Web and Social Media Analytics

Social media (sentiment analysis II)

Dr Philip Worrall

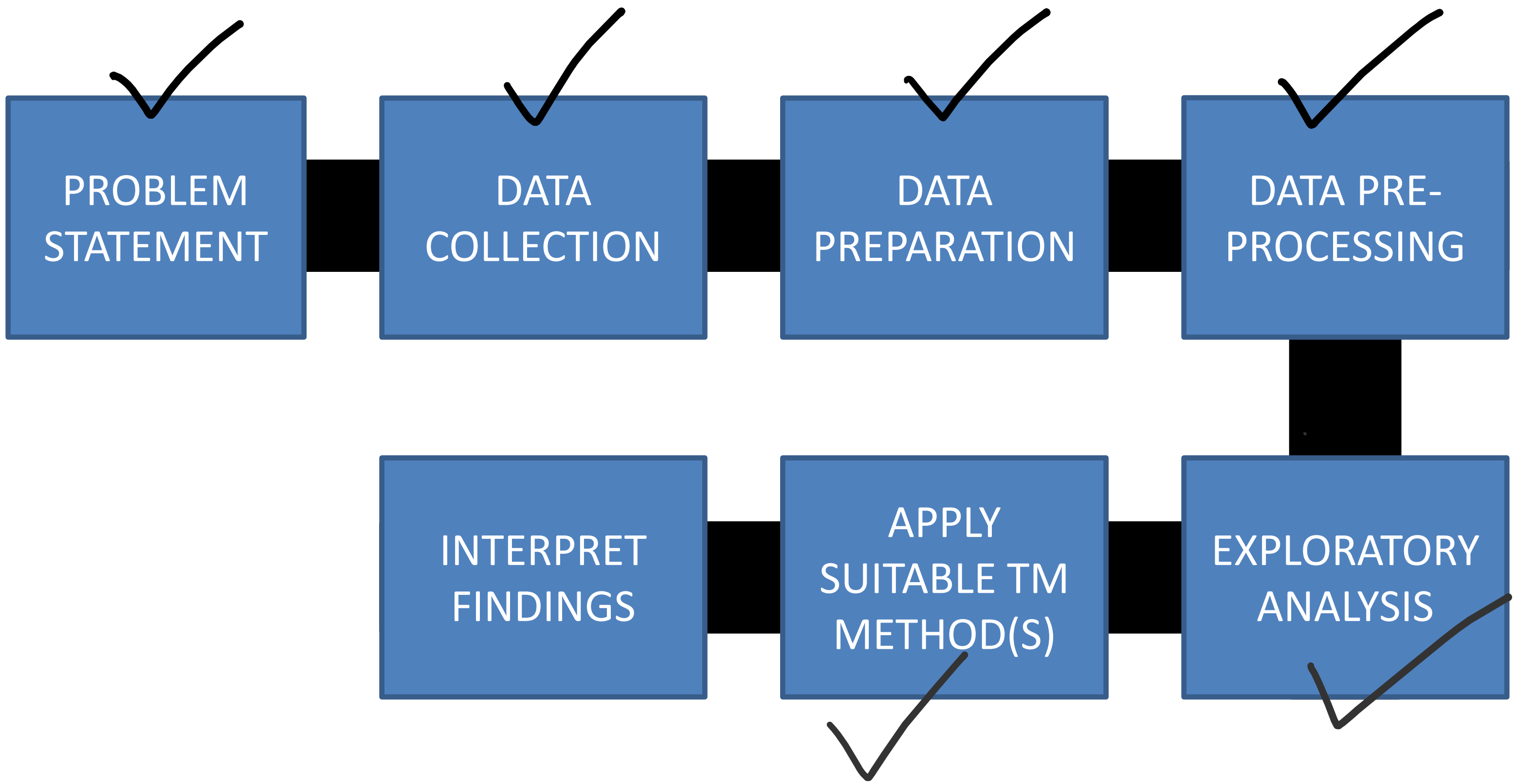
School of Computer Science and Engineering
115 New Cavendish Street
University of Westminster
London, W1W 6UW
worralph@westminster.ac.uk

LW12

Outline

- Recap of last week's material
- Sentiment models II
  - Sentiment scoring models (Lexicon-based approach)
  - Rule based approaches (Parts of Speech)
  - Statistical models (Machine Learning)

The text mining process…



extensions to sentiment modelling

Sentiment scoring models...

What are the posibilities?

LEXICONS → SCORING METHODOLOGY → SENTIMENT LABELS

from?
which?

What did we do in the tutorials?

How to validate results?

The SocialSent Lexicon database...

reddit

Community Specific

	word	sentiment	std.dev
0	ugly	-3.90	1.16
1	painful	-3.69	1.53
2	intent	-3.49	1.67
3	terrible	-3.38	1.55
4	drunk	-3.28	1.16
...	...	...	...
4919	perfectly	2.69	0.83
4920	romantic	2.70	0.76
4921	delicate	2.72	0.93
4922	beautiful	2.73	0.69
4923	wonderful	2.76	0.71

4924 rows × 3 columns

Social Sent 2016

domain Specific

SocialSent methodology…

- Different words have different meanings in different communities.
- Words come in and out of “fashion”.
- >=5% of words have switched their polarity between 1850-2000. “lean”
- It is time consuming and inefficient to manually label each lexicon (across multiple domains).
- Similarly, we are often working with unlabelled text data.

SocialSent methodology...

tell me

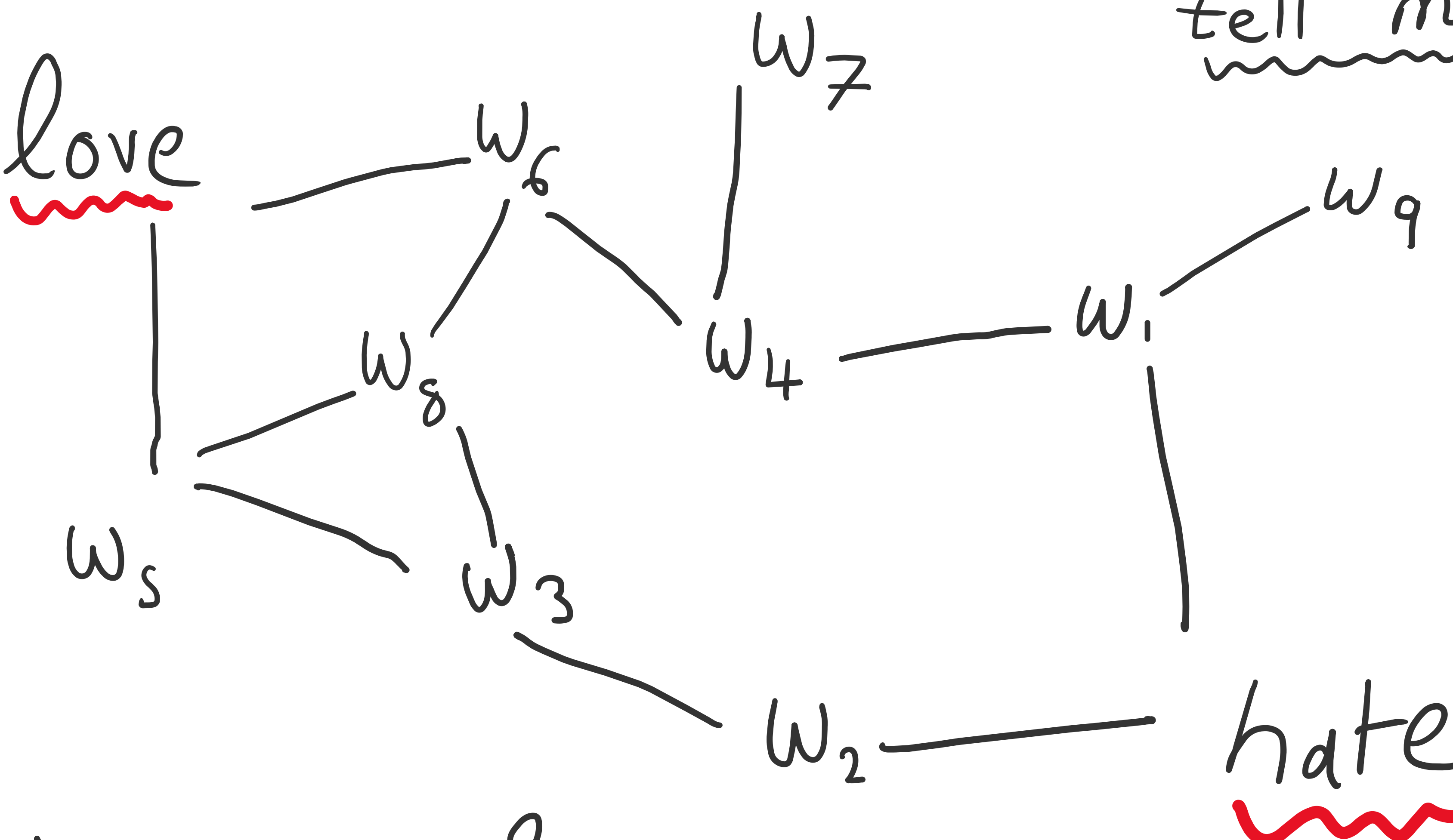


diagram type?

Alternative sources of lexicons…

- NRC Word-Emotion Association Lexicon
  - List of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive).
  - The annotations were manually done by crowdsourcing.

https://saifmohammad.com/WebDocs/Lexicons/NRC-Emotion-Lexicon.zip

National research council of Canada

Rule based approaches...

- How reliable are sentiment lexicon scores?
- Favour the simple over the complex
- Pointwise Mutual Information

"I forgot to lock my car"

Turney (2002), https://arxiv.org/ftp/cs/papers/0212/0212032.pdf

Parts of speech tagger...

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

from nltk.tokenize import word_tokenize

comment = "i forgot to lock my car"
tokens = word_tokenize(comment)

nltk.pos_tag(tokens)
```

LW10

```
[('i', 'NN'),
 ('forgot', 'VBD'),
 ('to', 'TO'),
 ('lock', 'VB'),
 ('my', 'PRP$'),
 ('car', 'NN')]
```

extract 2/3 word patterns

eg adj + noun

rule

verb (past)
+ to (infinitive)
+ verb

3 word rule…

- "forgot to lock"

$$pmi = \log \frac{Pr(\text{"forgot to lock"} \cap \text{"excellent"})}{Pr(\text{"forgot to lock"}) \cdot Pr(\text{"excellent"})}$$

Semantic orientation (sentiment score) $= pmi_{excellent} - pmi_{awful}$

PMI in Pandas...

Ex 1

```
df[df["text"].str.contains("forgot to lock")] / len(df.index)

df[df["text"].str.contains("excellent")] / len(df.index)
```

Ex2

```
df[(df["text"].str.contains("forgot to lock"))
      &(df["text"].str.contains("excellent"))] / len(df.index)
```

Statistical models…

STS-Gold

- Require (large) pre-labelled data instances
- Predict sentiment class based on word features

BoW

features

pay	hate	love	tough	Class
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0

context aware lexicons, custom features

Using the sample twitter data...

```python
from nltk.corpus import twitter_samples
nltk.download('twitter_samples')

positive_tweets = twitter_samples.strings('positive_tweets.json')
negative_tweets = twitter_samples.strings('negative_tweets.json')
```

```
positive_tweets
```

```
 '@oohdawg_ Hi liv :))',
 'Hello I need to know something can u fm me on Twitter?? — sure thing :) dm me x http://t.co/W6Dy130BV7',
 '#FollowFriday @MBandScott_ @Eric_FLE @pointsolutions3 for being top new followers in my community this week :)',
 "@rossbreadmore I've heard the Four Seasons is pretty dope. Penthouse, obvs #Gobigorgohome\nHave fun y'all :)",
 '@gculloty87 Yeah I suppose she was lol! Chat in a bit just off out x :))',
 'Hello :) Get Youth Job Opportunities follow &gt;&gt; @tolajobjobs @maphisa301',
 "💅💋 - :)))) haven't seen you in years",
 '@Bosslogic @amellywood @CW_Arrow @ARROWwriters Thank you! :-)',
```

5000 neg / pos

Gathering the data...

```
import pandas as pd
rows = []
for tweet in positive_tweets:
  rows.append({"text": tweet, "class": "pos"})
for tweet in negative_tweets:
  rows.append({"text": tweet, "class": "neg"})
df = pd.DataFrame(rows)
```

So far so good...

	text	class
0	#FollowFriday @France_Inte @PKuchly57 @Milipol...	pos
1	@Lamb2ja Hey James! How odd :/ Please call our...	pos
2	@DespiteOfficial we had a listen last night :)...	pos
3	@97sides CONGRATS :)	pos
4	yeaaaah yipppppy!!! my accnt verified rqst has...	pos
...	...	...
9995	I wanna change my avi but uSanele :(	neg
9996	MY PUPPY BROKE HER FOOT :(	neg
9997	where's all the jaebum baby pictures :((	neg
9998	But but Mr Ahmad Maslan cooks too :(https://t...	neg
9999	@eawoman As a Hull supporter I am expecting a ...	neg

10000 rows × 2 columns

Now for pre-processing...

```python
from nltk.corpus import stopwords
nltk.download('stopwords')

stopwords = stopwords.words('english')

def preprocess(row):
  text = row["text"].lower()
  keep = []
  for word in text.split():
    if word in stopwords:
      continue
    if word.startswith("@"):
      continue
    if word.startswith("http"):
      continue
    if word.startswith("#"):
      continue
    if word == "follow":
      continue
    if len(word) <= 1:
      continue
    keep.append(word)
  return ' '.join(keep)

df["cleaned_text"] = df.apply(preprocess, axis=1)
```

Identifying common terms...

```
from collections import Counter

word_counter = Counter()

for row in df.to_dict("records"):
   word_counter.update(row["cleaned_text"].split())
word_counter.most_common(10)
```

why?

```
[(':(', 3796),
 (':)', 3272),
 (':-)', 632),
 (':d', 629),
 ("i'm", 520),
 (':-(', 431),
 ('like', 410),
 ('love', 389),
 ('thanks', 372),
 ('get', 346)]
```

Extracting most frequent words...

tuples of words
and frequencies

```
features = [word for word, freq in word_counter.most_common(10)]

features

[':(', ':)', ':-)', ':d', "i'm", ':-(', 'like', 'love', 'thanks', 'get']
```

I will choose
50.

Extracting features...

```
def to_features(row):
  keep = []
  text = row["cleaned_text"]
  for word in text.split():
    if word in features:
      keep.append(word)
  return ' '.join(keep)

df["features"] = df.apply(to_features, axis=1)
```

With only extracted features...

class	cleaned_text	features
pos	top engaged members community week :)	:)
pos	hey james! odd :/ please call contact centre 02392441234 able assist :) many thanks!	please :)
pos	listen last night :) bleed amazing track. scotland?!	:)
pos	congrats :)	:)
pos	yeaaaah yipppy!!! accnt verified rqst succeed got blue tick mark fb profile :) 15 days	got :)
pos	one irresistible :)	one :)
pos	like keep lovely customers waiting long! hope enjoy! happy friday! lwwf :)	like hope happy :)
pos	second thought, there's enough time dd :) new shorts entering system. sheep must buying.	time :) new

Creating the train/test split...

```
shuffled = df.sample(frac=1)
```

```
train = shuffled[0:4000]
test = shuffled[4000:]
```

80 : 20 split

Expected input data format…

```
train = [
    ('I love this sandwich.', 'pos'),
    ('this is an amazing place!', 'pos'),
    ('I feel very good about these beers.', 'pos'),
    ('this is my best work.', 'pos'),
    ("what an awesome view", 'pos'),
    ("the exam was not very difficult", 'pos'),
    ('I do not like this restaurant', 'neg'),
    ('I am tired of this stuff.', 'neg'),
    ("I can't deal with this", 'neg'),
    ('he is my sworn enemy!', 'neg'),
    ('my boss is horrible.', 'neg')
]
test = [
    ('the beer was good.', 'pos'),
    ('I do not enjoy my job', 'neg'),
    ("I ain't feeling dandy today.", 'neg'),
    ("I feel amazing!", 'pos'),
    ('Gary is a friend of mine.', 'pos'),
    ("I can't believe I'm doing this.", 'neg')
]
```

Making a list of tuples…

```python
train_cls = []

for row in train.to_dict("records"):
  train_cls.append((row["features"], row["class"]))

train_cls
```

```
[(':(', 'neg'),
 (':)', 'pos'),
 (':(', 'neg'),
 (':d', 'pos'),
 (':(', 'neg'),
 (':)', 'pos'),
 ('back :)', 'pos'),
 ('get :( want', 'neg'),
 ('really really really really really really like :) :)', 'pos'),
 (':)', 'pos'),
 ("i'm :)", 'pos'),
 ('happy :-)', 'pos'),
```

Training the classifier…

```
from textblob.classifiers import NaiveBayesClassifier

classifier = NaiveBayesClassifier(train_cls)
```

the prepared dataset

```
classifier.show_informative_features(15)
```

```
Most Informative Features
           contains(miss) = True              neg : pos    =     26.0 : 1.0
             contains(hi) = True              pos : neg    =     13.2 : 1.0
          contains(great) = True              pos : neg    =     12.5 : 1.0
         contains(thanks) = True              pos : neg    =     12.0 : 1.0
          contains(happy) = True              pos : neg    =      9.3 : 1.0
          contains(sorry) = True              neg : pos    =      8.3 : 1.0
          contains(thank) = True              pos : neg    =      7.2 : 1.0
             contains(ca) = True              neg : pos    =      4.4 : 1.0
            contains(n't) = True              neg : pos    =      4.4 : 1.0
             contains(na) = True              neg : pos    =      4.1 : 1.0
            contains(wan) = True              neg : pos    =      4.1 : 1.0
           contains(feel) = True              neg : pos    =      4.1 : 1.0
              contains(3) = True              pos : neg    =      3.8 : 1.0
             contains(lt) = True              pos : neg    =      3.8 : 1.0
             contains(us) = True              pos : neg    =      3.3 : 1.0
```

Performance...

```
round(classifier.accuracy(test_cls), 3)
```

```
0.678
```

~35% improvement over random choice

* consider DecisionTree and MaxEntropy

In Summary

- Sentiment models may be constructed in a variety of ways.
- The most common methodologies involve the use of predefined lexicons with an existing sentiment score, rule based models and statistical approaches.
- The sentiment of individual lexicons is subject to debate and many point out that sentiment for a given word is likely to vary across time and in different domain contexts.
- Rule based approaches can be easier to comprehend and explain given their often intuitive explanation.
- Statistical approaches, including those based on machine learning techniques, involve the use of mathematical models to discriminate between different sentiment classes based on the absence/presence of selected terms.
- A drawback of using statistical approaches is their requirement for "large" amounts of pre-labelled data.

End