Social Media & Text Analysis

lecture 7 - Twitter NLP Pipeline
Tokenization, Normalization, POS/NE Tagging

CSE 5539-0010 Ohio State University
Instructor: Alan Ritter
Website: socialmedia-class.org
LangID Tool: langid.py

• main techniques:

  - Multinominal Naïve Bayes

  - diverse training data from multiple domains (Wikipedia, Reuters, Debian, etc.)

  - plus feature selection using Information Gain (IG) to choose features that are informative about language, but not informative about domain
Naïve Bayes

• For a document $d$, find the most probable class $c$:

$$c_{MAP} = \arg \max_{c \in C} P(t_1, t_2, ..., t_n \mid c)P(c)$$

$$c_{NB} = \arg \max_{c \in C} P(c) \prod_{t_i \in d} P(t_i \mid c)$$

Source: adapted from Dan Jurafsky
LangID features

- n-grams features:
  - 1-gram:
    “the” “following” “Wikipedia” “en” “español” ...
  - 2-gram:
    “the following” “following is” “Wikipedia en” “en español” ...
  - 3-gram:
    ....

The following is a list of words that occur in both Modern English and Modern Spanish, but which are pronounced differently and may have different meanings in each language.

Wikipedia en español es la edición en idioma español de Wikipedia. Actualmente cuenta con 1 185 590 páginas válidas de contenido y ocupa el décimo puesto en esta estadística entre...
Correlated Features

• For example, for spam email classification, word “win” often occurs together with “free”, “prize”.

• Solution:
  - feature selection
  - or other models (e.g. logistic/softmax regression)
A Heuristic from Information Theory

• Let $X$ be a random variable

• The *surprise* of each value of $X$ is defined as:

$$S(X = x) = - \log P(X = x)$$

• Notes:
  
  • An event with probability 1 has 0 surprise
  
  • An event with probability 0 has infinite surprise
Entropy & Information Gain

- **Entropy** is a measure of disorder in a dataset (expected surprise)

\[ H(X) = - \sum_{i} P(x_i) \log P(x_i) \]
Entropy & Information Gain

- **Entropy** is a measure of disorder in a dataset (expected surprise)

\[ H(X) = -\sum_{i} P(x_i) \log P(x_i) \]
Entropy & Information Gain

- **Entropy** is a measure of disorder in a dataset (expected surprise)

\[ H(X) = - \sum_{i} P(x_i) \log P(x_i) \]

- **Conditional Entropy** quantifies the amount of information needed to describe the outcome of \( Y \) given that \( X \) is known.

\[ H(Y \mid X) = \sum_{i} P(x_i) H(Y \mid X = x_i) \]
Entropy & Information Gain

- **Entropy** is a measure of disorder in a dataset

\[ H(X) = - \sum_i P(x_i) \log P(x_i) \]

- **Information Gain** is a measure of the decrease in disorder achieved by partitioning the original data set.

\[ IG(Y \mid X) = H(Y) - H(Y \mid X) \]
Information Gain

\[
H(X) = -\sum_i P(x_i) \log P(x_i) \quad \quad \quad IG(Y \mid X) = H(Y) - H(Y \mid X)
\]
Information Gain

wealth values: poor rich

agegroup  10s  2507  3  \[ H(\text{wealth} | \text{agegroup} = 10s) = 0.0133271 \]
20s  11262  743  \[ H(\text{wealth} | \text{agegroup} = 20s) = 0.334906 \]
30s  9468  3461  \[ H(\text{wealth} | \text{agegroup} = 30s) = 0.838134 \]
40s  6738  3986  \[ H(\text{wealth} | \text{agegroup} = 40s) = 0.951961 \]
50s  4110  2509  \[ H(\text{wealth} | \text{agegroup} = 50s) = 0.957376 \]
60s  2245  809  \[ H(\text{wealth} | \text{agegroup} = 60s) = 0.834049 \]
70s  668  147  \[ H(\text{wealth} | \text{agegroup} = 70s) = 0.680882 \]
80s  115  16  \[ H(\text{wealth} | \text{agegroup} = 80s) = 0.535474 \]
90s  42  13  \[ H(\text{wealth} | \text{agegroup} = 90s) = 0.788941 \]

\[ H(\text{wealth}) = 0.793844 \quad H(\text{wealth} | \text{agegroup}) = 0.709463 \]

\[ IG(\text{wealth} | \text{agegroup}) = 0.0843813 \]

Source: Andrew Moore
Information Gain used for?

- choose features that are informative (most useful) for discriminating between the classes.

**Wealth**

\[ IG(\text{wealth}|\text{gender}) = 0.0366896 \]

\[ IG(\text{wealth}|\text{agegroup}) = 0.0843813 \]

**Longevity**

\[ IG(\text{LongLife} | \text{HairColor}) = 0.01 \]

\[ IG(\text{LongLife} | \text{Smoker}) = 0.2 \]

\[ IG(\text{LongLife} | \text{Gender}) = 0.25 \]

\[ IG(\text{LongLife} | \text{LastDigitOfSSN}) = 0.00001 \]
LangID Tool: langid.py

- feature selection using Information Gain (IG)

Source: Lui and Baldwin "langid.py: An Off-the-shelf Language Identification Tool" ACL 2012
LangID Tool: langid.py

- main advantages:
  - cross-domain (works on all kinds of texts)
  - works for Twitter (accuracy = 0.89)
  - fast (300 tweets/second — 24G RAM)
  - currently supports 97 language
  - retrainable

Source: Lui and Baldwin "langid.py: An Off-the-shelf Language Identification Tool" ACL 2012
Summary

Language Identification

Tokenization → Part-of-Speech (POS) Tagging → Shallow Parsing (Chunking) → Named Entity Recognition (NER)

- Stemming
- Normalization

classification (Naïve Bayes)
NLP Pipeline

Language Identification → Tokenization → Part-of-Speech (POS) Tagging → Shallow Parsing (Chunking) → Named Entity Recognition (NER)

- Stemming
- Normalization
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.”
```
Tokenization

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

```python
>>> import twokenize
>>> input = "Clowns are pretty gross tho 0.o (I'm afraid of clowns :p) ask.fm/a/cc301167"
>>> twokenize.tokenizeRawTweetText(input)
['Clowns', 'are', 'pretty', 'gross', 'tho', '0.o', '(', ',', 'I', 'afraid', 'of', 'clowns', ':p', ')', 'ask.fm/a/cc301167']
```
Tool: twokenize.py

```python
# -*- coding: utf-8 -*-

""

Twokenize -- a tokenizer designed for Twitter text in English and some other European languages.
```
Tool: twokenize.py

Twokenize -- a tokenizer designed for Twitter text in English and some other European languages. This tokenizer code has gone through a long history:

(1) Brendan O'Connor wrote original version in Python, http://github.com/brendano/tweetmotif
    TweetMotif: Exploratory Search and Topic Summarization for Twitter.
    Brendan O'Connor, Michel Krieger, and David Ahn.
(2a) Kevin Gimpel and Daniel Mills modified it for POS tagging for the CMU ARK Twitter POS Tagger
(2b) Jason Baldridge and David Snyder ported it to Scala
(3) Brendan bugfixed the Scala port and merged with POS-specific changes for the CMU ARK Twitter POS Tagger
(4) Tobi Owoputi ported it back to Java and added many improvements (2012-06)

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_]++"

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

\[ \text{Hearts} = "(?:\<+/\?3\+)+" \]

- 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

```
>>> 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

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.

```python
>>> 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)
Tokenization

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

<table>
<thead>
<tr>
<th>country</th>
<th>Spearman $\rho$</th>
<th>AGE significant</th>
<th>GENDER significant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denmark</td>
<td>0.89</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>France</td>
<td>0.63</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Germany</td>
<td>0.83</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UK</td>
<td>0.83</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>US</td>
<td>0.82</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
# Tokenization

- language dependent

<table>
<thead>
<tr>
<th>Unpunctuated Chinese sentence</th>
<th>Unsegmented Chinese sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is raining, the god would like the guest to stay. Although the god wants you to stay, I do not!</td>
<td>I like New Zealand flowers</td>
</tr>
<tr>
<td>The rainy day, the staying day. Would you like me to stay? Sure!</td>
<td>I like fresh broccoli</td>
</tr>
</tbody>
</table>

Source: [http://what-when-how.com](http://what-when-how.com)
NLP Pipeline

Language Identification → Tokenization → Part-of-Speech (POS) Tagging → Shallow Parsing (Chunking) → Named Entity Recognition (NER)

- Stemming
- Normalization
NLP Pipeline

1. Language Identification
2. Tokenization
3. Part-of-Speech (POS) Tagging
4. Shallow Parsing (Chunking)
5. Named Entity Recognition (NER)

- Stemming
- Normalization
Stemming

• reduce inflected words to their word stem, base or root form (not necessarily the morphological root)

• studied since the 1960s

```python
>>> 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)

```python
>>> 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

Agglutinative

Turkish

Avrupa-|-li(-laşı)-|tir-|ama-|dik-|lar-|imiz-|dan

Europe-an become-ize NEG whom those we one.of

"Are you one of those whom we could not Europeanize?"

Source: All Things Linguistic
Text Normalization

- convert non-standard words to standard

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:

<table>
<thead>
<tr>
<th>Category</th>
<th>Ratio</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>letter&amp;numer</td>
<td>2.36%</td>
<td>b4 → before</td>
</tr>
<tr>
<td>letter</td>
<td>72.44%</td>
<td>shuld → should</td>
</tr>
<tr>
<td>number substitution</td>
<td>2.76%</td>
<td>4 → for</td>
</tr>
<tr>
<td>slang</td>
<td>12.20%</td>
<td>lol → laugh out loud</td>
</tr>
<tr>
<td>other</td>
<td>10.24%</td>
<td>sucha → such a</td>
</tr>
</tbody>
</table>

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

Performance
Precision = 0.847
Recall = 0.630
F1-Score = 0.723
Phrase-level Normalization

- word-level normalization is insufficient for many cases:

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-to-many</td>
<td>everytime → every time</td>
</tr>
<tr>
<td>incorrect IVs</td>
<td>can’t want for → can’t wait for</td>
</tr>
<tr>
<td>grammar</td>
<td>I’m going a movie → I’m going to a movie</td>
</tr>
<tr>
<td>ambiguities</td>
<td>4 → 4 / 4th / for / four</td>
</tr>
</tbody>
</table>

in-vocabulary words

Source: Wei Xu, Alan Ritter, Ralph Grishman

“Gathering and Generating Paraphrases from Twitter with Application to Normalization” BUCC 2013
NLP Pipeline (summary so far)

classification (Naïve Bayes)

Language Identification

Tokenization

Part-of-Speech (POS) Tagging

Shallow Parsing (Chunking)

Named Entity Recognition (NER)

Regular Expression

Stemming

Normalization
NLP Pipeline (next)

Language Identification → Tokenization → Part-of-Speech (POS) Tagging → Shallow Parsing (Chunking) → Named Entity Recognition (NER)

Stemming
Normalization

Sequential Tagging
Part-of-Speech (POS) Tagging

<table>
<thead>
<tr>
<th>Word</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cant</td>
<td>MD</td>
</tr>
<tr>
<td>wait</td>
<td>VB</td>
</tr>
<tr>
<td>for</td>
<td>IN</td>
</tr>
<tr>
<td>the</td>
<td>DT</td>
</tr>
<tr>
<td>ravens</td>
<td>NNP</td>
</tr>
<tr>
<td>game</td>
<td>NN</td>
</tr>
<tr>
<td>tomorrow</td>
<td>NN</td>
</tr>
<tr>
<td>...</td>
<td>:</td>
</tr>
<tr>
<td>go</td>
<td>VB</td>
</tr>
<tr>
<td>ray</td>
<td>NNP</td>
</tr>
<tr>
<td>rice</td>
<td>NNP</td>
</tr>
<tr>
<td>!!!!!!!</td>
<td>.</td>
</tr>
</tbody>
</table>

Cant wait for the ravens game tomorrow....go ray rice!!!!!!!
# Penn Treebank POS Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordinating conjunction</td>
</tr>
<tr>
<td>CD</td>
<td>Cardinal number</td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
</tr>
<tr>
<td>EX</td>
<td>Existential <em>there</em></td>
</tr>
<tr>
<td>FW</td>
<td>Foreign word</td>
</tr>
<tr>
<td>IN</td>
<td>Preposition/subordinating conjunction</td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
</tr>
<tr>
<td>JJR</td>
<td>Adjective, comparative</td>
</tr>
<tr>
<td>JJS</td>
<td>Adjective, superlative</td>
</tr>
<tr>
<td>LS</td>
<td>List item marker</td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
</tr>
<tr>
<td>NN</td>
<td>Noun, singular or mass</td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper noun, plural</td>
</tr>
<tr>
<td>PDT</td>
<td>Predeterminer</td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
</tr>
<tr>
<td>PRP</td>
<td>Personal pronoun</td>
</tr>
<tr>
<td>PP$</td>
<td>Possessive pronoun</td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
</tr>
<tr>
<td>RBR</td>
<td>Adverb, comparative</td>
</tr>
<tr>
<td>RBS</td>
<td>Adverb, superlative</td>
</tr>
<tr>
<td>RP</td>
<td>Particle</td>
</tr>
<tr>
<td>SYM</td>
<td>Symbol (mathematical or scientific)</td>
</tr>
<tr>
<td>TO</td>
<td><em>to</em></td>
</tr>
<tr>
<td>UH</td>
<td>Interjection</td>
</tr>
<tr>
<td>VB</td>
<td>Verb, base form</td>
</tr>
<tr>
<td>VBD</td>
<td>Verb, past tense</td>
</tr>
<tr>
<td>VBG</td>
<td>Verb, gerund/present participle</td>
</tr>
<tr>
<td>VBN</td>
<td>Verb, past participle</td>
</tr>
<tr>
<td>VBP</td>
<td>Verb, non-3rd ps. sing. present</td>
</tr>
<tr>
<td>VBZ</td>
<td>Verb, 3rd ps. sing. present</td>
</tr>
<tr>
<td>WDT</td>
<td><em>wh</em>-determiner</td>
</tr>
<tr>
<td>WP</td>
<td><em>wh</em>-pronoun</td>
</tr>
<tr>
<td>WP$</td>
<td>Possessive <em>wh</em>-pronoun</td>
</tr>
<tr>
<td>WRB</td>
<td><em>wh</em>-adverb</td>
</tr>
<tr>
<td>#</td>
<td>Pound sign</td>
</tr>
<tr>
<td>$</td>
<td>Dollar sign</td>
</tr>
<tr>
<td>.</td>
<td>Sentence-final punctuation</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
</tr>
<tr>
<td>:</td>
<td>Colon, semi-colon</td>
</tr>
<tr>
<td>(</td>
<td>Left bracket character</td>
</tr>
<tr>
<td>)</td>
<td>Right bracket character</td>
</tr>
<tr>
<td>&quot;</td>
<td>Straight double quote</td>
</tr>
<tr>
<td>’</td>
<td>Left open single quote</td>
</tr>
<tr>
<td>“</td>
<td>Left open double quote</td>
</tr>
<tr>
<td>‘</td>
<td>Right close single quote</td>
</tr>
<tr>
<td>”</td>
<td>Right close double quote</td>
</tr>
</tbody>
</table>
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.

Source: adapted from Chris Manning
Twitter-specific Tags

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

Retweet construction:

RT @user1: I never bought candy bars from those kids on my doorstep so I guess they’re all in gangs now.

Twitter discourse marker

RT @user2: LMBO! This man filed an EMERGENCY Motion for Continuance on account of the Rangers game tonight! 😱Wow lmao

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
“Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data” RANLP 2013
Chunking

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

ORG: organization
PER: person
LOC: location
NER: Basic Classes

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

ORG: organization
PER: person
LOC: location
NER: Rich Classes

Source: Strauss, Toma, Ritter, de Marneffe, Xu
Results of the WNUT16 Named Entity Recognition Shared Task (WNUT@COLING 2016)
## NER: Genre Differences

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

Source: Kalina Bontcheva and Leon Derczynski
"Tutorial on Natural Language Processing for Social Media" EACL 2014
Notable Twitter NE Research

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

<table>
<thead>
<tr>
<th>System</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>COTRAIN-NER (10 types)</td>
<td>0.55</td>
<td>0.33</td>
<td>0.41</td>
</tr>
<tr>
<td>T-NER(10 types)</td>
<td>0.65</td>
<td>0.42</td>
<td>0.51</td>
</tr>
<tr>
<td>COTRAIN-NER (PLO)</td>
<td>0.57</td>
<td>0.42</td>
<td>0.49</td>
</tr>
<tr>
<td>T-NER(PLO)</td>
<td>0.73</td>
<td>0.49</td>
<td>0.59</td>
</tr>
<tr>
<td>Stanford NER (PLO)</td>
<td>0.30</td>
<td>0.27</td>
<td>0.29</td>
</tr>
</tbody>
</table>

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

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

- data
- hbc
- lib
- mallet-2.0.6
- models

- added labels for weakly supervised NE categorization
- added README.md
- re-importing to blow away some large files in the history
- Fixed a bug in computing brown clusters reported by Yiye Ruan and Lu ...

latest commit 27c8190084
Tool: twitter_nlp

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

Average time per tweet = 5.49846148491s

xuwei@proteus100[twitter_nlp]$_
Summary

classification (Naïve Bayes)

Language Identification

Regular Expression

Tokenization

Part-of-Speech (POS) Tagging

Shallow Parsing (Chunking)

Named Entity Recognition (NER)

Stemming

Normalization

Sequential Tagging
Presentation 1