CSC401: Natural Language Processing

Tutorial: Assignment 1

Jan 20, 2017

TA: Aryan Arbabi

arbabi@cs.toronto.edu

(Slides adapted from Stefania Raimondo, Erin Grant, Siavash Kazemian, Varada Kolhatkar and Ka-Chun Won)

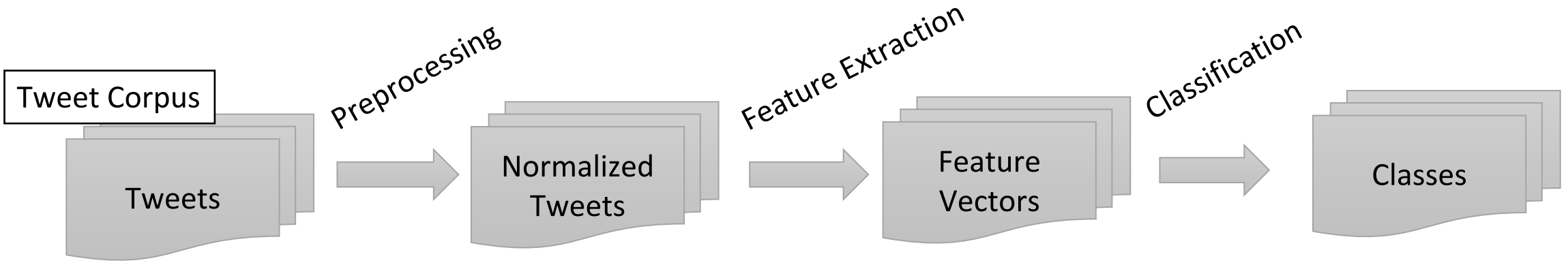
Goal

Perform sentiment analysis on individual tweets:

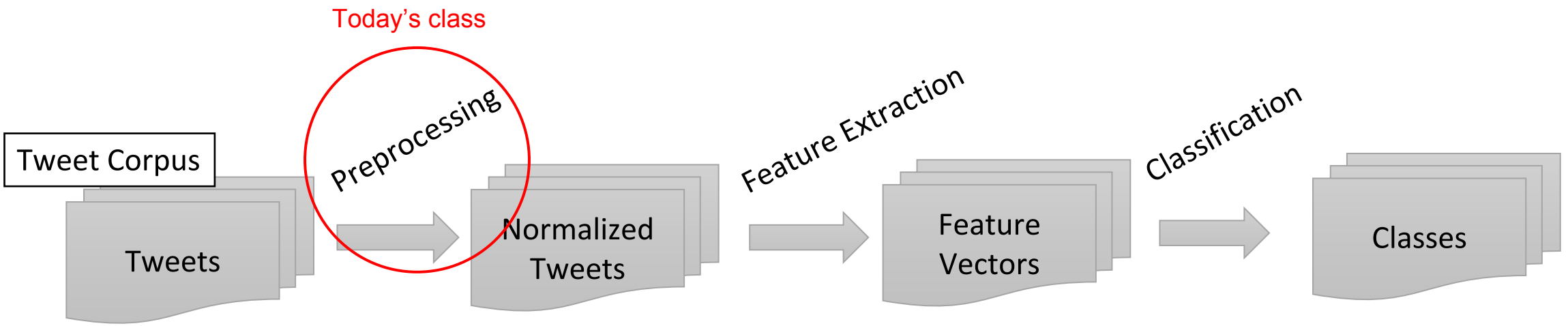
Binary classification of tweets as having positive or negative sentiment

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

Methodology



Methodology



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

`4, 2087, Sat May 16 23:58:44 UTC 2009, friends, bffever, Best @friendster ever! #omg`

Tweet Corpus

- Use a subset of 20,000 tweets from the training file

$$\begin{cases} \text{ID X } 10{,}000 + [0 \ldots 9999] \\ 800{,}000 + \text{ID X } 10{,}000 + [0 \ldots 9999] \end{cases}$$

- Where ID is your student ID module 80

Preprocessing: 3 steps

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

- There are 9 tasks
  - Implement one function for each (except 6)
  - Each function takes a string and outputs the modified string
  - Name the functions **twtt1** to **twtt9**

Preprocessing - Example

Table 2: Conversion from raw tweets to tagged tweets

Raw tweet:

```
Meet me today at the FEC in DC at 4. Wear a carnation so I know
it's you. <a href="Http://bit.ly/PACattack" target="_blank"
class="tweet-url web" rel="nofollow">Http://bit.ly/PACattack</a>.
```

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(“&”,&)
  - 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 it’s 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)

Tag	Name	Example
CC	Coordinating conjunction	*and*
CD	Cardinal number	*three*
DT	Determiner	*the*
EX	Existential *there*	*there [is]*
FW	Foreign word	*d'oeuvre*
IN	Preposition or subordinating conjunction	*in, of, like*
JJ	Adjective	*green, good*
JJR	Adjective, comparative	*greener, better*
JJS	Adjective, superlative	*greenest, best*
LS	List item marker	*(1)*
MD	Modal	*could, will*
NN	Noun, singular or mass	*table*
NNS	Noun, plural	*tables*
NNP	Proper noun, singular	*John*
NNPS	Proper noun, plural	*Vikings*
PDT	Predeterminer	*both [the boys]*
POS	Possessive ending	*'s, '*
PRP	Personal pronoun	*I, he, it*
PRP$	Possessive pronoun	*my, his, its*
RB	Adverb	*however, usually,*
RBR	Adverb, comparative	*better*
RBS	Adverb, superlative	*best*
RP	Particle	*[give] up*
SYM	Symbol (mathematical or scientific)	+
TO	*to*	*to [go] to [him]*
UH	Interjection	*uh-huh*
VB	Verb, base form	*take*
VBD	Verb, past tense	*took*
VBG	Verb, gerund or present participle	*taking*
VBN	Verb, past participle	*taken*
VBP	Verb, non-3rd-person singular present	*take*
VBZ	Verb, 3rd-person singular present	*takes*
WDT	*wh*-determiner	*which*
WP	*wh*-pronoun	*who, what*
WP$	Possessive *wh*-pronoun	*whose*
WRB	*wh*-adverb	*where, when*

Tag list (see handout)

Tag	Name	Example
#	Pound sign	£
$	Dollar sign	$
.	Sentence-final punctuation	!, ?, .
,	Comma	
:	Colon, semi-colon, ellipsis	
(	Left bracket character	
)	Right bracket character	
"	Straight double quote	
‘	Left open single quote	
“	Left open double quote	
’	Right close single quote	
”	Right close double quote	

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.