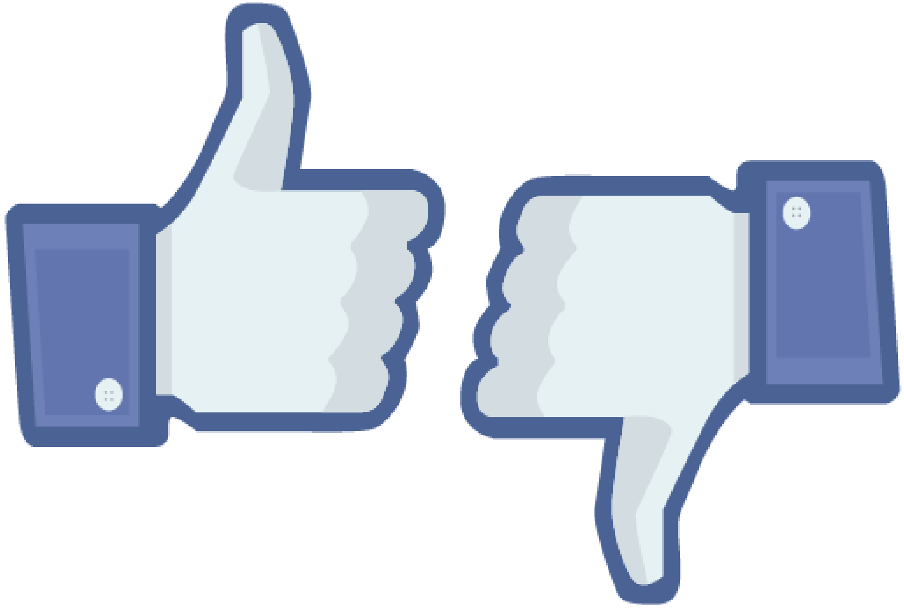
Algorithms for Natural Language Processing
Sentiment Analysis and Empirical Methods

(some slides adapted from Alex Lascarides)

25 September 2017

Sentiment Analysis



Goal: Predict the **opinion** expressed in a piece of text. E.g., $+$ or $-$. (Or a rating on a scale.)

Sentiment Analysis

Filled with horrific dialogue, laughable characters, a laughable plot, ad really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound effects, and most of the

KJ Proulx (/user/id/896976177/)
★ Super Reviewer

Extraordinarily faithful to the tone and style of the originals, The Force Awakens brings back the Old Trilogy's heart, humor, mystery, and fun. Since it is only the first piece in a new three-part journey it can't help but feel incomplete. But everything that's already there, from the stunning visuals, to the thrilling action sequences, to the charismatic new characters,

Matthew Samuel Mirliani (/user /id/896467979/)
★ Super Reviewer

RottenTomatoes.com

Sentiment Analysis

★★
Filled with horrific dialogue, laughable characters, a laughable plot, ad really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound effects, and most of the

KJ Proulx (/user/id/896976177/)
★ Super Reviewer

★★★★★
Extraordinarily faithful to the tone and style of the originals, The Force Awakens brings back the Old Trilogy's heart, humor, mystery, and fun. Since it is only the first piece in a new three-part journey it can't help but feel incomplete. But everything that's already there, from the stunning visuals, to the thrilling action sequences, to the charismatic new characters,

Matthew Samuel Mirliani (/user /id/896467979/)
★ Super Reviewer

RottenTomatoes.com

Sentiment Analysis

★★

Filled with horrific dialogue, laughable characters, a laughable plot, ad really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound effects, and most of the

KJ Proulx (/user/id/896976177/)
★ Super Reviewer

★★★★★

Extraordinarily faithful to the tone and style of the originals, The Force Awakens brings back the Old Trilogy's heart, humor, mystery, and fun. Since it is only the first piece in a new three-part journey it can't help but feel incomplete. But everything that's already there, from the stunning visuals, to the thrilling action sequences, to the charismatic new characters,

Matthew Samuel Mirliani (/user /id/896467979/)
★ Super Reviewer

RottenTomatoes.com + intuitions about positive/negative cue words

So, you want to build a sentiment analyzer

Questions to ask yourself:

1. What is the input for each prediction? (sentence? full review text? text+metadata?)

2. What are the possible outputs? (+ or −)

3. How will it decide?

4. How will you measure its effectiveness?

The last one, at least, requires data!

BEFORE you build a system, choose a dataset for evaluation!

Why is data-driven evaluation important?

- Good science requires controlled experimentation.
- Good engineering requires benchmarks.
- Your intuitions about typical inputs are probably wrong.

Sometimes you want multiple evaluation datasets: e.g., one for **development** as you hack on your system, and one reserved for final **testing**.

Where can you get a corpus?

- Many corpora are prepared specifically for linguistic/NLP research with text from content providers (e.g., newspapers). In fact, there is an entire subfield devoted to developing new **language resources**.
  - You have already seen SemCor for lexical semantics.

- You may instead want to collect a new one, e.g., by scraping websites. (There are tools to help you do this.)

Annotations

To evaluate and compare sentiment analyzers, we need reviews with **gold labels** ($+$ or $-$) attached. These can be

- derived automatically from the original data artifact (**metadata** such as star ratings), or
- added by a human annotator who reads the text
  - Issue to consider/measure: How consistent are human annotators? If they often have trouble deciding or agreeing, how can this be addressed?

More on these issues later in the course!

An evaluation measure

Once we have a dataset with gold (correct) labels, we can give the text of each review as input to our system and measure how often its output matches the gold label.

Simplest measure:

$$\textbf{accuracy} = \frac{\#\ \text{correct}}{\#\ \text{total}}$$

More measures later in the course!

Catching our breath

We now have:

✓ a definition of the sentiment analysis task (inputs and outputs)

✓ a way to measure a sentiment analyzer (accuracy on gold data)

So we need:

- an algorithm for predicting sentiment

A simple sentiment classification algorithm

Use a **sentiment lexicon** to count positive and negative words:

Positive:

absolutely	beaming	calm
adorable	beautiful	celebrated
accepted	believe	certain
acclaimed	beneficial	champ
accomplish	bliss	champion
achieve	bountiful	charming
action	bounty	cheery
active	brave	choice
admire	bravo	classic
adventure	brilliant	classical
affirm	bubbly	clean
. . .		. . .

Negative:

abysmal	bad	callous
adverse	banal	can't
alarming	barbed	clumsy
angry	belligerent	coarse
annoy	bemoan	cold
anxious	beneath	collapse
apathy	boring	confused
appalling	broken	contradictory
atrocious		contrary
awful		corrosive
		corrupt
		. . .

From `http://www.enchantedlearning.com/wordlist/`

Simplest rule: Count positive and negative words in the text. Predict whichever is greater.

Some possible problems with simple counting

1. Hard to know whether words that *seem* positive or negative tend to actually be used that way.
   - sense ambiguity
   - sarcasm/irony
   - text could mention expectations or opposing viewpoints, in contrast to author's actual opinon

2. Opinion words may be describing (e.g.) a character's attitude rather than an evaluation of the film.

3. Some words act as semantic modifiers of other opinion-bearing words/phrases, so interpreting the full meaning requires sophistication:

I **can't** stand this movie

vs.

I **can't** believe how great this movie is

What if we have more data?

Perhaps corpora can help address the first objection:

1. Hard to know whether words that *seem* positive or negative tend to actually be used that way.

A data-driven method: Use **frequency counts** to ascertain which words tend to be positive or negative.

NLTK

The Natural Language Toolkit (http://nltk.org) is a Python library for NLP. NLTK

- is open-source, community-built software
- was designed for teaching NLP: simple access to datasets, reference implementations of important algorithms
- contains wrappers for using (some) state-of-the-art NLP tools in Python

It will help if you familiarize yourself with Python **strings** and methods/libraries for manipulating them.

(If you are familiar with Python 2.7, know that strings and Unicode are handled differently in Python 3.)

Using an NLTK corpus

```
>>> from nltk.corpus import movie_reviews
>>> movie_reviews.words()
[u'plot', u':', u'two', u'teen', u'couples', u'go', ...]
>>> movie_reviews.sents()
[[u'plot', u':', u'two', u'teen', u'couples', u'go',
 ↪ u'to', u'a', u'church', u'party', u',', u'drink',
 ↪ u'and', u'then', u'drive', u'.'], [u'they',
 ↪ u'get', u'into', u'an', u'accident', u'.'], ...]
>>> print('\n'.join(' '.join(sent) for sent in
 ↪ movie_reviews.sents()[:5]))
plot : two teen couples go to a church party , drink
 ↪ and then drive .
they get into an accident .
one of the guys dies , but his girlfriend continues to
 ↪ see him in her life , and has nightmares .
what ' s the deal ?
watch the movie and " sorta " find out .
```

Using an NLTK corpus: word frequencies

```
>>> from nltk import FreqDist
>>> f = FreqDist(movie_reviews.words())
>>> f.most_common()[:20]
[(u',', 77717), (u'the', 76529), (u'.', 65876), (u'a',
 ↪ 38106), (u'and', 35576), (u'of', 34123), (u'to',
 ↪ 31937), (u"'", 30585), (u'is', 25195), (u'in',
 ↪ 21822), (u's', 18513), (u'"', 17612), (u'it',
 ↪ 16107), (u'that', 15924), (u'-', 15595), (u')',
 ↪ 11781), (u'(', 11664), (u'as', 11378), (u'with',
 ↪ 10792), (u'for', 9961)]
```

Using an NLTK corpus: word frequencies

```
>>> f = FreqDist(w for w in movie_reviews.words() if
  ↪ any(c.isalpha() for c in w))
>>> f.most_common()[:20]
[(u'the', 76529), (u'a', 38106), (u'and', 35576),
  ↪ (u'of', 34123), (u'to', 31937), (u'is', 25195),
  ↪ (u'in', 21822), (u's', 18513), (u'it', 16107),
  ↪ (u'that', 15924), (u'as', 11378), (u'with',
  ↪ 10792), (u'for', 9961), (u'his', 9587), (u'this',
  ↪ 9578), (u'film', 9517), (u'i', 8889), (u'he',
  ↪ 8864), (u'but', 8634), (u'on', 7385)]
```

Using an NLTK corpus: categories

```
>>> movie_reviews.categories()
[u'neg', u'pos']
>>> fpos =
  ↪ FreqDist(movie_reviews.words(categories='pos'))
>>> fneg =
  ↪ FreqDist(movie_reviews.words(categories='neg'))
>>> fMoreNeg = fneg - fpos
>>> fMoreNeg.most_common()[:20]
[(u'movie', 721), (u't', 700), (u'i', 685), (u'bad',
  ↪ 673), (u'?', 631), (u'"', 628), (u'have', 421),
  ↪ (u'!', 399), (u'no', 350), (u'plot', 321),
  ↪ (u'there', 318), (u'if', 301), (u'*', 286),
  ↪ (u'this', 282), (u'so', 267), (u'why', 250),
  ↪ (u'just', 221), (u'only', 219), (u'worst', 210),
  ↪ (u'even', 207)]
```

What if we have more data?

Perhaps corpora can help address the first objection:

1. Hard to know whether words that *seem* positive or negative tend to actually be used that way.

A data-driven method: Use frequency counts from a **training corpus** to ascertain which words tend to be positive or negative.

- Why separate the training and test data (held-out test set)? Because otherwise, it's just data analysis; no way to estimate how well the system will do on new data in the future.

Choice of training and evaluation data

We know that the way people use language varies considerably depending on **context**. Factors include:

- *Mode of communication:* speech (in person, telephone), writing (print, SMS, web)
- *Topic:* chitchat, politics, sports, physics, . . .
- *Genre:* news story, novel, Wikipedia article, persuasive essay, political address, tweet, . . .
- *Audience:* formality, politeness, complexity (think: child-directed speech), . . .

In NLP, **domain** is a cover term for all these factors.

Choice of training evaluation data

- Statistical approaches typically assume that the training data and the test data are sampled from the same distribution.
  - I.e., if you saw an example data point, it would be hard to guess whether it was from the training or test data.
- Things can go awry if the test data is appreciably different: e.g.,
  - different tokenization conventions
  - new vocabulary
  - longer sentences
  - more colloquial/less edited style
  - different distribution of labels
- **Domain adaptation** techniques attempt to correct for this assumption when something about the source/characteristics of the test data is known to be different.

Why do we need text corpora?

Two main reasons:

1. To evaluate our systems on
   - Good science requires controlled experimentation.
   - Good engineering requires benchmarks.

2. To help our systems work well (data-driven methods/machine learning)
   - When a system's behavior is determined solely by manual rules or databases, it is said to be **rule-based**, **symbolic**, or **knowledge-driven** (early days of computational linguistics)
   - **Learning**: collecting statistics or patterns automatically from corpora to govern the system's behavior (dominant in most areas of contemporary NLP)
     - **supervised learning**: the data provides example input/output pairs (main focus in this course)
     - core behavior: **training**; refining behavior: **tuning**