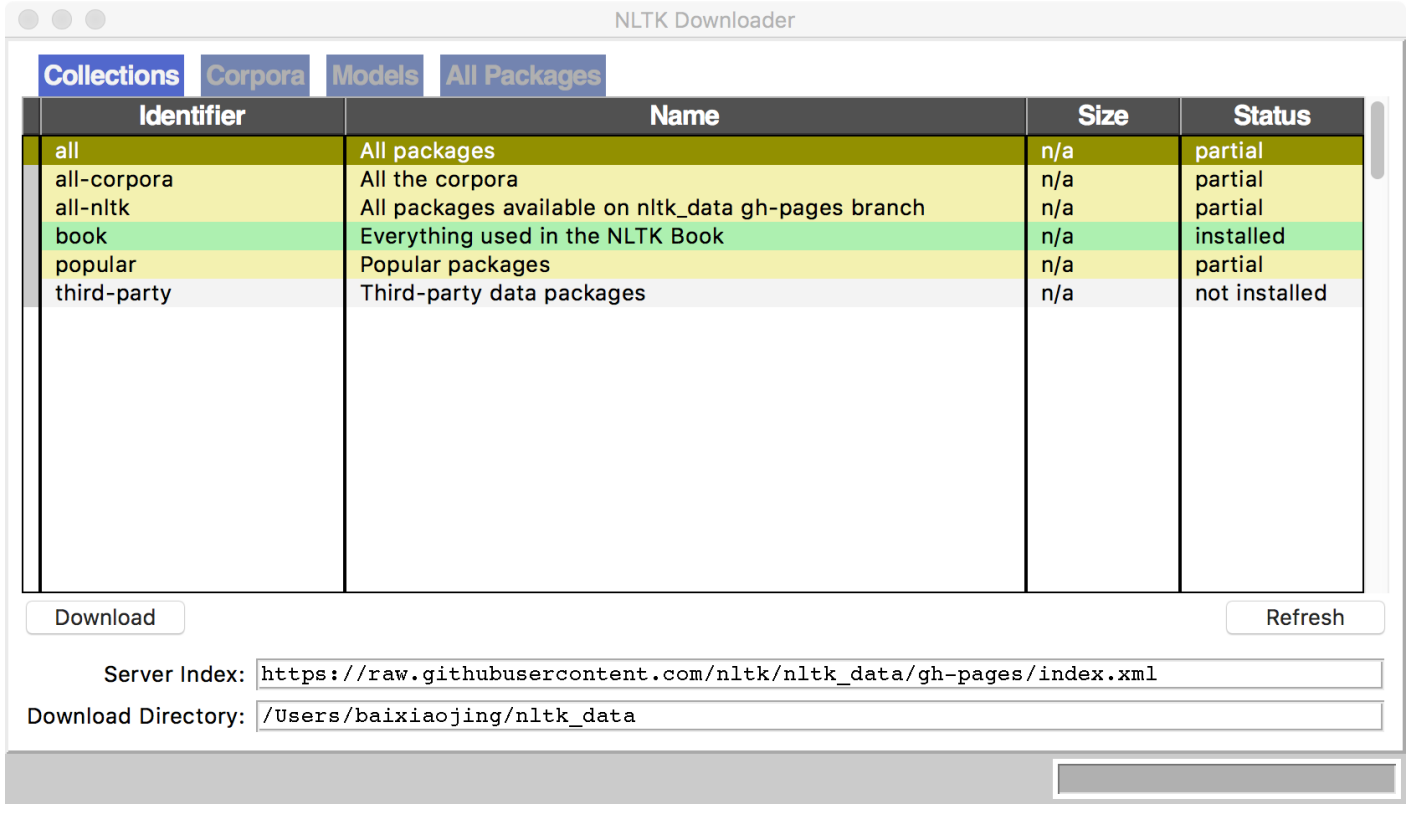
- Representative
- Balanced

Dimensions of variation

- Language
- Genre
- Writer/speaker
- Time

Later in this course (12. Language Resources in NLP):
history of corpus linguistics (ups and downs, people/reasons/events behind),
well-known corpora, use of corpora, etc.

```
>>> import nltk
>>> nltk.download()
```




More at http://www.nltk.org/book/ch01.html and http://www.nltk.org/book/ch02.html

Let's start our study of language from a computational perspective with ...

# What counts as a *word*?

- Text vs. speech
- Punctuations
- Lemma vs. wordform
- Word type vs. word token
- English vs. Chinese

E.g.

a. *He stepped out into the hall, was delighted to encounter a water brother.*

b. *I do uh main- mainly business data processing*

c. *His cat is different from other cats!*

d. *They picnicked by the pool, then lay back on the grass and looked at the stars.*

e. *他特别喜欢北京烤鸭。*

# Text normalization

## What every NLP task needs to do!

1. Tokenize/segment words in running text
2. Lemmatize word forms
3. Segment sentences in running text

# Tokenization: What counts as a *word*? (word boundary)

- Space? Punctuations? e.g. *m.p.h., Ph.D., AT&T, cap'n*
- Clitic contractions *what're, we're*
- Multiword expressions and hyphenated words *San Francisco-based*
- Special characters and numbers *$45.55, 01/02/06*
- URLs and email addresses *https://bxjthu.github.io/CompLing*, *bxj@tsinghua.edu.cn*

# Tokenization: What counts as a *word*? (word boundary)

Penn Treebank tokenization standard

**Input:** "The San Francisco-based restaurant," they said, "doesn't charge $10".

**Output:**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| " | The | San | Francisco-based | restaurant | , | " | they | |
| said | , | " | does | n't | charge | $ | 10 | " | . |

Tokenization in NLTK

```
>>> help(word_tokenize)
```

More at http://www.nltk.org/book/ch03.html

# Tokenization: language issues

- **Chinese**: no space between words

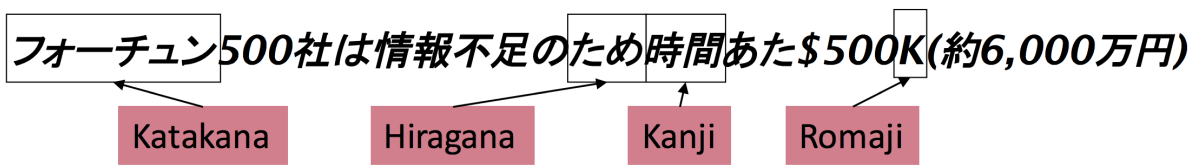
  E.g. *计算语言学课程是三个学时 → 计算语言学 课程 是 三 个 学时*

  Read: Word Segmentation in Chinese: the MaxMatch algorithm (J+M_2)

- **German**: noun compounds not segmented

  E.g. *Lebensversicherungsgesellschaftsangestellter 'life insurance company employee'*

- **Japanese**: multiple alphabets intermingled




- …

# Lemmatization

- Choose a single normalized form for words with multiple forms

  E.g. *USA, U.S.A., US; uh-huh, uhhuh*

- Reduce all letters to lower case (case folding)

  E.g. *US, us*

- Represent words by their lemmas
  - Simply reduce inflections or variant forms to base form

    E.g. *am, are, is* → *be; dinner, dinners* → *dinner*

  - Perform complete morphological parsing

    E.g. *fox* → *fox; cats* → *cat* + *-s*

# Lemmatization

Dealing with morphological variation in word forms:

- Finite-state transducers: full morphological parsing
- Stemmer: simpler but cruder chopping off of affixes
  - The Porter stemmer
  - The Lancaster stemmer

More at http://www.nltk.org/book/ch03.html

# Sentence segmentation

- Punctuations:
  - Relatively unambiguous: question marks, exclamation points
  - Quite ambiguous: periods
- Binary classifier
  - Hand-written rules
  - Machine-learning

  + Abbreviation dictionary

# Text normalization

## What every NLP task needs to do!

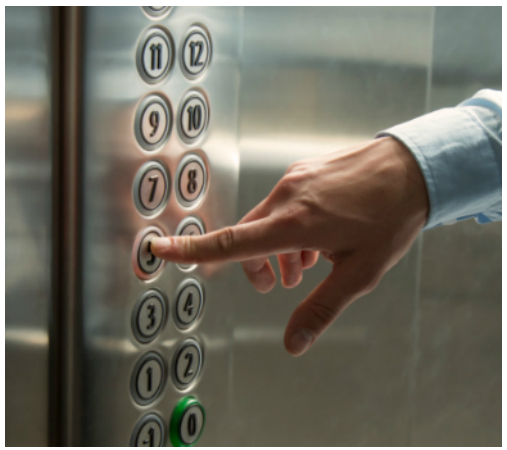
1. Tokenize/segment words in running text
2. Lemmatize word forms
3. Segment sentences in running text

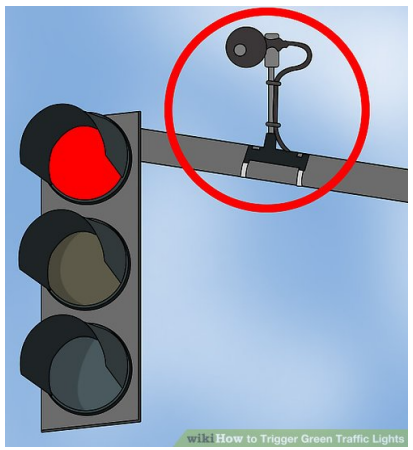
**Question:** How could NLTK help? >>> Practical 2

# Modelling *actions* and *events*

Situations with a predefined set of actions

Examples: lift, traffic lights with motion sensors, vending machines



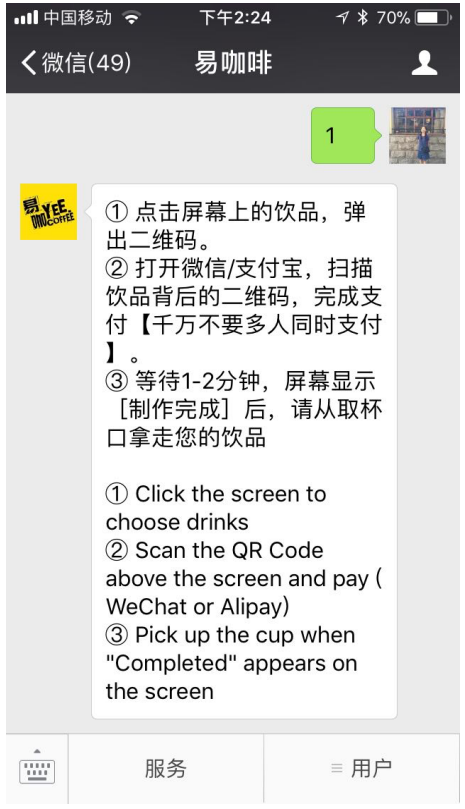




The order of actions depends on events happening at the time.

# Modelling *actions* and *events*: finite state automaton (FSA)

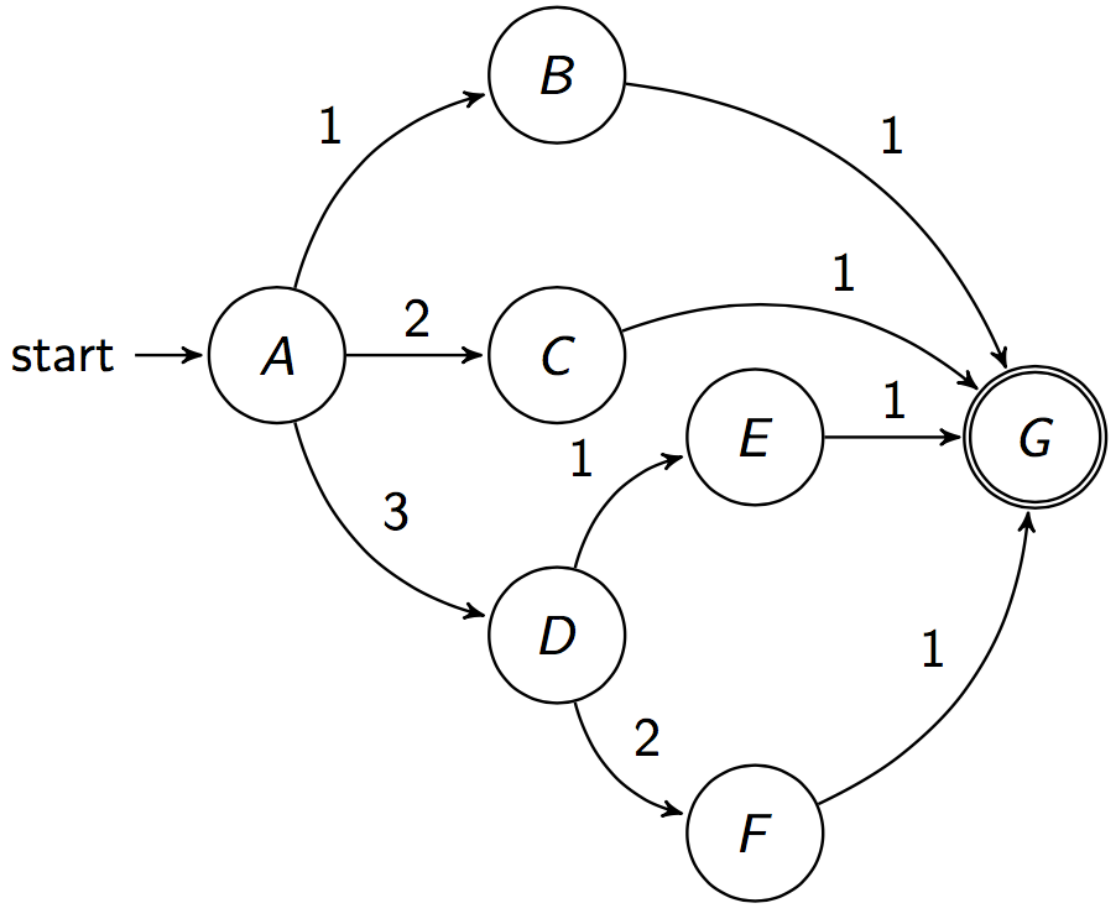




# A toy example of FSA: Tele-fruit-sales

Based on the wordy description of what happens when you call the Tele-fruit-sales line, can you represent all the information in a simpler way?

# A toy example of FSA: Tele-fruit-sales




| | 1 | 2 | 3 |
|---|---|---|---|
| A | B | C | D |
| B | G | - | - |
| C | G | - | - |
| D | E | F | - |
| E | G | - | - |
| F | G | - | - |
| G | - | - | - |

# FSA for recognizing language

- The sheep talk

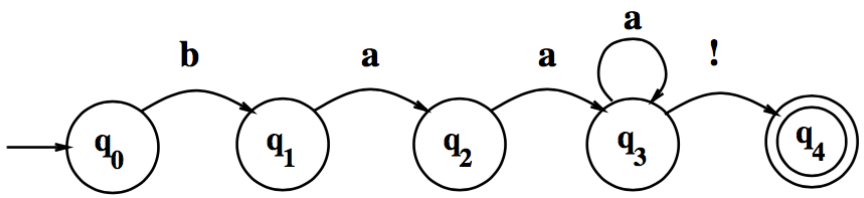
  baa!
  baaa!
  baaaa!
  baaaaa!
  ...

# FSA for recognizing language

- The sheep talk

  baa!
  baaa!
  baaaa!
  baaaaa!
  ...

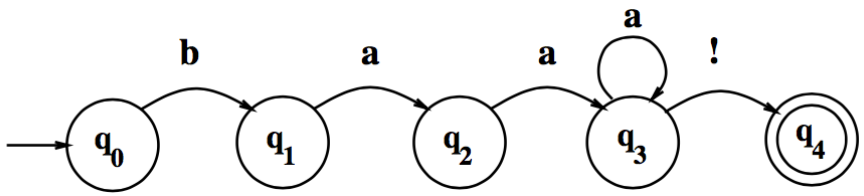
- FSA for the sheep talk




| | Input | | |
|---|---|---|---|
| State | b | a | ! |
| 0 | 1 | Ø | Ø |
| 1 | Ø | 2 | Ø |
| 2 | Ø | 3 | Ø |
| 3 | Ø | 3 | 4 |
| 4: | Ø | Ø | Ø |

# FSA for recognizing language

- FSA for the sheep talk




| | Input | | |
|---|---|---|---|
| State | b | a | ! |
| 0 | 1 | ∅ | ∅ |
| 1 | ∅ | 2 | ∅ |
| 2 | ∅ | 3 | ∅ |
| 3 | ∅ | 3 | 4 |
| 4: | ∅ | ∅ | ∅ |

$Q = \{q_0, q_1, q_2, \ldots q_{N-1}\}$: a finite set of $N$ **states**

$\Sigma$: a finite **input alphabet** of symbols

$q_0$: the **start state**

$F$: the set of **final states**, $F \subseteq Q$

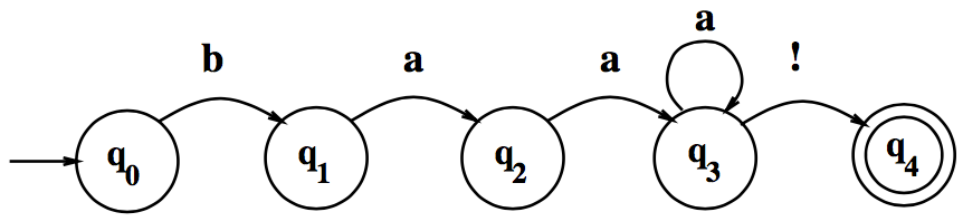
$\delta(q, i)$: the **transition function** or **transition matrix** between states. Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\delta(q, i)$ returns a new state $q' \in Q$.

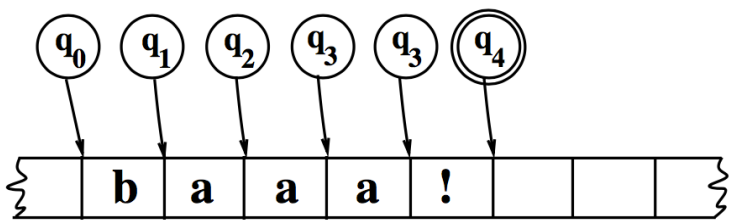
# FSA for recognizing language

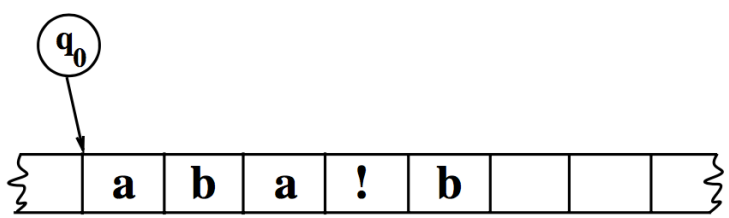
- The sheep talk

  baa!
  baaa!
  baaaa!
  baaaaa!
  ...

- FSA for the sheep talk



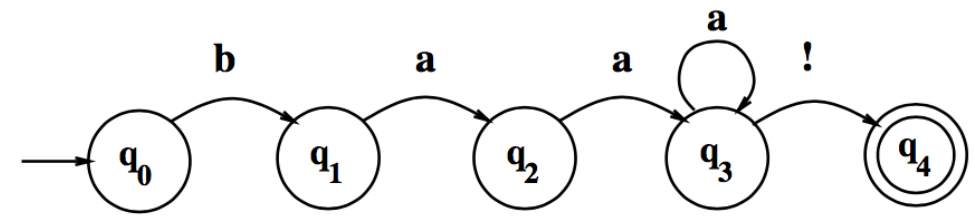



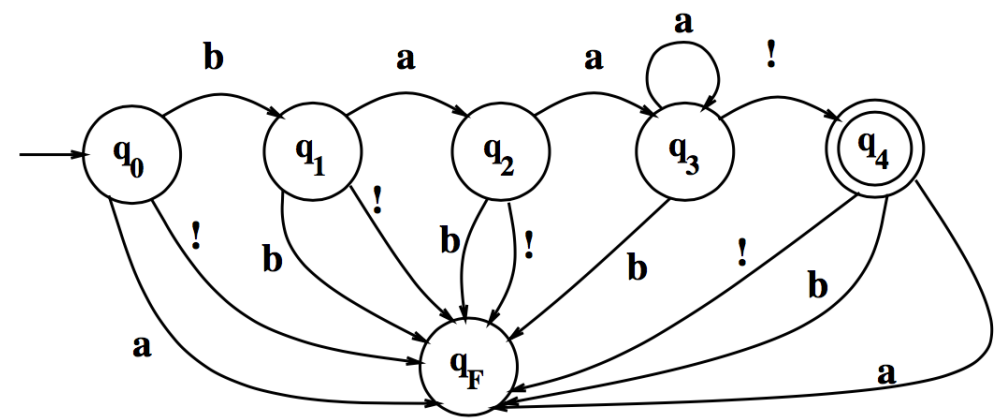
# FSA for recognizing language

- The sheep talk

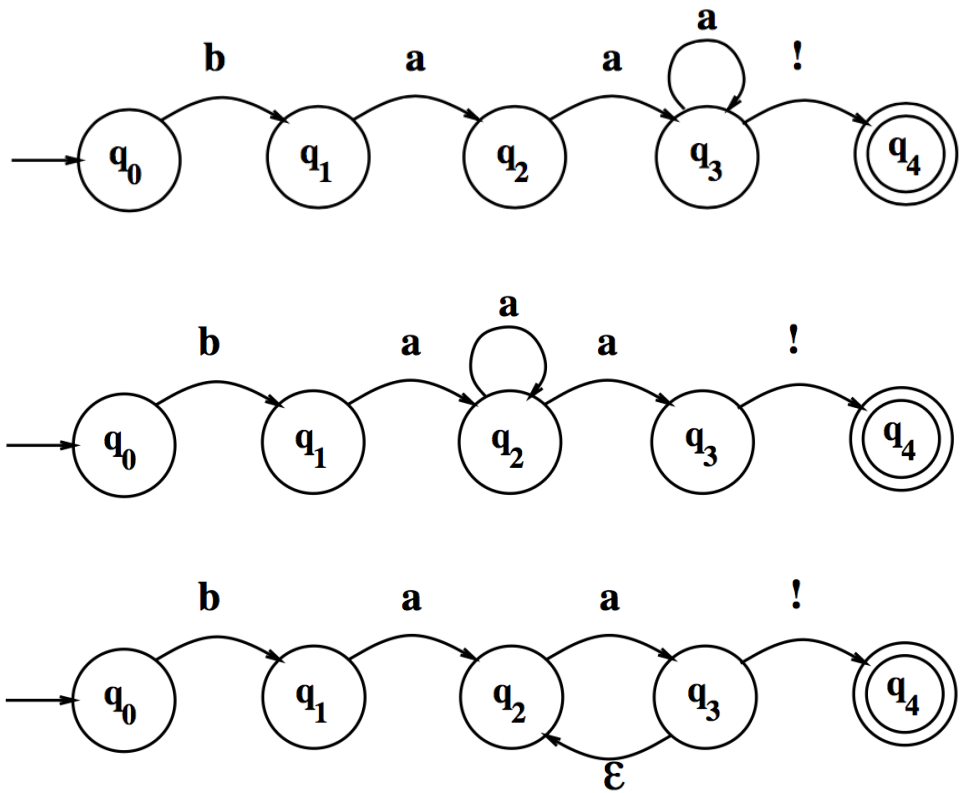
  baa!
  baaa!
  baaaa!
  baaaaa!
  ...

- FSA for the sheep talk



# Deterministic FSA vs. non-deterministic FSA




- In an DFSA, a state $q_i$ has only one possible next state given the input $i$.
- In an NFSA, a state $q_i$ may have more than one possible next state given an input $i$.
- For any NFSA, there is an exactly equivalent DFSA.

# How does a computer use an FSA?

A pseudo-code example

```
function D-RECOGNIZE(tape, machine) returns accept or reject
    index←Beginning of tape
    current-state←Initial state of machine
    loop
        if End of input has been reached then
          if current-state is an accept state then
              return accept
          else
              return reject
        elsif transition-table[current-state,tape[index]] is empty then
          return reject
        else
          current-state ← transition-table[current-state,tape[index]]
          index←index + 1
    end
```

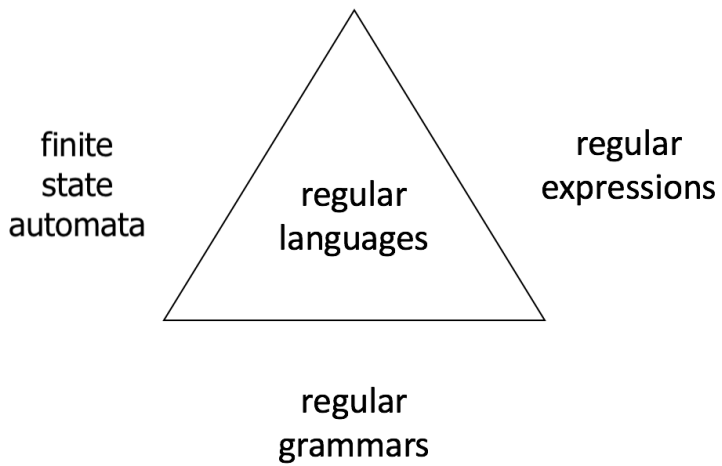
# Yet another formal description of the sheep talk

- The sheep talk

  baa!
  baaa!
  baaaa!
  baaaaa!
  …

- RE for the sheep talk

  /baa+!/

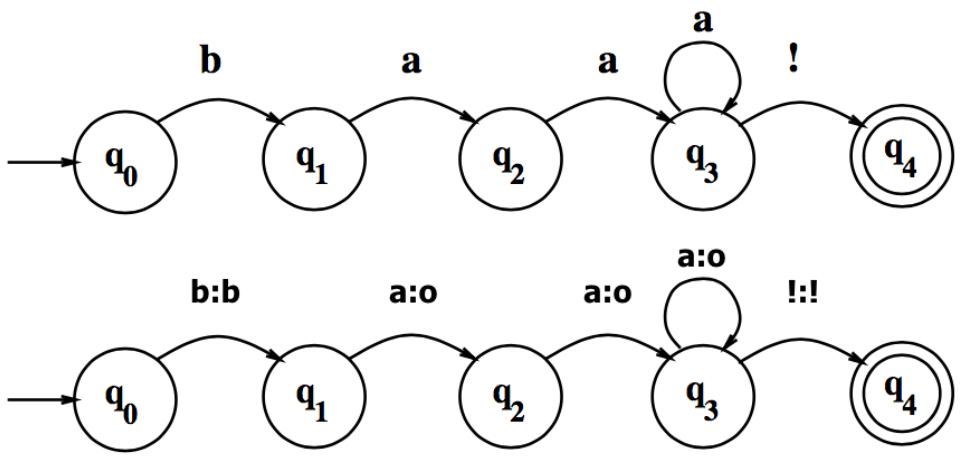



Three equivalent ways of describing regular languages

More to be learned in Lecture 3 …

# Finite state transducer (FST)

- Finite-state morphological parsing
- An augmentation to FSA
- Used to map between representations (e.g. from /baa+!/ to /boo+!/)



# FST: a formal definition

$Q$: a finite set of $N$ **states**

$\{q_0, q_1, q_2, \ldots q_{N-1}\}$

$\Sigma$: a finite **input alphabet** of symbols

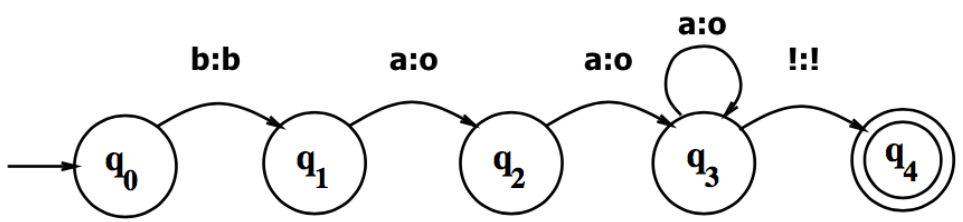
$\Delta$: a finite **output alphabet** of symbols

$q_0$: the **start state**

$F$: the set of **final states**, $F \subseteq Q$

$\delta(q, i)$: the **transition function**. Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\delta(q, i)$ returns a set of new states, each state $q' \in Q$.

$\sigma(q, i)$: the **output function**. Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\sigma(q, i)$ returns a set of output symbols, each symbol $o \in \Delta$ .



# FST: a formal definition

$Q=\{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma=\{b,a,!\}$

$\Delta=\{b,o,!\}$

$q_0=q_0$

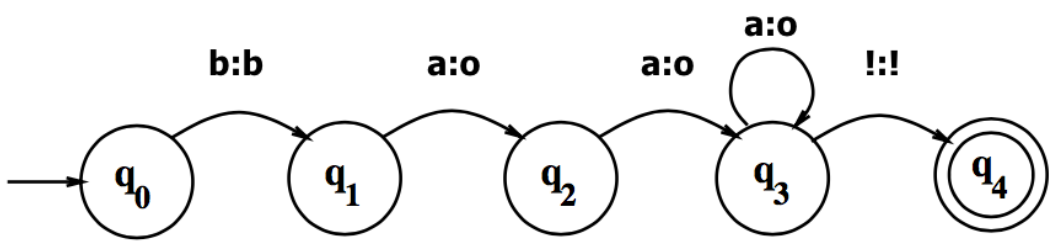
$F=\{q4\}$

$\delta(q,i) =$

| | $b$ | $a$ | $!$ |
|---|---|---|---|
| $q_0$ | $q_1$ | – | – |
| $q_1$ | – | $q_2$ | – |
| $q_2$ | – | $q_3$ | – |
| $q_3$ | – | $q_3$ | $q_4$ |
| $q_4$ | – | – | – |

$\sigma(q,i) =$

| | $b$ | $a$ | $!$ |
|---|---|---|---|
| $q_0$ | $b$ | – | – |
| $q_1$ | – | $o$ | – |
| $q_2$ | – | $o$ | – |
| $q_3$ | – | $o$ | ! |
| $q_4$ | – | – | – |



# Morphological parsing

Example: Searching for singulars and plurals for English words

*woodchuck*|*woodchucks*

*fox*|*foxes*; *fish*|*fish*; *peccuary*|*peccuaries*; *goose*|*geese*

# Morphological parsing

Example: Searching for singulars and plurals for English words

*woodchuck*|*woodchucks*

*fox*|*foxes*; *fish*|*fish*; *peccuary*|*peccuaries*; *goose*|*geese*

## Parsing

Take an input and produce some sort of linguistic structure for it

- Morphological, syntactic, semantic, discourse
- A string, or a tree, or a network

# Morphological parsing

English morphology

- Morphemes: stems and affixes
- Types of affixes
- Ways of combining morphemes to form words

| | Regular Nouns | | Irregular Nouns | |
|---|---|---|---|---|
| **Singular** | cat | thrush | mouse | ox |
| **Plural** | cats | thrushes | mice | oxen |

| Morphological Class | Regularly Inflected Verbs | | | |
|---|---|---|---|---|
| stem | walk | merge | try | map |
| *-s* form | walks | merges | tries | maps |
| *-ing* participle | walking | merging | trying | mapping |
| Past form or *-ed* participle | walked | merged | tried | mapped |

| Suffix | Base Verb/Adjective | Derived Noun |
|---|---|---|
| -ation | computerize (V) | computerization |
| -ee | appoint (V) | appointee |
| -er | kill (V) | killer |
| -ness | fuzzy (A) | fuzziness |

| Suffix | Base Noun/Verb | Derived Adjective |
|---|---|---|
| -al | computation (N) | computational |
| -able | embrace (V) | embraceable |
| -less | clue (N) | clueless |

# Morphological parsing

Uses

- Input for other parsings, esp. syntactic parsing
- Web search, spell checking, …

# Morphological parsing

| **Input** | **Morphological Parse** |
|---|---|
| cat | *cat* + *N* + *Sing* |
| cats | *cat* + *N* + *PL* |
| hope | *hope* + *V* |
| hopes | *hope* + *V* + *3P* + *Sing* |
| fox | *fox* + *N* + *Sing* |
| fox | *fox* + *V* |
| foxes | *fox* + *N* + *PL* |
| foxes | *fox* + *V* + *3P* + *Sing* |
| foxed | *fox* + *V* + *PastPart* |

| **Stem** | **Category** | **Affix** | **Category** |
|---|---|---|---|
| cat | *N* | ^s | *PL* |
| hope | *V* | ^s | + *3P* + *Sing* |
| fox | *N* | ^ed | + *PastPart* |
| fox | *V* | | |

cats → *cat*^*s* → *cat* + *N* + *PL*

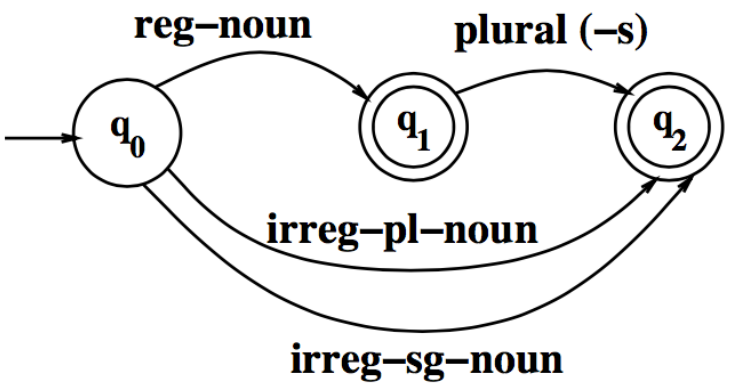
foxed → *fox*^*ed* → *fox* + *V* + *PastPart*

# Lexions and morphotactic FSAs

**Lexion**: a list of the stems and affixes of a language
**morphotactics**: a model to show how the stems and affixes can fit together

| reg-noun | irreg-sg-noun | irreg-pl-noun | plural |
|---|---|---|---|
| fox | goose | geese | s |
| cat | sheep | sheep | |
| dog | mouse | mice | |

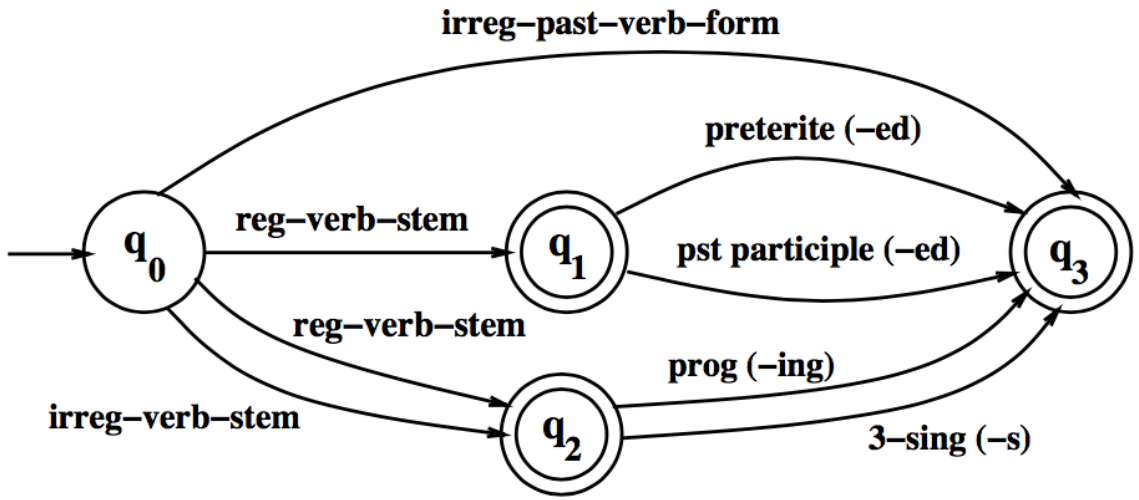
A morphotactic FSA for English nominal inflection



# Lexions and morphotactic FSAs

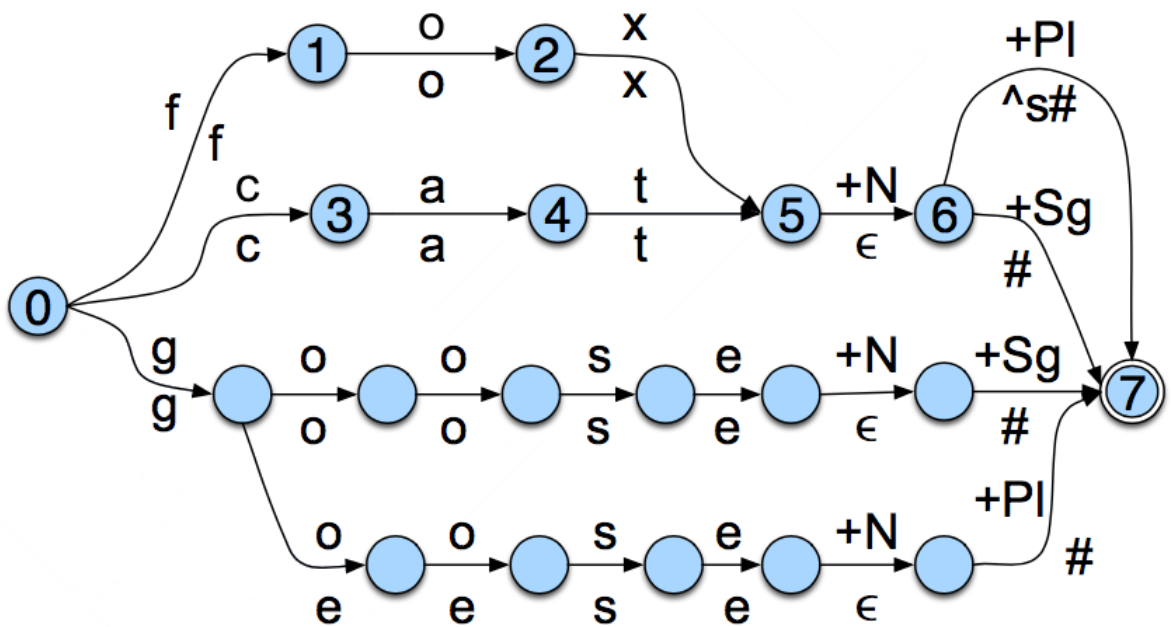
| reg-verb | irreg-verb | irreg-past | past | past-part | pres-part | 3sg |
|---|---|---|---|---|---|---|
| walk | cut | caught | -ed | -ed | -ing | -s |
| fry | speak | ate | | | | |
| talk | sing | eaten | | | | |
| impeach | | sang | | | | |

A morphotactic FSA for English verbal inflection



# FSTs for morphological parsing

| **reg-noun** | **irreg-pl-noun** | **irreg-sg-noun** |
|---|---|---|
| fox | g o:e o:e s e | goose |
| cat | sheep | sheep |
| aardvark | m o:i u:ɛ s:c e | mouse |




cat → *cat* + *N* + *Sg*

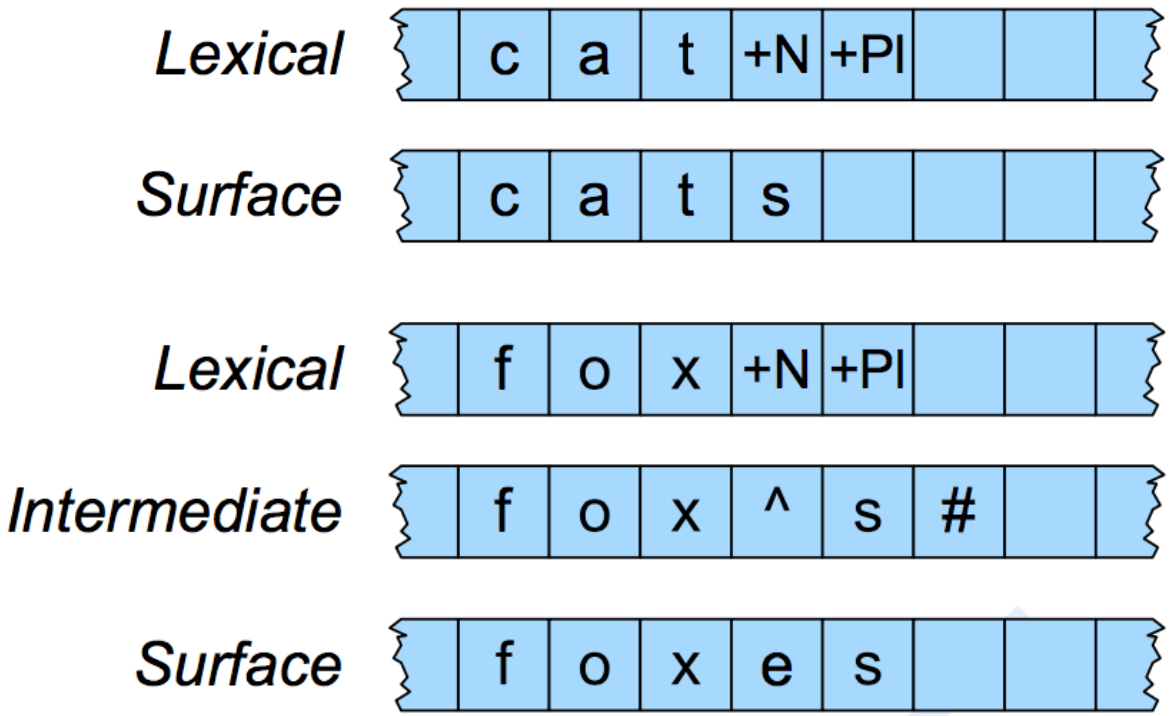
cat^s → *cat* + *N* + *PL*

geese → *goose* + *N* + *PL*

fox^s → *fox* + *N* + *PL*

# FSTs for morphological parsing

**Orthographic rules**: a model to show the changes that occur in a word

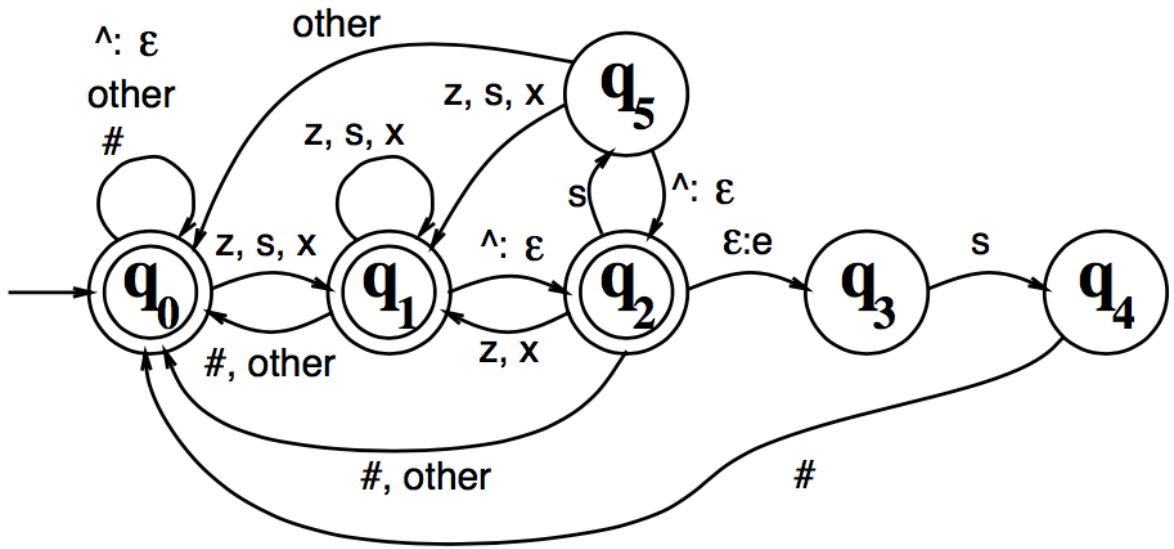



$$\epsilon \rightarrow e \,/\, \left\{ \begin{matrix} x \\ s \\ z \end{matrix} \right\} \hat{} \_\_ \; s\#$$

# FSTs for morphological parsing

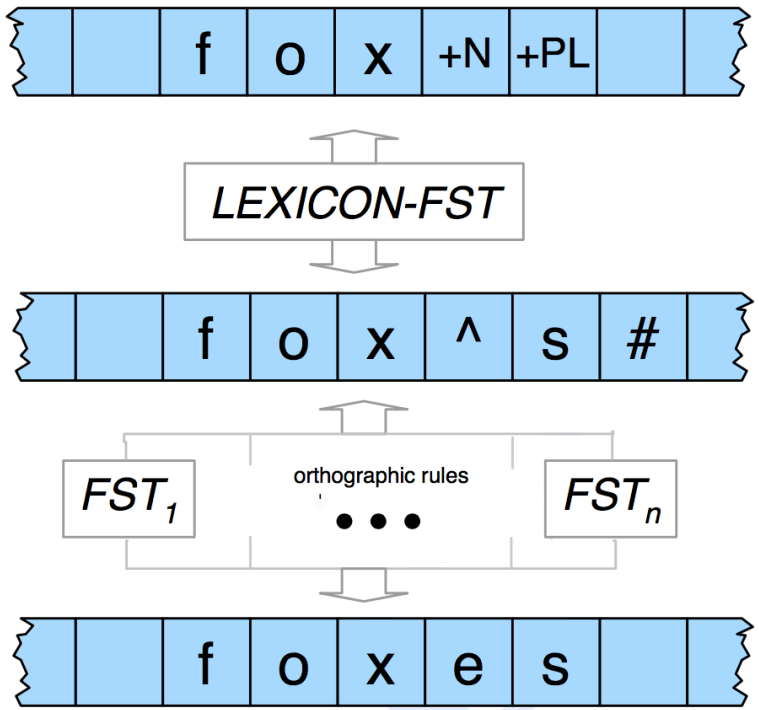
$$\epsilon \rightarrow \text{e} / \left\{ \begin{matrix} x \\ s \\ z \end{matrix} \right\} \hat{} \_\_ \text{s\#}$$

An FST for the
E-insertion rule



# Morphological parsing with FST lexion and rules

- **Lexion**: a list of the stems and affixes of a language
- **morphotactics**: a model to show how the stems and affixes can fit together
- **Orthographic rules**: a model to show the changes that occur in a word



# At the end of this session you will

- learn that corpora are important linguistic data for NLP;
- know the basic tasks in text processing;
- understand what are finite state automata and finite state transducers;
- understand how a finite state transducer is used in morphological parsing;
- learn Python functions and the functions in NLKT for text normalization.

# Homework

- Review: (Quiz 2 on Oct. 10, 2018)
  - J+M_2 (2.2, 2.3, 2.4)
  - J+M_second_edition_2 (2.2)
- Read:
  - J+M_2 (2.1, 2.5)
  - J+M_second_edition_3 (3.1)
- Read and practice:
  - http://www.nltk.org/book/ch01.html
  - http://www.nltk.org/book/ch02.html
  - http://www.nltk.org/book/ch03.html

# Next session

Regular Expressions and Edit Distance