CSC401: Natural Language Processing

Tutorial: Assignment 1
Preprocessing/Tokenizing/Tagging

Jan 29, 2015
TA: Stefania Raimondo
t6raimon@cdf.Toronto.edu

(Slides adapted from Siavash Kazemian, previous prepared by Varada Kolhatkar and Ka-Chun Won)
Goal

Perform sentiment analysis on individual tweets:
Binary classification of tweets as having positive or negative sentiment

Input

“I love my BrandNameProduct!”
“Never shop at X-Store. Total garbage.”

Output

Positive
Negative
Methodology

Part 1-3: You + Weka

Tweet Corpus → Preprocessing → Normalized Tweets → Feature Extraction → Feature Vectors → Classification → Classes

Part 4: Watson

Tweet Corpus → Classification → Tweets → Classes
Methodology

Part 1-3: You + Weka

Tweet Corpus

Tweets

Preprocessing

Normalized Tweets

Feature Extraction

Feature Vectors

Classification

Classes

Part 4: Watson

Tweet Corpus

Tweets

Classification

Classes

TODAY
Tweet Corpus

- /u/cs401/A1/tweets
  - 1,600,000 training tweets - training.1600000.processed.noemoticon.csv
  - 359 testing tweets - testdata.manualSUBSET.2009.06.14.csv

- Format: 1 tweet/line in .csv

Date of Tweet
Tweet Id
Polarity: 0 (negative), 4 (positive)
Tweet text
User
Query (or NO_QUERY)
Tweet Corpus

• Your group’s specific of Training Data:
  • 11,000 tweets, chunks based on your GID
  • 1st 5500 (class 0): \([GID \times 5500 \ldots (GID+1) \times 5500 - 1]\)
  • 2nd 5500 (class 4): \(800,000 + [GID \times 5500 \ldots (GID+1) \times 5500 - 1]\)
Preprocessing: 3 Steps

1. “Pre-processing” aka cleaning tweets
2. Tokenizing
3. Tagging
Preprocessing: 3 Steps

Table 2: Conversion from raw tweets to tagged tweets

Raw tweet:


Output from twtt.py:

...  
<A=4> 
Meet/VB me/PRP today/NN at/IN the/DT FEC/NN in/IN DC/NN at/IN 4/NN ./  
Wear/VB a/DT carnation/NN so/RB I/PRP know/VB it/PRP ’s/POS you/PRP ./  
<A=0>  
...
Preprocessing: Detailed Steps

• Remove HTML tags/attributes/characters
• Remove URLs
• Twitter # and @ symbol removal
• Sentence boundary identification
• Tokenize
• POS Tag
• Delimit Tweets
Removing HTML/URLs

• Regex is your friend!

• For fixed patterns, you can use *string replace*
  - Ex. `mystring.replace("&amp;","&")`
  - Note: strings are immutable

• For variable patterns, you’ll need *regular expressions*
  - Ex. For html start tags (e.g., `<html>`, `<ahref="google.com">`) use `re.sub`
  - Note: re is greedy! (so ‘<.+>’ isn’t good enough)
Sentence Boundaries: Hard

• Sentences end with ‘.’, ‘?’ or ‘!’

• But not all periods are EOS (e.g. abbreviations)
  e.g., How much does the U.S. president get paid?

• But some abbreviations are EOS
  e.g., After the UK tour ends next week, he returns to the U.S.

• Possible solution: consider checking if the following letter is lowercase
  But what about: e.g., After U.S. Attorney General...

• List of common abbreviations:
  • /u/cs401/Wordlists/abbrev.english
Sentence Boundaries: Hard (con’t)

• Don’t break multiple times for multiple punctuation(e.g. !!!)

• But not all ellipsis are EOS
  e.g., I dunno Manny... do you want to go?

• Quotations: after the punctuation, but part of the sentence
  e.g., “You remind me,” she remarked, “of your mother.”

• There is no perfect sentence parser!

• See Manning and Schütze, Section 4.2.4 for some good ideas
Tokenization: Splitting sentences into tokens

• Simple words: Use `line.strip().split()`
  e.g., ‘an apple’ → [‘an’, ‘apple’]

• Punctuation should be its own token
  e.g., ‘she said,’ → [‘she’, ‘said’, ‘,’]

• But not always...
  e.g., ‘paid $10,000’ → [‘paid’, ‘$’, ‘10,000’]

• Including clitics and contractions
  e.g., “can’t” → [“ca”, “n’t”]
Tokenization (con’t)

• Possessives
e.g., “she’s” → [“she”, “’s”]

• Compounds (your choice)
e.g., time-consuming

• Don’t break up ellipsis...
POS Tagging

• Use the module we’ve provided: `import NLPlib`

• Only load the tagger **once**!
  
  `tagger = NLPlib.NLPlib()`

• Pass a list of tokens to the tag method:
  
  `tags = tagger.tag([‘the’, ‘boy’])`

  Returns `['DT', 'NN']`

• Do not tag empty strings
## Tag list (see handout)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Name</th>
<th>Example</th>
<th>POS</th>
<th>PRP</th>
<th>PRP$</th>
<th>RB</th>
<th>RBR</th>
<th>RBS</th>
<th>RP</th>
<th>SYM</th>
<th>TO</th>
<th>UH</th>
<th>VB</th>
<th>VBD</th>
<th>VBG</th>
<th>VBN</th>
<th>VBP</th>
<th>VBZ</th>
<th>WDT</th>
<th>WP</th>
<th>WP$</th>
<th>WRB</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordinating conjunction</td>
<td>and three the there [is]</td>
<td>Possessive ending</td>
<td>I, he, it</td>
<td>my, his, its</td>
<td>however, usually,</td>
<td>better</td>
<td>best</td>
<td>[give] up</td>
<td>to</td>
<td>to [go] to [him]</td>
<td>uh-huh</td>
<td>take</td>
<td>took</td>
<td>taking</td>
<td>taken</td>
<td>take</td>
<td>takes</td>
<td>which</td>
<td>who, what</td>
<td>whose</td>
<td>where, when</td>
</tr>
</tbody>
</table>
Tag list (see handout)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Pound sign</td>
<td>£</td>
</tr>
<tr>
<td>$</td>
<td>Dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>.</td>
<td>Sentence-final punctuation</td>
<td>!, ?, .</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>Colon, semi-colon, ellipsis</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>Left bracket character</td>
<td></td>
</tr>
<tr>
<td>)</td>
<td>Right bracket character</td>
<td></td>
</tr>
<tr>
<td>&quot;</td>
<td>Straight double quote</td>
<td></td>
</tr>
<tr>
<td>‘</td>
<td>Left open single quote</td>
<td></td>
</tr>
<tr>
<td>“</td>
<td>Left open double quote</td>
<td></td>
</tr>
<tr>
<td>‘</td>
<td>Right close single quote</td>
<td></td>
</tr>
<tr>
<td>”</td>
<td>Right close double quote</td>
<td></td>
</tr>
</tbody>
</table>
Delimit tweets

• Output file: (*.twt)...
• Space between tokens (" ".join(tokens))
• Each line is a sentence, *not* a tweet ("\n".join(sents))
• Each tweet is separated "<A=#>" on a separate line

• If a tweet is empty (e.g. only url), include the empty tweet!
  • Your feature extractor must handle this condition
Example .twt file

<A=0>
Hindsight/NN ./.
Yeah/UH ,/, that/IN was/VBD probably/RB a/DT poorly/RB worded/VBN tweet/NN ./.
<A=4>
Pick/VB up/IN the/DT jacket/NN .../:
Tips

• Sanity check often
• Peek at the tweets
• Use your best judgement
  • Check out how these tools handle specific cases:
    • https://code.google.com/p/splitta/
    • http://nlp.stanford.edu/software/tokenizer.shtml
• **Finish Part 1 ASAP!**
  • Get it working. Don’t worry about perfecting it. There’s no such thing as a perfect parser.