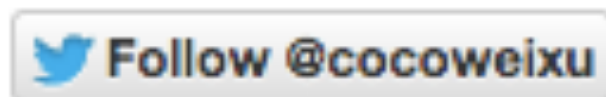


Social Media & Text Analysis

lecture 7 - Twitter NLP Pipeline

Tokenization, Normalization, POS/NE Tagging

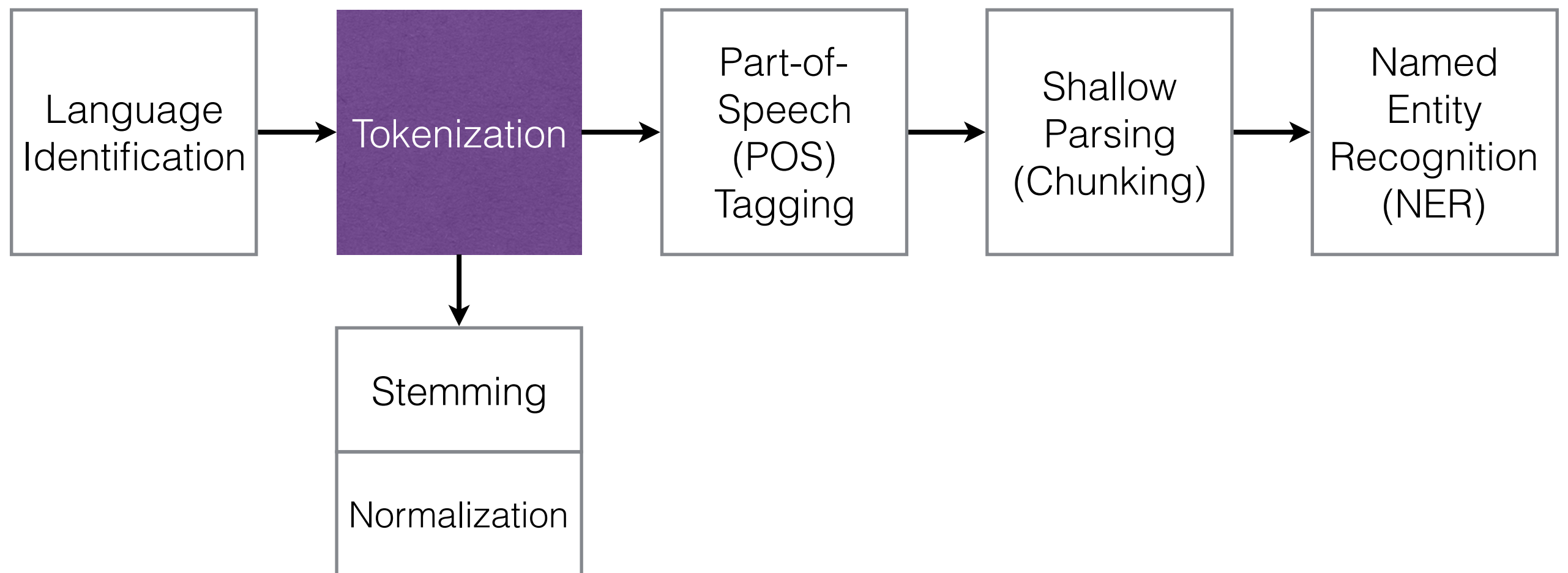


CSE 5539-0010 Ohio State University

Instructor: Wei Xu

Website: socialmedia-class.org

NLP Pipeline



Tokenization

- breaks up the string into words and punctuation
- need to handle:
 - abbreviations (“jr.”), number (“5,000”) ...

```
seas479:training weixu$ ./penn-treebank-tokenizer.perl
Tokenizer v3
Language: en
```

```
Ms. Hilton last year called Mr. Rothschild "the love of my life."
```

```
Ms. Hilton last year called Mr. Rothschild " the love of my life . "
```

← **input**
← **output**

Tokenization


- for Twitter, additionally need to handle:
 - emoticons, urls, #hashtags, @mentions ...


```
>>> import twokenize
>>> input = "Clowns are pretty gross tho O.o (I'm afraid of clowns :p) ask.fm/a/cc301167"
>>> twokenize.tokenizeRawTweetText(input)
['Clowns', 'are', 'pretty', 'gross', 'tho', 'O.o', '(', '"I'm", 'afraid', 'of', 'clowns', ':p', ')', 'ask.fm/a/cc301167']
```

← input


← output

Tool: twokenize.py



 This repository Search Pull requests Issues Gist

branch: master ark-twokenize-py / twokenize.py

 myleott on Apr 29, 2013 Initial commit

1 contributor

Executable File | 300 lines (247 sloc) | 12.993 kB  

```
1 # -*- coding: utf-8 -*-
2 """
3 Twokenize -- a tokenizer designed for Twitter text in English and some other European languages.
```

Tool: twokenize.py

```
3 Twokenize -- a tokenizer designed for Twitter text in English and some other European languages.
4 This tokenizer code has gone through a long history:
5
6 (1) Brendan O'Connor wrote original version in Python, http://github.com/brendano/tweetmotif
7     TweetMotif: Exploratory Search and Topic Summarization for Twitter.
8     Brendan O'Connor, Michel Krieger, and David Ahn.
9     ICWSM-2010 (demo track), http://brenocon.com/oconnor\_krieger\_ahn.icwsm2010.tweetmotif.pdf
10 (2a) Kevin Gimpel and Daniel Mills modified it for POS tagging for the CMU ARK Twitter POS Tagger
11 (2b) Jason Baldridge and David Snyder ported it to Scala
12 (3) Brendan bugfixed the Scala port and merged with POS-specific changes
13     for the CMU ARK Twitter POS Tagger
14 (4) Tobi Owoputi ported it back to Java and added many improvements (2012-06)
15
16 Current home is http://github.com/brendano/ark-tweet-nlp and http://www.ark.cs.cmu.edu/TweetNLP
```

Tokenization

- main techniques:
 - hand-crafted rules as regular expressions

Regular Expression

- a pattern matching language
- invented by American Mathematician Stephen Kleene in the 1950s
- used for search, find, replace, validation ... (very frequently used when dealing with strings)
- supported by most programming languages
- easy to learn, but hard to master

Regular Expression

147	Hashtag = "[a-zA-Z0-9_]+"
-----	---------------------------

- [] indicates a set of characters:
 - [amk] will match 'a', 'm', or 'k'
 - [a-z] will match any lowercase letter ('abcdefghijklmnopqrstuvwxyz')
 - [a-zA-Z0-9_] will match any letter or digit or '_'
- + matches 1 or more repetitions of preceding RE

Regular Expression

147	Hashtag = "[a-zA-Z0-9_]+"
-----	---------------------------

- will match strings that:
 - start with a '#'
 - follow with one or more letters/digits/'_'

Regular Expression

147	Hashtag = "[a-zA-Z0-9_]+"
-----	---------------------------

```
>>> import re
>>> Hashtag = "[a-zA-Z0-9_]+"
>>> hashtagpattern = re.compile(Hashtag)
>>> hashtagpattern.findall("So that's what #StarWars")
['#StarWars']
```

Regular Expression

133	Hearts = "(?:<+/?3+)+"
-----	------------------------

- will match strings that:
 - start with one or more '<'
 - then maybe a '/'
 - then one or more '3'
 - and maybe repetitions of the above

Regular Expression

133	Hearts = "(?:<+/?3+)+"
-----	------------------------

- ‘+’ matches 1 or more repetitions of the preceding RE
 - ‘<+’ matches ‘<’, ‘<<’, ‘<<<’ ...
 - ‘3+’ matches ‘3’, ‘33’, ‘333’ ...
- ‘?’ matches 0 or 1 repetitions of the preceding RE
 - ‘/?’ matches ‘/’ or nothing (so handles ‘</3’)
- (?: ...) is a non-capturing version of (...)
- (...) matches whatever RE is inside the parentheses

Regular Expression

133	Hearts = "(?:<+/?3+)+"
-----	------------------------

```
>>> import re
>>> Hearts = "(?:<+/?3+)+"
>>> heartspattern = re.compile(Hearts)
>>> heartspattern.findall("I <3 u <3<333333")
['<3', '<3<333333']
>>> heartspattern.findall("sooo sad </3")
['</3']
```

Regular Expression

133	Hearts = "(?:<+/?3+)+"
-----	------------------------

```
Python 2.7.10 (default, Feb  7 2017, 00:08:15)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> import re
>>> heart1 = "<+/?3+)"
>>> heartpattern1 = re.compile(heart1)
>>> heartpattern1.findall("I <3 u <3<333")
['<3', '<333']
>>>
>>> heart2 = "(?:<+/?3+)+"
>>> heartpattern2 = re.compile(heart2)
>>> heartpattern2.findall("I <3 u <3<333")
['<3', '<3<333']
>>>
>>> █
```


Regular Expression

- learn more (<https://docs.python.org/2/library/re.html>)

Python » 2.7.10 Documentation » The Python Standard Library » 7. String Services » previous | next | modules | index

Table Of Contents

- 7.2. `re` — Regular expression operations
 - 7.2.1. Regular Expression Syntax
 - 7.2.2. Module Contents
 - 7.2.3. Regular Expression Objects
 - 7.2.4. Match Objects
 - 7.2.5. Examples
 - 7.2.5.1. Checking For a Pair
 - 7.2.5.2. Simulating `scanf()`
 - 7.2.5.3. `search()` vs. `match()`
 - 7.2.5.4. Making a Phonebook
 - 7.2.5.5. Text Munging
 - 7.2.5.6. Finding all Adverbs
 - 7.2.5.7. Finding all Adverbs and their Positions
 - 7.2.5.8. Raw String Notation

7.2. `re` — Regular expression operations

This module provides regular expression matching operations similar to those found in Perl. Both patterns and strings to be searched can be Unicode strings as well as 8-bit strings.

Regular expressions use the backslash character (`'\'`) to indicate special forms or to allow special characters to be used without invoking their special meaning. This collides with Python's usage of the same character for the same purpose in string literals; for example, to match a literal backslash, one might have to write `'\\\\'` as the pattern string, because the regular expression must be `\\`, and each backslash must be expressed as `\\` inside a regular Python string literal.

The solution is to use Python's raw string notation for regular expression patterns; backslashes are not handled in any special way in a string literal prefixed with `'r'`. So `r"\n"` is a two-character string containing `'\'` and `'n'`, while `"\n"` is a one-character string containing a newline. Usually patterns will be expressed in Python code using this raw string notation.

It is important to note that most regular expression operations are available as module-level functions and `RegexObject` methods. The functions are shortcuts that don't require you to compile a regex object first, but miss some fine-tuning parameters.

7.2.1. Regular Expression Syntax

Tokenization

- for Twitter, additionally need to handle:
 - emoticons, urls, #hashtags, @mentions ...

```
>>> import twokenize
>>> input = "Clowns are pretty gross tho O.o (I'm afraid of clowns :p) ask.fm/a/cc301167"
>>> twokenize.tokenizeRawTweetText(input)
['Clowns', 'are', 'pretty', 'gross', 'tho', 'O.o', '(', '"I'm", 'afraid', 'of', 'clowns', ':p', ')', 'ask.fm/a/cc301167']
```

← input

← output

Emoticons

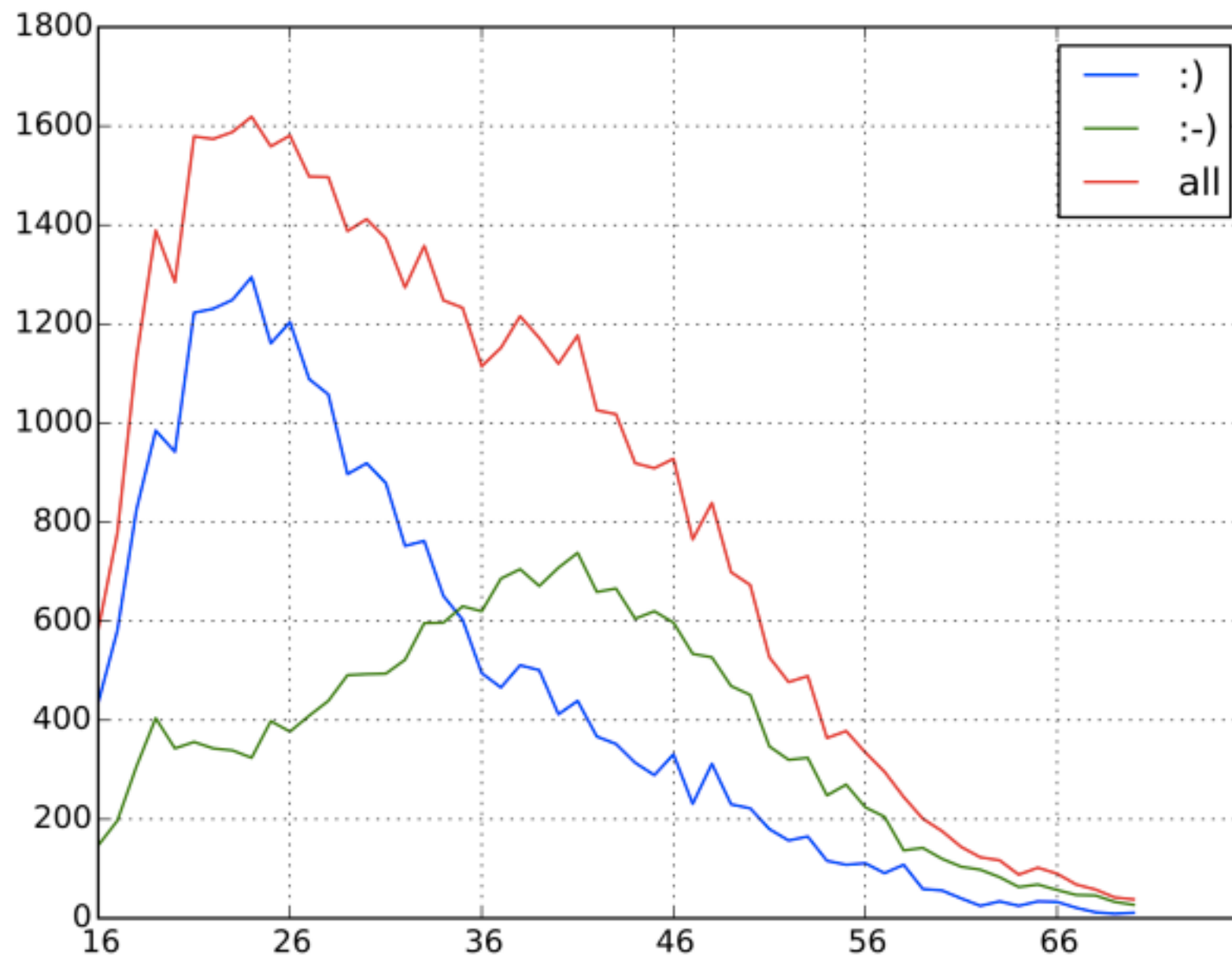


Figure 3: Usage of emoticons with and without nose by age group, aggregated over all countries

Dirk Hovy, Anders Johannsen, and Anders Søgaard.

User review sites as a resource for large-scale sociolinguistic studies. WWW, 2015

Emoticons

With respect to gender, we find that women tend to use the noseless variant significantly more than men, except for France, where the difference between genders is not statistically significant at the chosen level.

country	AGE		GENDER significant
	Spearman ρ	significant	
Denmark	0.89	yes	yes
France	0.63	yes	no
Germany	0.83	yes	yes
UK	0.83	yes	yes
US	0.82	yes	yes

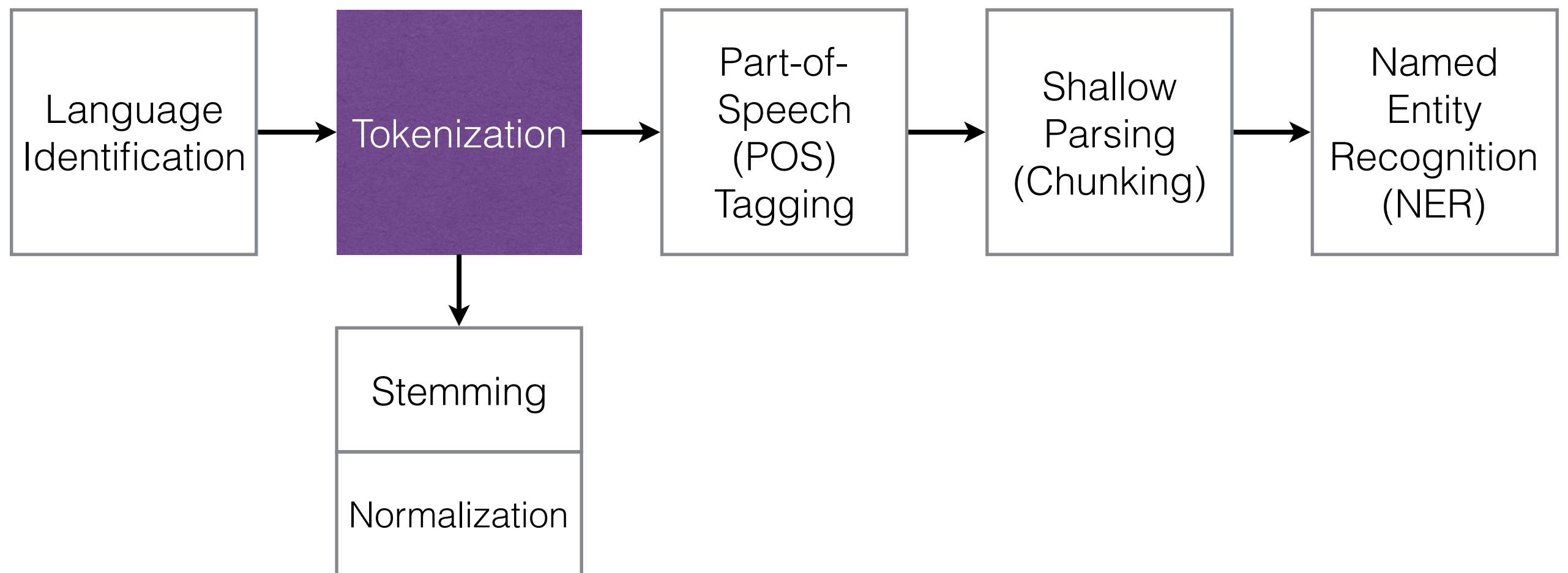
Tokenization

- language dependent

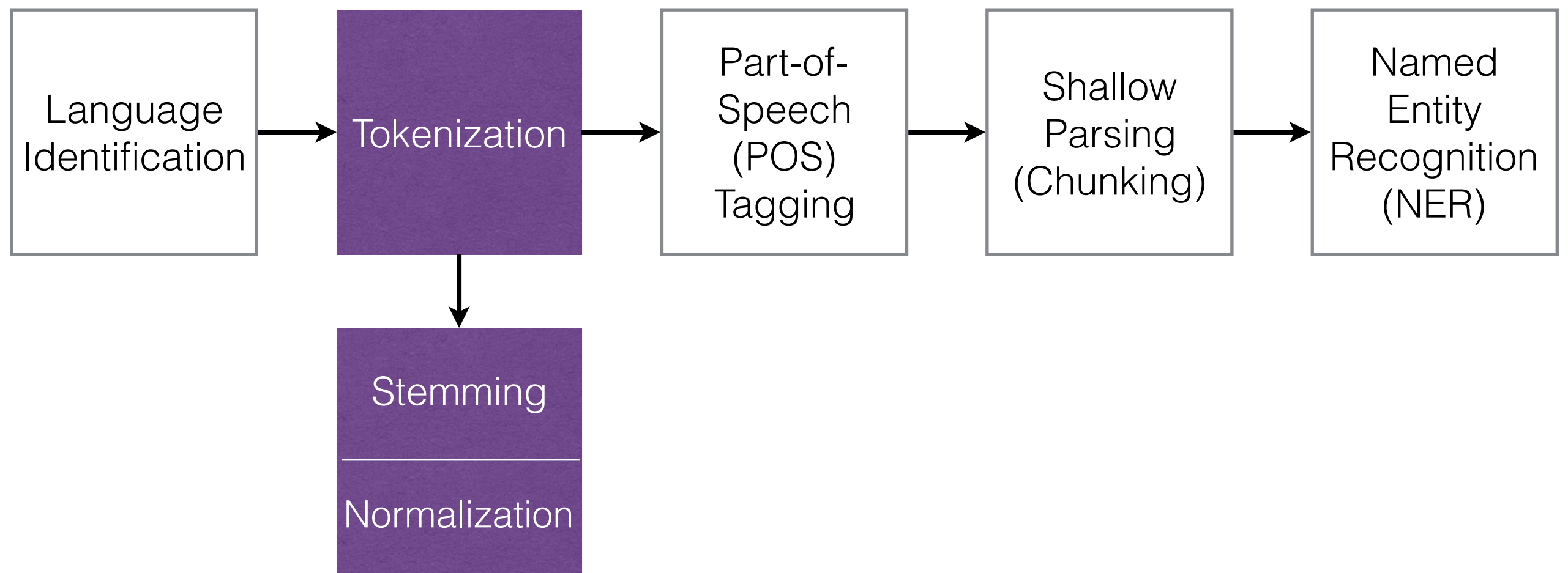
下雨天留客天留我不留	Unpunctuated Chinese sentence
下雨、天留客。天留、我不留！	<i>It is raining, the god would like the guest to stay. Although the god wants you to stay, I do not!</i>
下雨天、留客天。留我不？ 留！	<i>The rainy day, the staying day. Would you like me to stay? Sure!</i>

我喜欢新西兰花	Unsegmented Chinese sentence
我 喜欢 新西兰 花	<i>I like New Zealand flowers</i>
我 喜欢 新 西兰花	<i>I like fresh broccoli</i>

NLP Pipeline



NLP Pipeline



Stemming

- reduce inflected words to their word stem, base or root form (not necessarily the morphological root)
- studied since the 1960s

```
>>> from nltk.stem.porter import PorterStemmer
>>> stemmer = PorterStemmer()
>>> stemmer.stem('automate')
'autom'
>>> stemmer.stem('automates')
'autom'
>>> stemmer.stem('automation')
'autom'
```

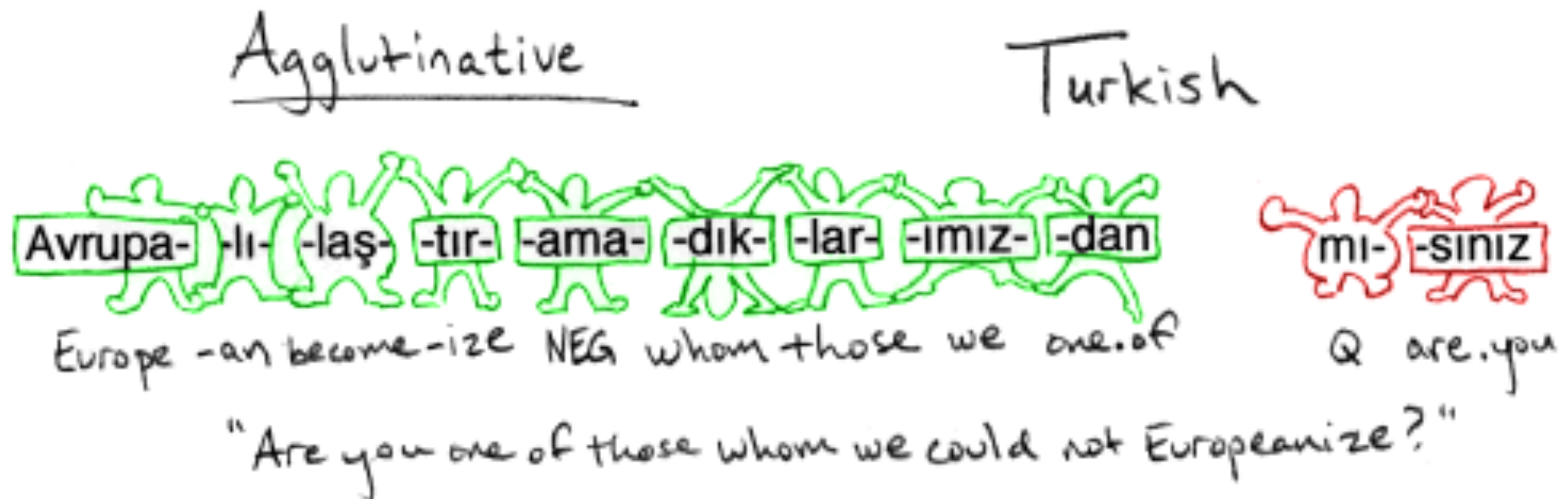

Stemming

- different steamers: Porter, Snowball, Lancaster ...
- WordNet's built-in lemmatized (dictionary-based)

```
>>> from nltk.stem import WordNetLemmatizer
>>> wordnet_lemmatizer = WordNetLemmatizer()
>>> wordnet_lemmatizer.lemmatize('leaves', pos='n')
'leaf'
>>> wordnet_lemmatizer.lemmatize('leaves', pos='v')
'leave'
```


Stemming

- language dependent



Text Normalization

- convert non-standard words to standard

Original tweet

@USER, r u cuming 2 MidCorner dis Sunday?

Normalized tweet

@USER, are you coming to MidCorner this Sunday?

Original tweet

Still have to get up early 2mr thou 😞 so Gn 😴

Normalized tweet

Still have to get up early tomorrow though 😞 so Good night 😴

Source: Tim Baldwin, Marie de Marneffe, Han Bo, Young-Bum Kim, Alan Ritter, Wei Xu
Shared Tasks of the 2015 Workshop on Noisy User-generated Text:
Twitter Lexical Normalization and Named Entity Recognition

Text Normalization

- types of non-standard words in 449 English tweets:

Category	Ratio	Example
letter&numer	2.36%	b4 → before
letter	72.44%	shuld → should
number substitution	2.76%	4 → for
slang	12.20	lol → laugh out loud
other	10.24%	sucha → such a

most non-standard words are morphophonemic “errors”

Source: Bo Han and Timothy Baldwin

“Lexical normalisation of short text messages: Makn sens a #twitter” ACL 2011

A Normalization Lexicon

- automatically derived from Twitter data + dictionary

41169	costumess	costumes
41170	nywhere	anywhere
41171	sandwich	sandwich
41172	aleksander	alexander
41173	juns	jun
41174	showi	showing
41175	washinq	washing
41176	jscript	script
41177	fundin	funding
41178	itxted	fitted
41179	cheeeap	cheap
41180	fawesome	awesome
41181	untalented	talented
41182		

Performance

Precision = 0.847

Recall = 0.630

F1-Score = 0.723

Phrase-level Normalization

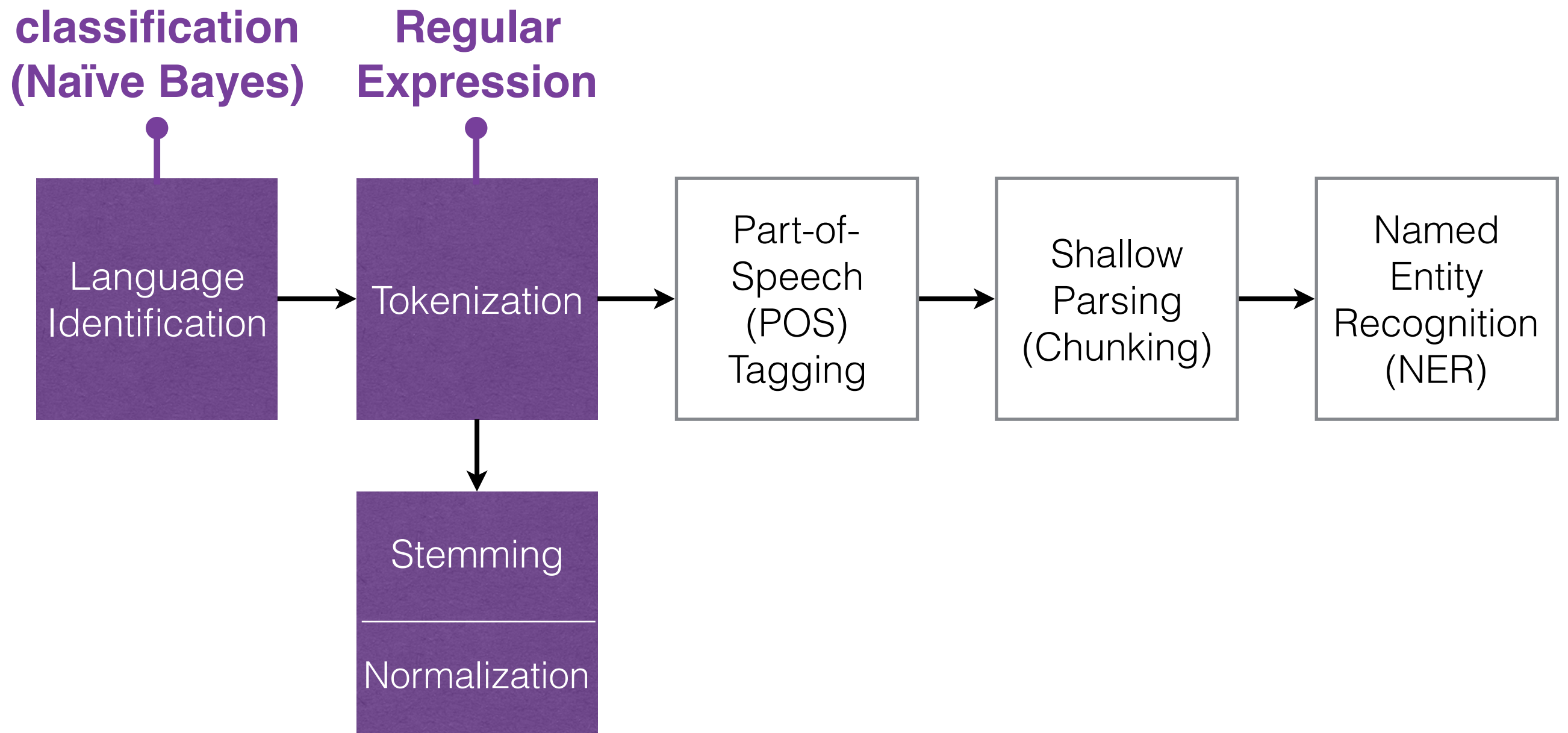
- word-level normalization is insufficient for many cases:

in-vocabulary words

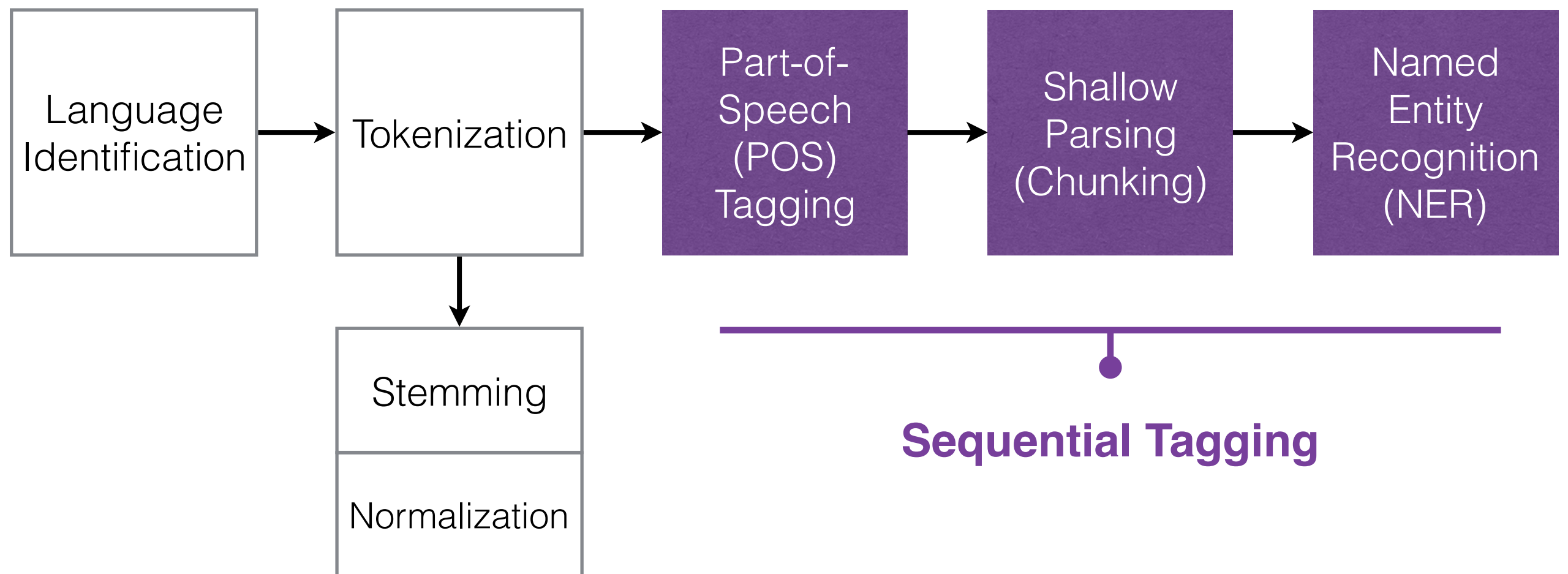
Category	Example
1-to-many	everytime → every time
incorrect IVs	can't want for → can't wait for
grammar	I'm going a movie → I'm going to a movie
ambiguities	4 → 4 / 4th / for / four

Source: Wei Xu, Alan Ritter, Ralph Grishman

NLP Pipeline (summary so far)



NLP Pipeline (next)



Part-of-Speech (POS) Tagging

Cant	MD
wait	VB
for	IN
the	DT
ravens	NNP
game	NN
tomorrow	NN
...	:
go	VB
ray	NNP
rice	NNP
!!!!!!!	.



Penn Treebank POS Tags

1. CC	Coordinating conjunction	25. TO	<i>to</i>
2. CD	Cardinal number	26. UH	Interjection
3. DT	Determiner	27. VB	Verb, base form
4. EX	Existential <i>there</i>	28. VBD	Verb, past tense
5. FW	Foreign word	29. VBG	Verb, gerund/present participle
6. IN	Preposition/subordinating conjunction	30. VBN	Verb, past participle
7. JJ	Adjective	31. VBP	Verb, non-3rd ps. sing. present
8. JJR	Adjective, comparative	32. VBZ	Verb, 3rd ps. sing. present
9. JJS	Adjective, superlative	33. WDT	<i>wh</i> -determiner
10. LS	List item marker	34. WP	<i>wh</i> -pronoun
11. MD	Modal	35. WP\$	Possessive <i>wh</i> -pronoun
12. NN	Noun, singular or mass	36. WRB	<i>wh</i> -adverb
13. NNS	Noun, plural	37. #	Pound sign
14. NNP	Proper noun, singular	38. \$	Dollar sign
15. NNPS	Proper noun, plural	39. .	Sentence-final punctuation
16. PDT	Predeterminer	40. ,	Comma
17. POS	Possessive ending	41. :	Colon, semi-colon
18. PRP	Personal pronoun	42. (Left bracket character
19. PP\$	Possessive pronoun	43.)	Right bracket character
20. RB	Adverb	44. "	Straight double quote
21. RBR	Adverb, comparative	45. '	Left open single quote
22. RBS	Adverb, superlative	46. "	Left open double quote
23. RP	Particle	47. '	Right close single quote
24. SYM	Symbol (mathematical or scientific)	48. "	Right close double quote

Part-of-Speech (POS) Tagging

- Words often have more than one POS:
 - The back door = JJ
 - On my back = NN
 - Win the voters back = RB
 - Promised to back the bill = VB
- POS tagging problem is to determine the POS tag for a particular instance of a word.

Twitter-specific Tags

- #hashtag
- @mention
- url
- email address
- emoticon
- discourse marker
- symbols
- ...



Source: Gimpel et al.

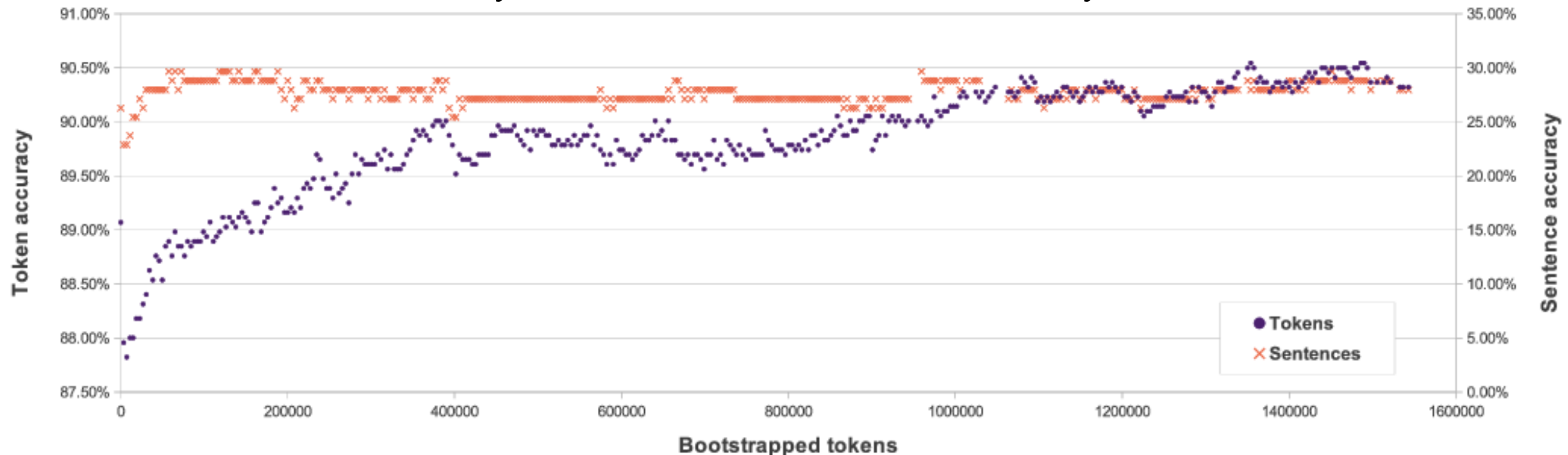
“Part-of-Speech Tagging for Twitter : Annotation, Features, and Experiments” ACL 2011

Notable Twitter POS Taggers

- Gimpel et al., 2011
- Ritter et al., 2011
- Derczynski et al, 2013
- Owoputi et al. 2013

State-of-the-art:
Token Accuracy: ~ 88% Sentence Accuracy ~20%

(97% on news text)



Source: Derczynski, Ritter, Clark, Bontcheva

Chunking

Cant	VP
wait	
for	PP
the	NP
ravens	
game	NP
tomorrow	
...	
go	VP
ray	NP
rice	
!!!!!!!	



Chunking

- recovering phrases constructed by the part-of-speech tags
- a.k.a shallow (partial) parsing:
 - full parsing is expensive, and is not very robust
 - partial parsing can be much faster, more robust, yet sufficient for many applications
 - useful as input (features) for named entity recognition or full parser

Named Entity Recognition(NER)

Cant	
wait	
for	
the	
ravens	ORG
game	
tomorrow	
...	
go	
ray	
rice	PER
!!!!!!!	.



ORG: organization

PER: person

LOC: location

NER: Basic Classes

Cant	
wait	
for	
the	
ravens	ORG
game	
tomorrow	
...	
go	
ray	
rice	PER
!!!!!!!	.



ORG: organization

PER: person

LOC: location

NER: Rich Classes

sportsteam sportsteam geo-loc
India vs Australia 2014-15 , 4th Test in Sydney

company product
Samsung to launch Galaxy S6 in March

tvshow tvshow
New Suits and Brooklyn Nine-Nine tomorrow ... Happy days

NER: Genre Differences

	News	Tweets
PER	Politicians, business leaders, journalists, celebrities	Sportsmen, actors, TV personalities, celebrities, names of friends
LOC	Countries, cities, rivers, and other places related to current affairs	Restaurants, bars, local landmarks/areas, cities, rarely countries
ORG	Public and private companies, government organisations	Bands, internet companies, sports clubs

Notable Twitter NE Research


- Liu et al., 2011
- Ritter et al., 2011
- Owoputi et al. 2013
- Plank et al, 2014
- Cherry & Guo, 2015


System	P	R	F ₁
COTRAIN-NER (10 types)	0.55	0.33	0.41
T-NER(10 types)	0.65	0.42	0.51
COTRAIN-NER (PLO)	0.57	0.42	0.49
T-NER(PLO)	0.73	0.49	0.59
Stanford NER (PLO)	0.30	0.27	0.29

Table 12: Performance at predicting both segmentation and classification. Systems labeled with PLO are evaluated on the 3 MUC types *PERSON*, *LOCATION*, *ORGANIZATION*.

Tool: twitter_nlp


https://github.com/aritter/twitter_nlp

 This repository Search Pull requests Issues Gist


 aritter / twitter_nlp Watch 71 Star

Twitter NLP Tools

55 commits 2 branches 0 releases 1 contributor

 branch: master twitter_nlp / +

a few corrections to the NER annotation from Brendan 1 comment

 aritter authored on Nov 8, 2014 latest commit 27c8190084

data	a few corrections to the NER annotation from Brendan	8 months ago
hbc	added labels for weakly supervised NE categorization	2 years ago
lib	added README.md	3 years ago
mallet-2.0.6	re-importing to blow away some large files in the history	4 years ago
models	Fixed a bug in computing brown clusters reported by Yiye Ruan and Lu ...	a year ago

Tool: twitter_nlp



+ Follow

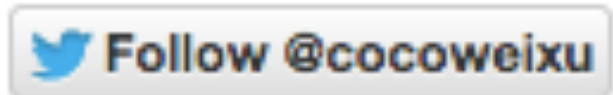
Had a great time in New York w my love :) !

```
xuwei@proteus100[twitter_nlp]$ export TWITTER_NLP=./
xuwei@proteus100[twitter_nlp]$
xuwei@proteus100[twitter_nlp]$ echo "Had a great time in New York w my
love :) ! " | python python/ner/extractEntities2.py

Had/O a/O great/O time/O in/O New/B-ENTITY York/I-ENTITY w/O my/O love/
O :)/O !/O
Average time per tweet = 3.04769945145s
xuwei@proteus100[twitter_nlp]$
xuwei@proteus100[twitter_nlp]$ echo "Had a great time in New York w my
love :) ! " | python python/ner/extractEntities2.py --pos --chunk

Had/O/VBD/B-VP a/O/DT/B-NP great/O/JJ/I-NP time/O/NN/I-NP in/O/IN/B-PP
New/B-ENTITY/NNP/B-NP York/I-ENTITY/NNP/I-NP w/O/IN/B-PP my/O/PRP$/B-NP
love/O/NN/I-NP :)/O/UH/B-INTJ !/O/./I-INTJ
Average time per tweet = 5.49846148491s
xuwei@proteus100[twitter_nlp]$ _
```

Thank You!



Instructor: Wei Xu

www.cis.upenn.edu/~xwe/

Course Website: socialmedia-class.org