NLP 201: Tokenization

Jeffrey Flanigan
University of California Santa Cruz
jmflanig@ucsc.edu

Many slides from Diyi Yang

Fall 2023

Outline

- Applications of FSAs and Regex
- Text normalization and tokenization

Example applications of FSAs

- Regular expressions
  - String search
  - Tokenization (in NLP 220)
- Finite state dialog systems

Simple Application: ELIZA

- Early NLP system that imitated a Rogerian psychotherapist
  - Joseph Weizenbaum, 1966.

- Uses pattern matching to match, e.g.,:
  - “I need X”

and translates them into, e.g.

  - “What would it mean to you if you got X?

Simple Application: ELIZA

Men are all alike.
IN WHAT WAY

They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE

Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE

He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED

How ELIZA works

- s/.* I'M (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
- s/.* I AM (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
- s/.* all .*/IN WHAT WAY?/
- s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE?/

Demo: ELIZA

http://psych.fullerton.edu/mbirnbaum/psych101/eliza.htm

Words

- I like the Piedmont Park in Atlanta
- ttyl, lol
- This no there is a typo
- 欢迎来到亚特兰大 (“Welcome to Atlanta”)

Punctuation

- Punctuation can be important
  - Signals boundaries (sentence, clausal boundaries, etc)
  - Has illocutionary force, like exclamation points (!) and question marks (?)
- Emoticons are strong signals of sentiment

How many words in a sentence?

- "I do uh main- mainly business data processing"
  - Fragments, filled pauses
- ”Emily’s cat in the hat is different from other cats!"
  - **Lemma**: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

How many words in a sentence?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.

- How many?
  - 15 tokens (or 14)
  - 13 types (or 12) (or 11?)

How many words in a corpus?

N = number of tokens

V = vocabulary = set of types, ***|V|*** is size of vocabulary

	Tokens = N	**Types = |V|**
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13+ million

Text Normalization

Every NLP task requires text normalization:

1. Tokenizing (segmenting) words
2. Normalizing word formats
3. Segmenting sentences

Issues in tokenization

Can't just blindly remove punctuation:

- m.p.h., Ph.D., AT&T, cap'n
- prices ($45.55)
- dates (01/02/06)
- URLs (http://gatech.edu)
- hashtags (#nlproc)
- email addresses (someone@cs.colorado.edu)

Clitic: a word that doesn't stand on its own

- "are" in we're, French "je" in j'ai, "le" in l'honneur

When should multiword expressions (MWE) be words?

- New York, rock 'n' roll

Tokenization in NLTK

Bird, Loper and Klein (2009), *Natural Language Processing with Python*. O'Reilly

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+        # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*        # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [][.,;"'?():-_`]  # these are separate tokens; includes ], [
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

Word Normalization

- Putting words/tokens in a standard format
  - U.S.A. or USA
  - uhhuh or uh-huh
  - Fed or fed
  - am, is, be, are

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***

- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)

Lemmatization

Represent all words as their lemma, their shared root
= dictionary headword form:

- *am, are, is* → *be*
- *car, cars, car's, cars'* → *car*
- *He is reading detective stories*
  → *He be read detective story*

Stemming

- Reduce terms to stems, chopping off affixes crudely

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.

Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note
.

Porter Stemmer

- Based on a series of rewrite rules run in series
  - A cascade, in which output of each pass fed to next pass
- Some sample rules:

$$\begin{aligned} \text{ATIONAL} &\rightarrow \text{ATE} \quad (\text{e.g., relational} \rightarrow \text{relate}) \\ \text{ING} &\rightarrow \epsilon \quad \text{if stem contains vowel (e.g., motoring} \rightarrow \text{motor)} \\ \text{SSES} &\rightarrow \text{SS} \quad (\text{e.g., grasses} \rightarrow \text{grass}) \end{aligned}$$

Dealing with complex morphology is necessary for many languages

e.g., the Turkish word:

Uygarlastiramadiklarimizdanmissinizcasina

`*(behaving) as if you are among those whom we could not civilize*'

- Uygar `civilized' + las `become'
  + tir `cause' + ama `not able'
  + dik `past' + lar 'plural'
  + imiz 'p1pl' + dan 'abl'
  + mis 'past' + siniz '2pl' + casina 'as if'

FSA for Dialog (Jurafsky & Martin)

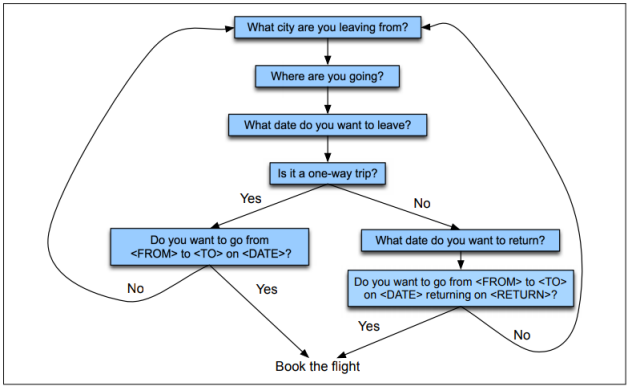

Figure 24.9 A simple finite-state automaton architecture for frame-based dialog.

FSA for Dialog (Flight booking system, Levinson 2005)

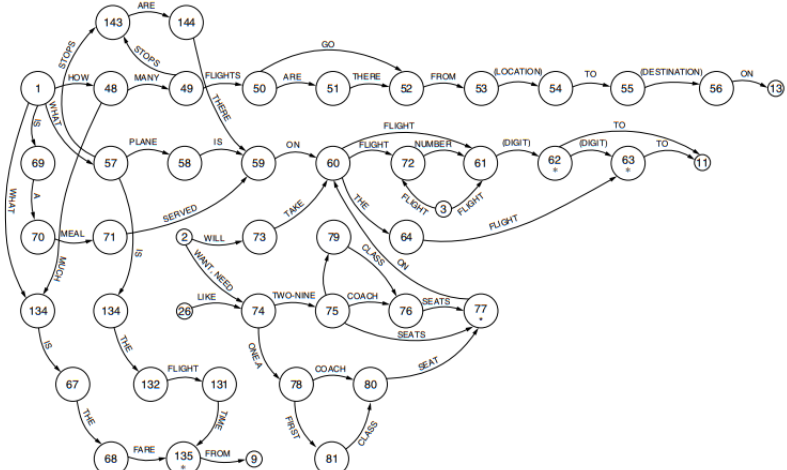

Figure 8.2 Formal language used by the semantic interpreter

FSA for Dialog (Flight booking system, Levinson 2005)

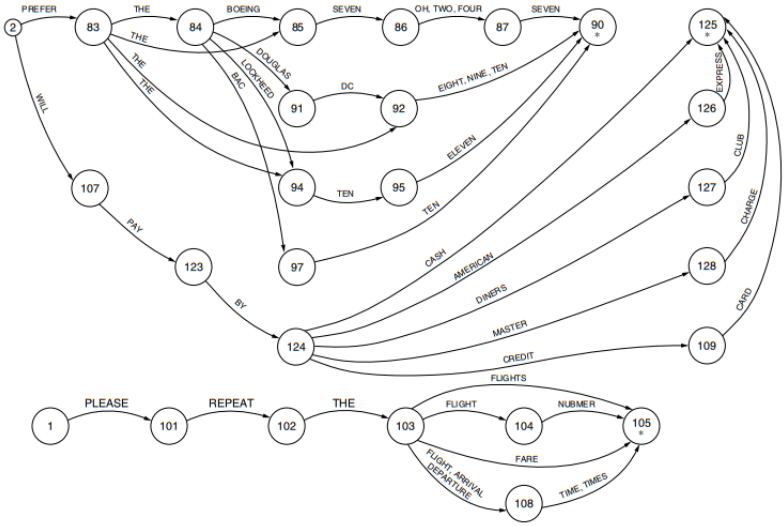

Figure 8.2 (*continued*)

FSA for Dialog (Flight booking system, Levinson 2005)

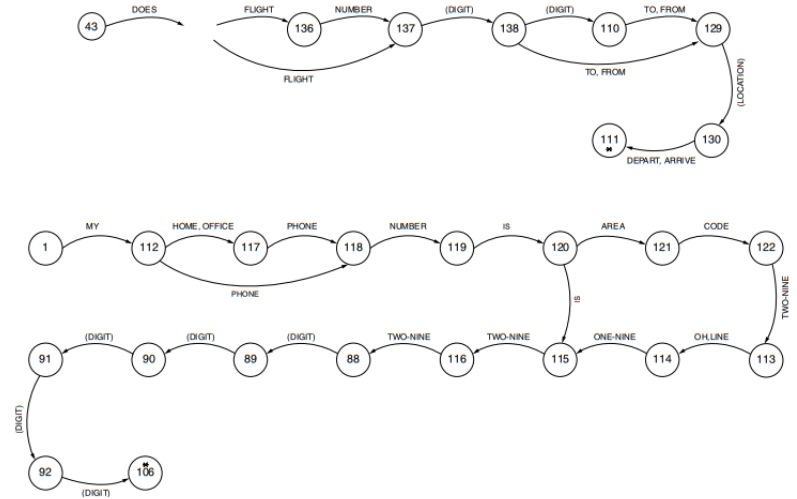

Figure 8.2 (*continued*)

FSA for Dialog (Flight booking system, Levinson 2005)

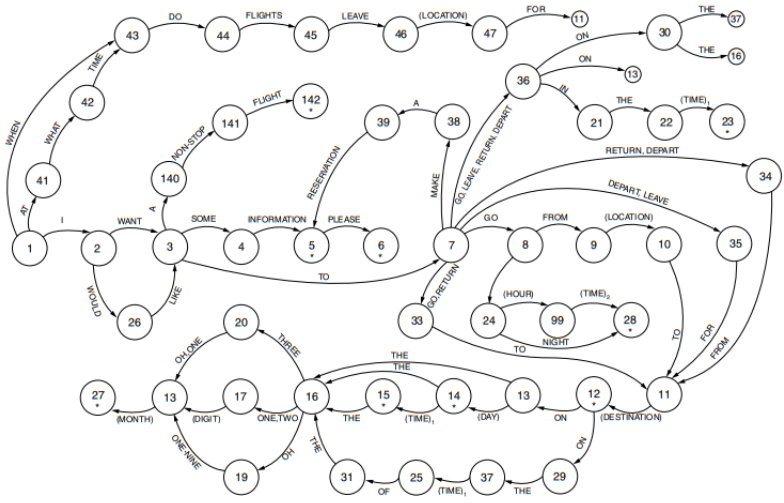

Figure 8.2 (*continued*)

So We Have Talked about Words.
How About Subword ?

Byte Pair Encoding

Instead of

- white-space segmentation
- single-character segmentation

Use the data to tell us how to tokenize.

Subword tokenization (because tokens can be parts of words as well as whole words)

Subword tokenization

Three common algorithms:

- **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
- **Unigram language modeling tokenization** (Kudo, 2018)
- **WordPiece** (Schuster and Nakajima, 2012)

All have 2 parts:

- A token **learner** that takes a raw training corpus and induces a vocabulary (a set of tokens).
- A token **segmenter** that takes a raw test sentence and tokenizes it according to that vocabulary

BPE Token Learner

Let vocabulary be the set of all individual characters

= {A, B, C, D,..., a, b, c, d....}

- Repeat:
  - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
  - Add a new merged symbol 'AB' to the vocabulary
  - Replace every adjacent 'A' 'B' in the corpus with 'AB'.
- Until *k* merges have been done.

BPE Token Learner Algorithm

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters
for $i = 1$ **to** k **do** # merge tokens til k times
  $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C
  $t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
  $V \leftarrow V + t_{NEW}$ # update the vocabulary
  Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus
return V

BPE Addendum

Most subword algorithms are run inside space-separated tokens.

So we commonly first add a special end-of-word symbol '__' before space in training corpus

Next, separate into letters.

BPE Token Learner

Original (very fascinating 🙄) corpus:

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

Add end-of-word tokens, resulting in this vocabulary:

vocabulary

_, d, e, i, l, n, o, r, s, t, w

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w
2 l o w e s t _	
6 n e w e r _	
3 w i d e r _	
2 n e w _	

Merge e r to er

corpus	vocabulary
5 l o w _	_, d, e, i, l, n, o, r, s, t, w, er
2 l o w e s t _	
6 n e w er _	
3 w i d er _	
2 n e w _	

corpus

5 l o w _
2 l o w e s t _
6 n e w er _
3 w i d er _
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er

Merge er _ to er_

corpus

5 l o w _
2 l o w e s t _
6 n e w er_
3 w i d er_
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

corpus		vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2	l o w e s t _	
6	n e w er_	
3	w i d er_	
2	n e w _	

Merge n e to ne

corpus		vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne
2	l o w e s t _	
6	ne w er_	
3	w i d er_	
2	ne w _	

BPE Token Learner

The next merges are:

Merge	Current Vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

BPE Token Segmenter Algorithm

On the test data, run each merge learned from the training data:

- Greedily
- In the order we learned them
- (test frequencies don't play a role)

Result:

- Test set "n e w e r _" would be tokenized as a full word
- Test set "l o w e r _" would be two tokens: "low er_"