Sentiment tutorial home **Christopher Potts**, **Stanford Linguistics**

Sentiment Symposium Tutorial

Sentiment Analysis Symposium, San Francisco, November 8-9, 2011

Instructor: **Christopher Potts** (Stanford Linguistics)

Overview

1. **Overview: goals, plan, and applications**
2. **Sentiment in language and cognition**
3. Text preparation:
   a. **Tokenization**
   b. **Stemming**
   c. **Advanced linguistic structure**
4. **Sentiment lexicons**
5. **Classifier models for sentiment**
6. **Vector space models**
7. **Context-dependency and social relationships**
8. **Sentiment summarization**
9. **Bibliography**

Tutorial demos

1. **Tokenizing**
2. **Simple WordNet propagation**
3. **Sentiment lexicons word viewer**
4. **Text scoring with sentiment lexicons**
5. **Trained and probabilistic classifier models**
6. **Word-vector similarity in diverse corpora**

Publicly-available tutorial data and implementations

- **Basic, extensible Python sentiment tokenizer with random-tweet tokenizing function**

- Word distributions:
  - **Multi-corpus word distributions** (zip archive)
  - **Multi-linguial, multi-corpus word distributions**
  - **POS-tagged word distributions from IMDB** (zip archive)
  - **Multi-corpus statistics on WordNet adjectives and adverbs**
  - **Heavily documented R code for creating lexicons and scales from data like the above**
- **Large Movie Review dataset**
- **Simple WordNet sense propagation in Python/NLTK**
- **WordNet score propagation in Python/NLTK**; **example lexicon**; **documentation**
- **Cosine score propagation algorithm in Python**

Home |

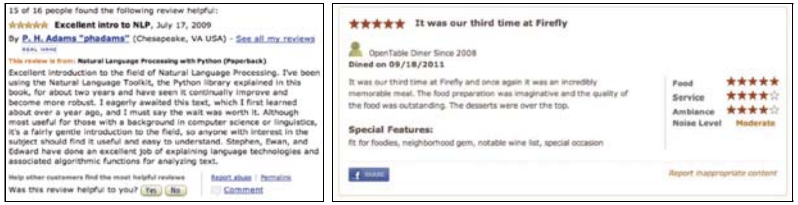
Sentiment tutorial home **Christopher Potts**, **Stanford Linguistics**

Sentiment Symposium Tutorial: Overview

1. **Outline**
2. **General goals**
3. **Applications**
    - 3.1 **Understanding customer feedback**
    - 3.2 **Brand analysis, reputation management, and social mentions**
    - 3.3 **The emotions of a nation**
    - 3.4 **Understanding public opinion**
    - 3.5 **Monitoring real-world events**
    - 3.6 **Media studies**
    - 3.7 **Realistic text-to-speech**
    - 3.8 **Financial prediction**
4. **Other publicly available sentiment tutorials**

The amount of user-generated content on the Internet has risen exponentially over the last decade, and such content is now always at our fingertips. As a result, nearly all our decision-making is social; before buying products (attending events, trying services, voting for candidates, visiting specialists), we see what our peers are saying about them. The fate of a new offering is often sealed by those evaluations.

Figure 1 A wealth of user-generated content to guide decision making.



15 of 16 people found the following review helpful:

★★★★★ **Excellent intro to NLP**, July 17, 2009

By P. H. Adams "phadams" (Chesapeake, VA USA) - See all my reviews

REAL NAME

This review is from: **Natural Language Processing with Python (Paperback)**

Excellent introduction to the field of Natural Language Processing. I've been using the Natural Language Toolkit, the Python library explained in this book, for about two years and have seen it continually improve and become more robust. I eagerly awaited this text, which I first learned about over a year ago, and I must say the wait was worth it. Although most useful for those with a background in computer science or linguistics, it's a fairly gentle introduction to the field, so anyone with interest in the subject should find it useful and easy to understand. Stephen, Ewan, and Edward have done an excellent job of explaining language technologies and associated algorithmic functions for analyzing text.

Help other customers find the most helpful reviews

Was this review helpful to you? Yes No

Report abuse | Permalink

Comment

★★★★★ **It was our third time at Firefly**

OpenTable Diner Since 2008

Dined on 09/18/2011

It was our third time at Firefly and once again it was an incredibly memorable meal. The food preparation was imaginative and the quality of the food was outstanding. The desserts were over the top.

Special Features:

fit for foodies, neighborhood gem, notable wine list, special occasion

Food	★★★★★
Service	★★★★☆
Ambiance	★★★★☆
Noise Level	Moderate

Sentiment analysis, the computational study of how opinions, attitudes, emotions, and perspectives are expressed in language, provides a rich set of tools and techniques for extracting this evaluative, subjective information from large datasets and summarizing it. It can thus be vital to service providers, allowing them to quickly assess how new products and features are being received. Recent breakthroughs mean that this analysis can go beyond a general measure of positive vs. negative, isolating a fuller spectrum of emotions and evaluations and controlling for different topics and community norms.

1 Outline

This tutorial covers all aspects of building effective sentiment analysis systems for textual data, with and without sentiment-relevant metadata like star ratings. We proceed from pre-processing techniques to advanced uses cases, assessing common approaches and identifying best practices:

1. Overview: goals, plan, and applications
2. **Sentiment in language and cognition**
3. Text preparation:
    a. **Tokenization**
    b. **Stemming**
    c. **Advanced linguistic structure**
4. **Sentiment lexicons**
5. **Classifier models for sentiment**
6. **Vector space models**
7. **Context-dependency and social relationships**
8. **Sentiment summarization**
9. **Bibliography**

At each stage, I offer concrete advice supported by extensive experimental evidence. There are also **interactive demos** throughout, and **a collection of supporting data sets and implementations**.

2 General goals

1. To help you make informed decisions when setting up your own sentiment analysis systems.
2. To introduce you to techniques and technologies that will help you build such systems.
3. To help you evaluate other sentiment systems using both technical and empirical methods.
4. To argue that there is no break-away model for sentiment. We can measure the quality of a system by its data and linguistic analysis.

3 Applications

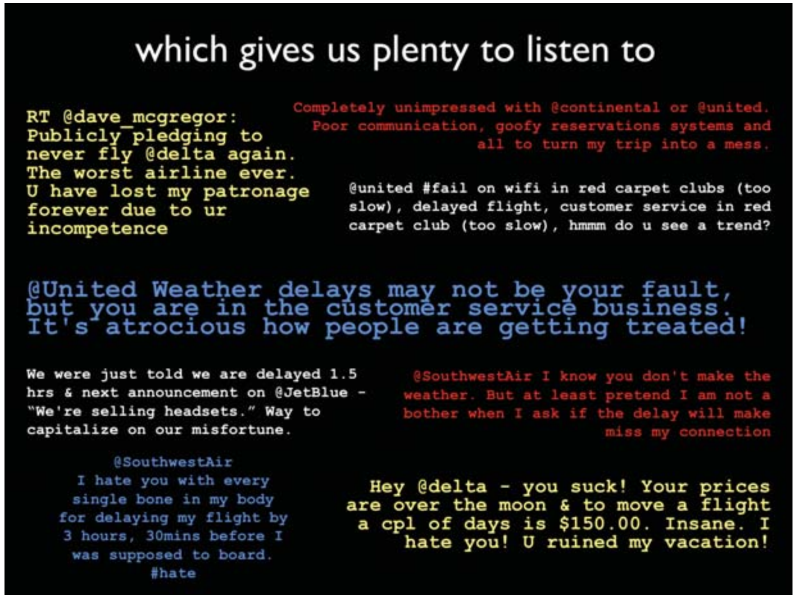
You're probably here because you have an application for sentiment analysis already in mind. Nonetheless, it's useful briefly survey some standard applications of it in business, finance, and the media.

3.1 Understanding customer feedback

Customer feedback is now directed not only at companies directly, but also broadcast on the Net via weblogs, Twitter, Facebook, and comments at retailers' websites.

Figure 2 is a sampling of Twitter feedback on airlines. Unusually for the Web (but perhaps not for airlines), the feedback is mostly negative here.

Figure 2 From Jeffrey Breen's **R by example: mining Twitter for attitudes towards airlines**.



which gives us plenty to listen to

RT @dave_mcgregor: Publicly pledging to never fly @delta again. The worst airline ever. U have lost my patronage forever due to ur incompetence

Completely unimpressed with @continental or @united. Poor communication, goofy reservations systems and all to turn my trip into a mess.

@united #fail on wifi in red carpet clubs (too slow), delayed flight, customer service in red carpet club (too slow), hmmm do u see a trend?

@United Weather delays may not be your fault, but you are in the customer service business. It's atrocious how people are getting treated!

We were just told we are delayed 1.5 hrs & next announcement on @JetBlue - "We're selling headsets." Way to capitalize on our misfortune.

@SouthwestAir I know you don't make the weather. But at least pretend I am not a bother when I ask if the delay will make miss my connection

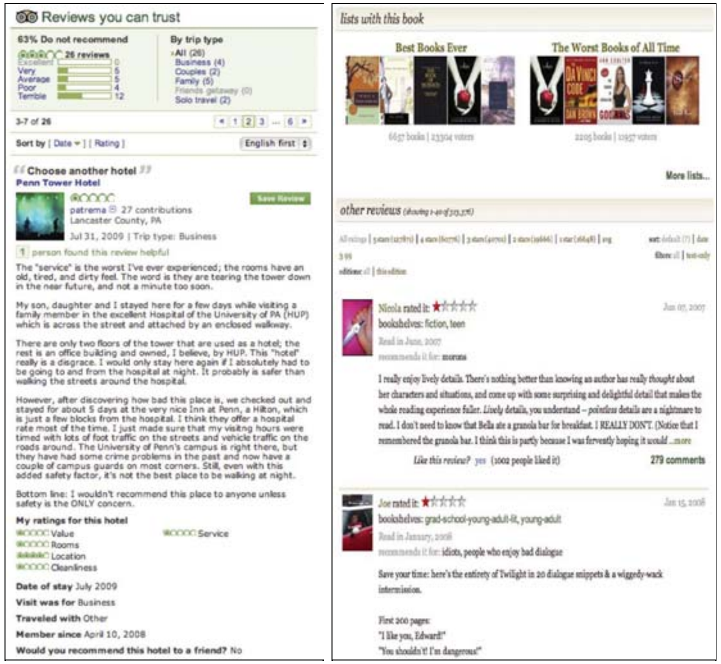
@SouthwestAir I hate you with every single bone in my body for delaying my flight by 3 hours, 30mins before I was supposed to board. #hate

Hey @delta - you suck! Your prices are over the moon & to move a flight a cpl of days is $150.00. Insane. I hate you! U ruined my vacation!

Figure 3 suggests how rich the customer feedback can be: targeted evaluations of specific aspects of the product, information about the author/reviewer, and feedback from readers about

the review.

Figure 3 User-generated content addressing multiple aspects of products in a social setting.



Reviews you can trust

63% Do not recommend

26 reviews

Excellent	0
Very	5
Average	5
Poor	4
Terrible	12

By trip type
All (26)
Business (4)
Couples (2)
Family (5)
Friends getaway (0)
Solo travel (2)

3-7 of 26 ◂ 1 2 3 ... 6 ▸

Sort by [Date ▾] [Rating] English first

"Choose another hotel"
Penn Tower Hotel

patrema 27 contributions
Lancaster County, PA
Jul 31, 2009 | Trip type: Business

Save Review

1 person found this review helpful

The "service" is the worst I've ever experienced; the rooms have an old, tired, and dirty feel. The word is they are tearing the tower down in the near future, and not a minute too soon.

My son, daughter and I stayed here for a few days while visiting a family member in the excellent Hospital of the University of PA (HUP) which is across the street and attached by an enclosed walkway.

There are only two floors of the tower that are used as a hotel; the rest is an office building and owned, I believe, by HUP. This "hotel" really is a disgrace. I would only stay here again if I absolutely had to be going to and from the hospital at night. It probably is safer than walking the streets around the hospital.

However, after discovering how bad this place is, we checked out and stayed for about 5 days at the very nice Inn at Penn, a Hilton, which is just a few blocks from the hospital. I think they offer a hospital rate most of the time. I just made sure that my visiting hours were timed with lots of foot traffic on the streets and vehicle traffic on the roads around. The University of Penn's campus is right there, but they have had some crime problems in the past and now have a couple of campus guards on most corners. Still, even with this added safety factor, it's not the best place to be walking at night.

Bottom line: I wouldn't recommend this place to anyone unless safety is the ONLY concern.

My ratings for this hotel
Value
Rooms
Location
Cleanliness
Service

Date of stay July 2009
Visit was for Business
Traveled with Other
Member since April 10, 2008
Would you recommend this hotel to a friend? No

lists with this book

Best Books Ever
6657 books | 23304 voters

The Worst Books of All Time
2205 books | 10957 voters

More lists...

other reviews (showing 1-40 of [illegible])

All ratings | 5 stars | 4 stars | 3 stars | 2 stars | 1 star | avg 3.99 sort: default | date
filter: all | text-only
editions: all | this edition

Nicola rated it: ★☆☆☆☆ Jun 07, 2007
bookshelves: fiction, teen
Read in June, 2007
recommends it for: morons

I really enjoy lively details. There's nothing better than knowing an author has really *thought* about her characters and situations, and come up with some surprising and delightful detail that makes the whole reading experience fuller. *Lively* details, you understand – *pointless* details are a nightmare to read. I don't need to know that Bella ate a granola bar for breakfast. I REALLY DON'T. (Notice that I remembered the granola bar. I think this is partly because I was fervently hoping it would ...more

Like this review? yes (1002 people liked it) 279 comments

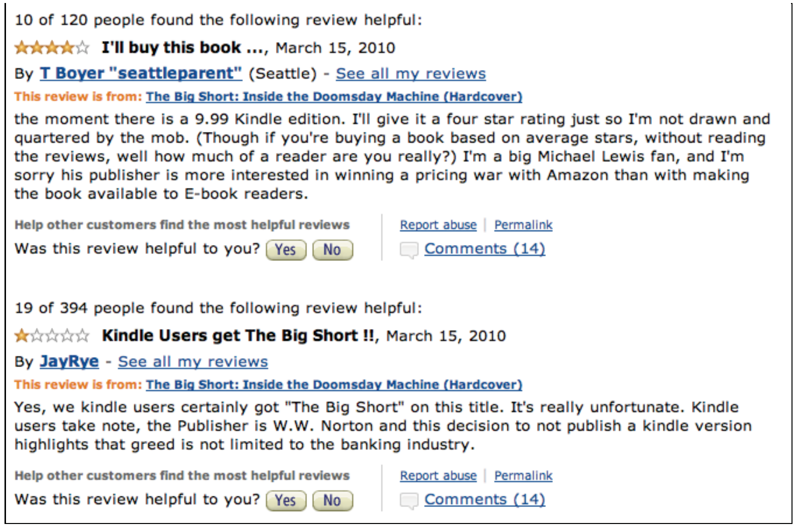
Joe rated it: ★☆☆☆☆ Jan 15, 2008
bookshelves: grad-school-young-adult-lit, young-adult
Read in January, 2008
recommends it for: idiots, people who enjoy bad dialogue

Save your time: here's the entirety of Twilight in 20 dialogue snippets & a wiggedy-wack intermission.

First 200 pages:
"I like you, Edward!"
"You shouldn't! I'm dangerous!"

Figure 4 suggests how challenging it is to process and accurately synthesize all of this data. (See also the wry irony of the Goodreads reviews in **figure 3**.)

Figure 4 Reviews of Michael Lewis's *The Big Short*. These reviews are not critical of the book, but rather of a decision by the publisher about when to release an electronic edition. Accurately identifying the target of the criticisms is crucial.



10 of 120 people found the following review helpful:

★★★★☆ **I'll buy this book ...**, March 15, 2010

By **T Boyer "seattleparent"** (Seattle) - See all my reviews

This review is from: The Big Short: Inside the Doomsday Machine (Hardcover)

the moment there is a 9.99 Kindle edition. I'll give it a four star rating just so I'm not drawn and quartered by the mob. (Though if you're buying a book based on average stars, without reading the reviews, well how much of a reader are you really?) I'm a big Michael Lewis fan, and I'm sorry his publisher is more interested in winning a pricing war with Amazon than with making the book available to E-book readers.

Help other customers find the most helpful reviews
Was this review helpful to you? Yes No
Report abuse | Permalink
Comments (14)

19 of 394 people found the following review helpful:

★☆☆☆☆ **Kindle Users get The Big Short !!**, March 15, 2010

By **JayRye** - See all my reviews

This review is from: The Big Short: Inside the Doomsday Machine (Hardcover)

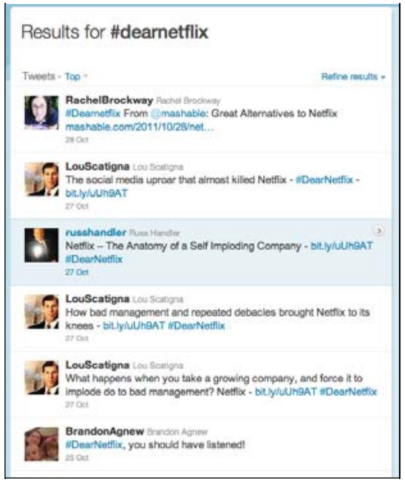
Yes, we kindle users certainly got "The Big Short" on this title. It's really unfortunate. Kindle users take note, the Publisher is W.W. Norton and this decision to not publish a kindle version highlights that greed is not limited to the banking industry.

Help other customers find the most helpful reviews
Was this review helpful to you? Yes No
Report abuse | Permalink
Comments (14)

3.2 Brand analysis, reputation management, and social mentions

In service industries, a company's stock price is likely to be tied to how its brand is being discussed online (**figure 5**). Similarly, public figures find that their own power and influence are intimately tied to the way they are being talked about on social media.

Figure 5 **Twitter discussion** following Netflix' announcement that it would be increasing its subscription prices. The feedback on **its Facebook page** has a similar tenor and volume. The impact on its stock price has been dramatic.



Results for **#dearnetflix**

RachelBrockway Rachel Brockway
#Deametflix From @mashable: Great Alternatives to Netflix mashable.com/2011/10/26/net...
28 Oct

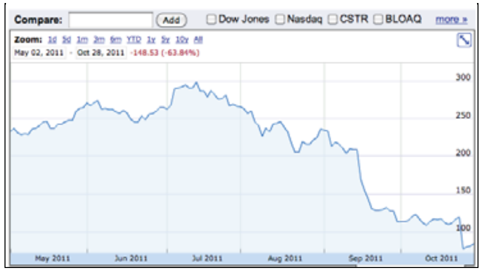
LouScatigna Lou Scatigna
The social media uproar that almost killed Netflix - #DearNetflix - bit.ly/uUh9AT
27 Oct

russhandler Russ Handler
Netflix – The Anatomy of a Self Imploding Company - bit.ly/uUh9AT #DearNetflix
27 Oct

LouScatigna Lou Scatigna
How bad management and repeated debacles brought Netflix to its knees - bit.ly/uUh9AT #DearNetflix
27 Oct

LouScatigna Lou Scatigna
What happens when you take a growing company, and force it to implode do to bad management? Netflix - bit.ly/uUh9AT #DearNetflix
27 Oct

BrandonAgnew Brandon Agnew
#DearNetflix, you should have listened!
25 Oct



Compare: Add □ Dow Jones □ Nasdaq □ CSTR □ BLOAQ more »

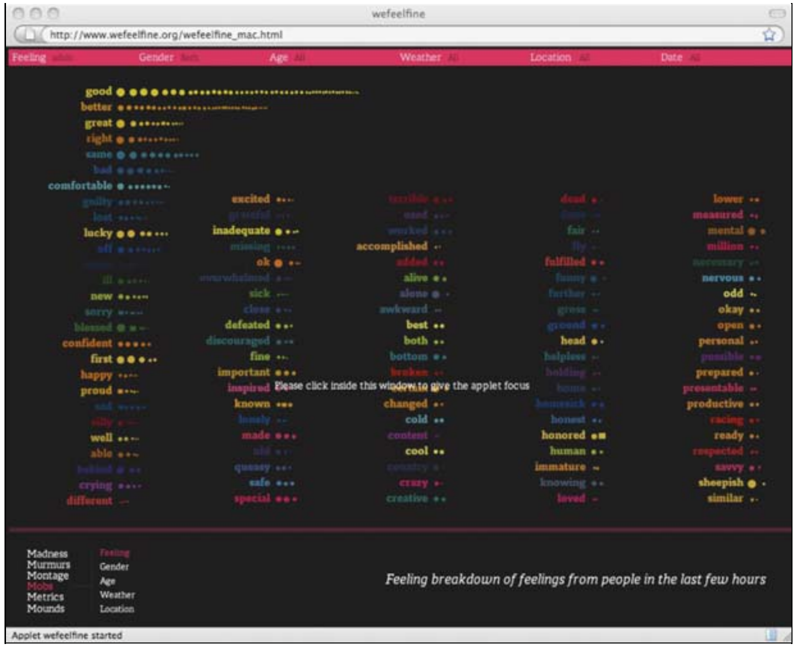
Zoom: 1d 5d 1m 3m 6m YTD 1y 5y 10y All

May 02, 2011 - Oct 28, 2011 -148.53 (-63.84%)

3.3 The emotions of a nation

Large-scale projects like **We Feel Fine** (**figure 6**) and Facebook's **Gross National Happiness** (**figure 7**) seek to provide snapshots of the emotional state of the nation. Such information might be used to time a product launch, find receptive markets, and obtain a deeper understanding of a target audience.

Figure 6 Sentiment "mobs" from **We Feel Fine**.



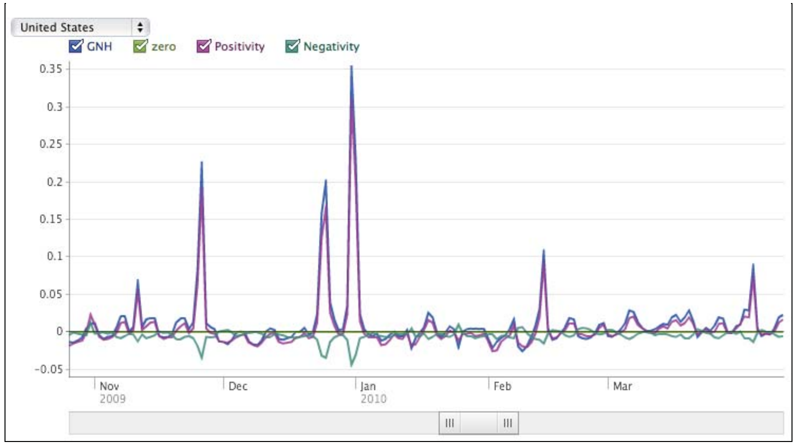


Figure 7 Facebook's **Gross National Happiness** interface. Holidays register large happiness spikes. The happiness dips in January correspond roughly with **the earthquake in Haiti** (Jan 12) and its most serious aftershock (Jan 20).

3.4 Understanding public opinion

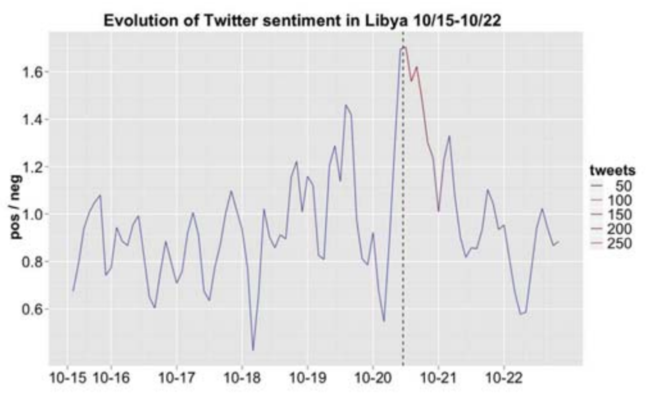
Sentiment analysis can complement and inform public opinion polling:

> We analyze several surveys on consumer confidence and political opinion over the 2008 to 2009 period, and find they correlate to sentiment word frequencies in contemporaneous Twitter messages.
>
> (**O'Connor, Balasubramanyan, Routledge, and Smith 2010**)

3.5 Monitoring real-world events

Linguists at UT Austin are **using sentiment analysis to get a read on the rapidly evolving situation in the middle east**. Apparently, **the CIA is playing this game as well**.

Figure 8 Twitter sentiment in tweets about Libya, from the project **Modeling Discourse and Social Dynamics in Authoritarian Regimes**. The vertical line marks the timing of the announcement that Gaddafi had been killed.



3.6 Media studies

Sentiment analysis is being used to study media coverage and media bias:

> BROOKE GLADSTONE: How do you measure positive and negative press, 'cause you're talkin' about news coverage as much as editorial and opinion.
>
> MARK JURKOWITZ: Yes we are, and this is kind of a new research tool for us. It was a computer algorithm developed by a company called Crimson Hexagon. And we actually used our own human researchers and coders to **train the computer basically to look for positive, negative and neutral assertions**. Our sample was over 11,000 different media outlets.
>
> **The media, the President, and the horse race**. On the Media, Oct. 21, 2011.

3.7 Realistic text-to-speech

Text-to-speech technology is increasingly reliable and sophisticated, but it continues to stumble when it comes to predicting where and how the affect will change:

> Since the mid-1990s, expanding "digital libraries" have allowed for storage of more phonemes that could be split into even smaller units, adding authenticity to the "voice." But even today's state-of-the-art systems, like AT&T’s Natural Voices, still don't capture the range of human emotion.
>
> **The voice in the machine: is lifelike synthetic speech finally within reach?**. The Atlantic Monthly, Nov. 2011.

3.8 Financial prediction

There is evidence that the moods of the nation, as measured by tweets, correlate with changes in stock prices:

> Their results showed that rises and falls in the number of instances of words related to a calm mood could be used to predict the same moves in the Dow's closing price between two and six days later.
>
> **This is how a Twitter-based hedge fund beat the stock market**. The Atlantic Wire, Aug 17, 2011.
>
> (**Bollen, Mau, and Zeng 2010**)

It might also help to predict which movies will open big:

> we use the chatter from Twitter.com to forecast box-office revenues for movies. We show that a simple model built from the rate at which tweets are created about particular topics can outperform market-based predictors. We further demonstrate how sentiments extracted from Twitter can be further utilized to improve the forecasting power of social media.
>
> (**Asur and Huberman 2010**)

For more on such Twitter prognostication, see **this xrXiv.org page**, **this ACL page**, and **this short New York Times piece by Ben Zimmer**.

4 Other publicly available sentiment tutorials

- **R by example: mining Twitter for attitudes towards airlines** (Jeffrey Breen)
- **AAAI Sentiment Analysis Tutorial** (Bing Liu)

- **Sentiment Analysis Tutorial in LingPipe**
- **My Stanford course with Dan Jurafsky: Extracting Social Meaning and Sentiment**
- **Bo Pang and Lillian Lee's Opinion Mining and Sentiment Analysis monograph**

Home | ©2011 **Christopher Potts**

Sentiment Symposium Tutorial: Tokenizing

1. **Overview**
2. **Basic text normalization**
3. **Whitespace tokenizer**
4. **Treebank-style**
5. **Sentiment-aware tokenizer**
    - 5.1 **Emoticons**
    - 5.2 **Twitter mark-up**
    - 5.3 **Informative HTML tags**
    - 5.4 **Masked curses**
    - 5.5 **Additional punctuation**
    - 5.6 **Capitalization**
    - 5.7 **Lengthening**
    - 5.8 **Multi-word expressions**
    - 5.9 **Putting the pieces together**
6. **Evaluation**
    - 6.1 **Classification accuracy**
    - 6.2 **Speed**
7. **Summary of conclusions**

1 Overview

Tokenizing (splitting a string into its desired constituent parts) is fundamental to all NLP tasks.

There is no single right way to do tokenization. The right algorithm depends on the application.

I suspect that tokenization is even more important in sentiment analysis than it is in other areas of NLP, because sentiment information is often sparsely and unusually represented — a single cluster of punctuation like `>:-(` might tell the whole story.

The next few subsections define and illustrate some prominent tokenization strategies. To get a feel for how they work, I illustrate with the following invented tweet-like text:

```
@SentimentSymp:  can't wait for the Nov 9 #Sentiment talks!
YAAAAAAY!!! &gt;:-D http://sentimentsymposium.com/.
```

Demo Experiment with the tokenizers discussed here using your own text or a live Twitter stream:

http://sentiment.christopherpotts.net/tokenizing/

Implementation Code for a basic but extensible sentiment tokenizer:

happyfuntokenizing.py

2 Basic text normalization

I assume throughout that the text has gone through the following preprocessing steps:

1. All HTML and XML mark-up has been identified and isolated.
2. HTML character entities like `&lt;` and `&#60;` have been mapped to their Unicode counterparts (`<` for both of my examples).

These steps should be taken in order, so that one distinguishes the token `<sarcasm>`, which is frequently written out as part of a text, from true HTML mark-up (which is not seen directly but which can play a role in tokenization, as discussed below).

These preprocessing steps affect just the emoticon in the sample text:

```
@SentimentSymp:  can't wait for the Nov 9 #Sentiment talks!
YAAAAAAY!!! >:-D http://sentimentsymposium.com/.
```

3 Whitespace tokenizer

The whitespace tokenizer simply downcases the string and splits the text on any sequence of whitespace, tab, or newline characters:

```
@SentimentSymp:  can't wait for the Nov 9 #Sentiment talks!
YAAAAAAY!!! >:-D http://sentimentsymposium.com/.
```

```
@sentimentsymp:
can't
wait
for
```

```
the
nov
9
#sentiment
talks!
yaaaaaay!!!
>:-d
http://sentimentsymposium.com/.
```

Observations:

- The Twitter username stays intact
- The URL stays intact, but following it would fail because of its final period
- The emoticon is intact but its mouth changed
- Some tokens are unintuitive — for example, `talks!`
- We are likely to mis-identify the Twitter username as `@sentimentsymp:` rather than `@sentimentsymp`

4 Treebank-style

The Treebank-style is the one used by the **Penn Treebank** and many other important large-scale corpora for NLP. Thus, it is a de facto standard. This alone makes it worth considering, since it can facilitate the use of other tools.

```
@SentimentSymp:  can't wait for the Nov 9 #Sentiment talks!
YAAAAAAY!!! >:-D http://sentimentsymposium.com/.
```

```
@
SentimentSymp
:
ca
n't
wait
for
the
Nov
9
#
Sentiment
talks
!
```

```
YAAAAAAY
!
!
!
&gt;
;
:
-D
http
:
//sentimentsymposium.com/
.
```

Some drawbacks to the Treebank style for sentiment:

- Contractions like `can't` contribute their own sentiment, as distinct from co-occurrence of `can` and a negation, whereas the Treebank style splits them into two tokens.
- Almost all tokens that involve punctuation are split apart — URLs, Twitter mark-up, phone numbers, dates, email addresses ... Thus, emoticons are collapsed with their component parts, URLs are not constituents, and Twitter mark-up is lost.
- The purported value of this style is that it can smooth integration with other NLP tools. However, because the tokenization of raw Web text is so bad, later applications are likely to stumble also if used unmodified.

5 Sentiment-aware tokenizer

I now review some of the major aspects of a sentiment-aware tokenizer. You are likely to want to tailor these suggestions to your own data and applications.

Implementation Code for a basic but extensible sentiment tokenizer: **happyfuntokenizing.py**

5.1 Emoticons

Emoticons are extremely common in many forms of social media, and they are reliable carriers of sentiment.

The following regular expression captures 96% of the emoticon tokens occurring on Twitter, as estimated by the **InfoChimps Smileys Census**. (It captures just 36% of the emoticon types, but most are extremely rare and highly confusable with other chunks of text, so I've not

tried to capture them.)

```
[<>]?                          # optional hat/brow
[:;=8]                         # eyes
[\-o\*\']?                     # optional nose
[\)\]\(\[dDpP/\:\}\{@\|\\]    # mouth
|                              #### reverse orientation
[\)\]\(\[dDpP/\:\}\{@\|\\]    # mouth
[\-o\*\']?                     # optional nose
[:;=8]                         # eyes
[<>]?                          # optional hat/brow
```

5.2 Twitter mark-up

Twitter includes topic and user mark-up that is useful for advanced sentiment modeling. Your tokenizer should capture this mark-up if you are processing Twitter data.

Usernames:

```
@+[\w_]+
```

Hashtags (topics):

```
\#+[\w_]+[\w\'_\-]*[\w_]+
```

5.3 Informative HTML tags

Basic HTML mark-up like the `strong`, `b`, `em`, and `i` tags can be indicators of sentiment. Their opening or closing elements can be treated as individual tokens. (Don't count them twice.)

Where sparseness is not an issue, informative tags can be seen as annotating all the words they contain. For strong, `strong`, `b`, `em`, and `i`, I often capitalize them, to collapse them with words written in all caps for emphasis.

```
<strong>really bad idea<strong>
```

becomes

```
REALLY BAD IDEA
```

For domain-specific applications using Web data, it can be fruitful to study the mark-up:

1. Sentiment information implicit in metadata tags like `<span class="rating>2 of 5</span>`
2. Semantic mark-up like `<span class="title>The Good, the Bad, and the Ugly</span>`

5.4 Masked curses

Some websites change curses to sequences of asterisks, perhaps with letters at the edges (`****`, `s***t`). Similarly, some writers use random sequences of non-letter words in place of swears (`$#!@`). Thus, there can be value in treating such sequences as tokens. (I tend to split apart sequences of exclamation points and question marks, though.)

5.5 Additional punctuation

Punctuation should be kept at the tokenization stage. We will shortly use it to identify further structure in the tokenized string. Thus, the goal for tokenizing is to properly distinguish various senses for the individual punctuation marks.

My basic strategy for handling punctuation is to try to identify all the word-internal marks first, so that any others can be tokenized as separate elements. Some considerations:

1. We already tokenized a variety of things that involve word-internal punctation: emoticons, Twitter and HTML mark-up, and masked curses.
2. In general, sequences mixing only letters, numbers, apostrophes, single dashes (hyphens), and underscores are words.
3. Sequences consisting entirely of digits, commas, and periods are likely to be numbers and so can be tokenized as words. Optional leading monetary signs and closing percentage signs are good to allow as well.
4. Sequences of two or more periods are likely to be ellipsis dots and can be collapsed to `...`

The remaining punctuation can be kept as separate words. By and large, this means question marks, exclamation points, and dollar signs without following digits. I find that it works well to tokenize sequences like `!!!` into three separate exclamation marks, and similarly for `!?!?` and the like, since the progression from `!` to `!!` is somewhat additive.

At later stages, you might want to filter some punctuation, because its very high frequency can cause problems for some models. I advise not doing this filtering at the tokenization stage, though, as it can be used to efficiently identify further structure.

5.6 Capitalization

Preserving capitalization across all words can result in unnecessary sparseness. Words written in all caps are generally worth preserving, though, as they tend to be acronyms or words people intended to emphasize, which correlates with sentiment information.

5.7 Lengthening

Lengthening by character repetition is a reliable indicator of heightened emotion. In English, sequences of three or more identical letters in a row are basically unattested in the standard lexicon, so such sequences are very likely to be lengthening.

The amount of lengthening is not predictable, and small differences are unlikely to be meaningful. Thus, it is effective to map sequences of length 3 or greater to sequences of length 3:

Rewrite:

```
(.)\1{2,}
```

as

```
\1\1\1
```

5.8 Multi-word expressions

Even in English, whitespace is only a rough approximation of token-hood in the relevant sense:

1. Named entities
2. Phone numbers
3. Dates
4. Idioms like `out of this world`
5. Multi-word expressions like `absolutely amazing`

The basic strategy is to tokenize these greedily, first, and then proceed to substrings, so that, for example, `November 9` is treated as a single token, whereas an isolated occurrence of `November` is tokenized on its own.

If one starts including n-grams like `really good` as tokens, it is hard to know where to stop.

For large enough collections, bigram or even trigram features might be included (in which case you can tokenize without paying attention to these phrases). For smaller collections, some of the mark-up strategies discussed later on can approximate such information (and often prove more powerful).

5.9 Putting the pieces together

The tokenizer that I use for sentiment seeks to isolate as much sentiment information as possible, and it also identifies and normalizes dates, URLs, phone numbers, and various kinds of digital address. These steps help to keep the vocabulary as small as possible, and they provide chances to identify sentiment in areas that would be overlooked by simpler tokenization strategies (`July 4th`, `September 11`).

Here's the output of my tokenizer on our sample text:

```
@SentimentSymp:  can't wait for the Nov 9 #Sentiment talks!
YAAAAAAY!!! >:-D http://sentimentsymposium.com/.
```

```
@sentimentsymp
:
can't
wait
for
the
Nov_09
#sentiment
talks
!
YAAAY
!
!
!
>:-D
http://sentimentsymposium.com/
.
```

The social-media mark-up is all left intact, the date is normalized, and `YAAAAAAY` has been put into a canonical elongated form.

Demo Experiment with the tokenizers discussed here using your own text or a live Twitter stream:

http://sentiment.christopherpotts.net/tokenizing/

6 Evaluation

How important is careful tokenization for sentiment? Is it worth the extra resources? I now address these questions with some experimental data concerning classifier accuracy and tokenization speed.

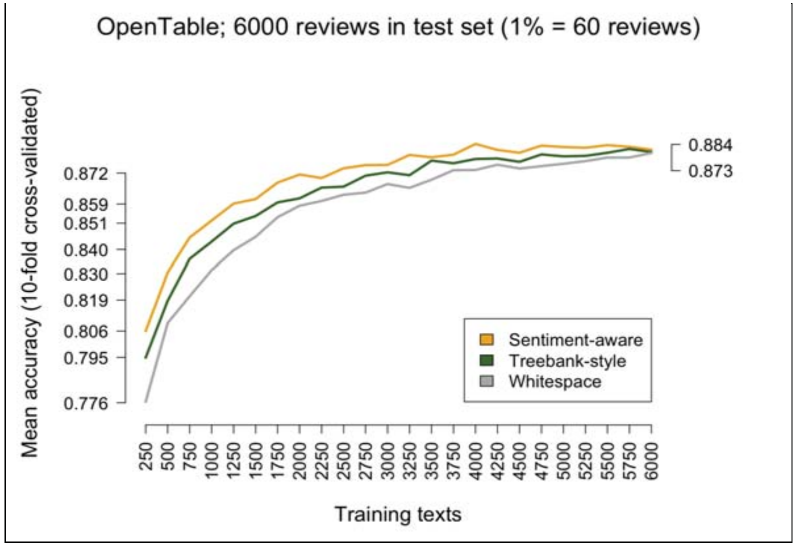
6.1 Classification accuracy

My first classifier accuracy experimental set-up is as follows:

1. I randomly selected 12,000 **OpenTable** reviews. The set was balanced in the sense that 6000 were positive (4-5 stars) and 6000 were negative (1-2 stars).
2. The classifier was **a maximum entropy model**. The features were all the word-level features determined by the tokenizing function in question.
3. The amount of training data is likely to be a major factor in performance, so I tested at training sizes from 250 texts to 6000 texts, in increments of 250.
4. At each training-set size *N*, I performed 10-fold cross-validation: for 10 runs, the data were randomly split into a training set of size *N* and a testing set of size 6000. The accuracy results from these 10 runs were averaged.

Figure 1 reports the results of these experiments.

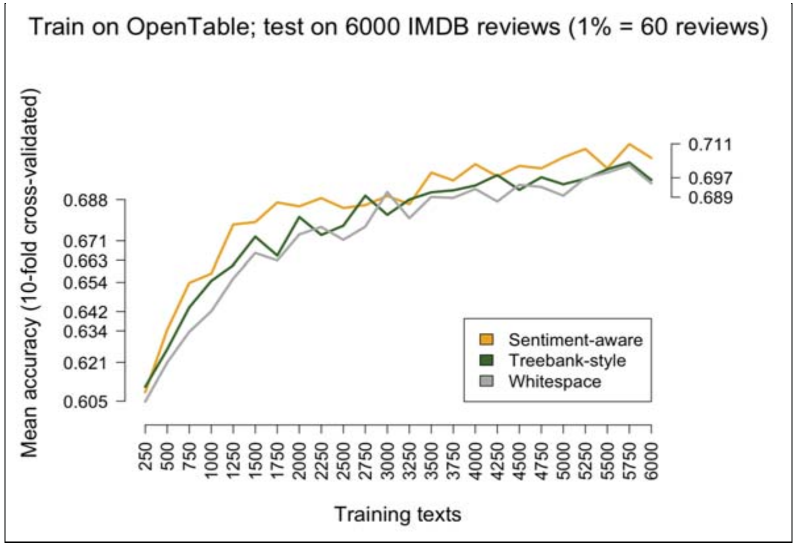
Figure 1 Assessing tokenization algorithms via classification.



In addition, I ran a version of the above experiment where the testing data were drawn, not from the same source as the training data, but rather from another corpus with the same kind of star-rating mark-up — in this case, 6000 user-supplied **IMDB** reviews. Such **out-of-domain testing** provides insight into how portable the classifier model is.

Figure 2 summarizes the results from this experiment. Overall, the performance is less good and more volatile, but sentiment-aware tokenization is still the best option.

Figure 2 Assessing tokenization algorithms via classification: out-of-domain testing on IMDB reviews.



My take-away message is that careful tokenization pays-off, especially where there is relatively little training data available. Where there is a lot of training data, tokenization matters less, since there is enough data for the model to learn that, e.g., `happy` and `happy,"` are basically both the same token, and it becomes less important to capture the effects of any particular word, emoticon, etc.

6.2 Speed

As tokenizers get more complicated, they of necessity become less efficient. **Table 1** illustrates. For many applications, this is not a problem, but it can be a pressing issue if real-time results are needed.

The Catch-22 is that the really fast tokenizers require a lot more data to perform well, whereas the slow tokenizers perform well with limited data.

Tokenization is easily parallelized, so the effects of the slow-down can be mitigated by good infrastructure.

Table 1 Tokenizer speed for 12,000 OpenTable reviews. The numbers are averages for 100 rounds. The average review length is about 50 words.

Tokenizer	Total time (secs)	Average secs/text
Whitespace	1.305	0.0001
Treebank	9.085	0.001
Sentiment	29.915	0.002

7 Summary of conclusions

1. Good tokenizer design will pay off, especially where the amount of training data is limited.
   a. Increased classifier effectiveness
   b. Increased model portability
2. Where there is a large amount of data, careful tokenizing is less important.
3. Good tokenizer design might be especially important for sentiment analysis, where a lot of information is encoded in punctuation and non-standard words.
4. Good tokenizing takes time, which might be an issue for real-time interactive systems. However, the times involved are unlikely to be prohibitive for off-line systems: tokenization is easily parallelized and can be optimized based on known properties of the text.

Sentiment Symposium Tutorial: Stemming

1. **Overview**
2. **Porter stemmer**
3. **Lancaster stemmer**
4. **WordNet stemmer**
5. **Assessment**
   5.1 **Classification accuracy**
   5.2 **Speed**
6. **Summary of conclusions**

1 Overview

Stemming is a method for collapsing distinct word forms. This could help reduce the vocabulary size, thereby sharpening one's results, especially for small data sets.

This section reviews three common stemming algorithms in the context of sentiment: the Porter stemmer, the Lancaster stemmer, and the WordNet stemmer.

My overall conclusion is that the Porter and Lancaster stemmers destroy too many sentiment distinctions. The WordNet stemmer does not have this problem nearly so severely, but it doesn't do enough collapsing to be worth the resources necessary to run it.

Demo See the effects of Porter stemming on our own input texts: **http://sentiment.christopherpotts.net/tokenizing/**

2 Porter stemmer

The **Porter stemmer** is one of the earliest and best-known stemming algorithms. It works by heuristically identifying word suffixes (endings) and stripping them off, with some regularization of the endings.

The Porter stemmer often collapses sentiment distinctions, by mapping two words with different sentiment into the same stemmed form. **Table 1** provides examples of such collapsing relative to the disjoint Positiv/Negativ classess of **the Harvard General Inquirer**, a large gold-standard semantic resource containing extensive sentiment information.

Table 1 Porter stemming. 36 instances in which an Harvard Inquirer Positiv/Negativ distinction is destroyed by the algorithm.

Positiv	Negativ	Stemmed
captivation	captive	captiv
common	commoner	common
defend	defendant	defend
defense	defensive	defens
dependability	dependent	depend
dependable	dependent	depend
desirable	desire	desir
dominance	dominate	domin
dominance	domination	domin
extravagance	extravagant	extravag
home	homely	home
pass	passe	pass
patron	patronize	patron
prosecute	prosecution	prosecut
affection	affectation	affect
capitalize	capital	capit
closeness	close	close
commitment	commit	commit
competence	compete	compet
competency	compete	compet
competent	compete	compet
conviction	convict	convict
defender	defendant	defend
desirous	desire	desir
impetus	impetuous	impetu
indulgence	indulge	indulg
objective	object	object
objective	objection	object
rational	ration	ration
subsidize	subside	subsid
temperance	temper	temper
temperate	temper	temper
tolerance	tolerable	toler
tolerant	tolerable	toler
tolerate	tolerable	toler
toleration	tolerable	toler

3 Lancaster stemmer

The **Lancaster stemmer** is another widely used stemming algorithm. However, for sentiment analysis, it is arguably even more problematic than the Porter stemmer, since it collapses even more words of differing sentiment. **Table 2** illustrates with a randomly chosen selection of such collapses, again using the Harvard General Inquirer's Positiv/Negativ distinction as a gold standard.

Table 2 Lancaster stemming. 50 randomly selected instances in which a Harvard Inquirer Positiv/Negativ distinction is destroyed by the algorithm.

Positiv	Negativ	Stemmed
apprehend	apprehensive	apprehend
arbitrate	arbitrary	arbit
arbitration	arbitrary	arbit
audible	audacious	aud
call	callous	cal

capitalize	capital	capit
captivation	capture	capt
captivation	captive	capt
comical	commiseration	com
comely	commiseration	com
comic	commiseration	com
commitment	commit	commit
competency	compete	compet
compliment	complicate	comply
compliment	complication	comply
consummate	consumptive	consum
content	conceal	cont
contentment	conceal	cont
conviction	convict	convict
credentials	credulous	cred
credibility	credulous	cred
cute	cut	cut
deference	defeat	def
defender	defensive	defend
defend	defensive	defend

Positiv	**Negativ**	**Stemmed**
defend	defendant	defend
dependability	dependent	depend
desirous	desire	desir
dominance	dominate	domin
famous	famished	fam
fill	filth	fil
flourish	floor	flo
meaningful	mean	mean
notoriety	notorious	not
notable	notorious	not
passionate	passe	pass
pass	passe	pass
patronage	patronize	patron
rational	ration	rat
refuge	refugee	refug
repentance	repeal	rep
repent	repeal	rep
ripe	rip	rip
savings	savage	sav
simplify	simplistic	simpl
simplicity	simplistic	simpl
suffice	sufferer	suff
temperate	temper	temp
tolerant	tolerable	tol
truth	truant	tru

4 WordNet stemmer

WordNet (Fellbaum 1998) has high-precision stemming functionality, but it is probably of limited use for sentiment analysis. To effect real change, it requires (word, part-of-speech tag) pairs, where the part-of-speech is a, n, r (adverb), or v. When given such pairs, it collapses tense, aspect, and number marking.

The only danger I know of for sentiment analysis is that it collapses base, comparative, and superlative adjective forms. Table 3 provides some illustrations.

Table 3 WordNet stemming. Representative examples of what the stemmer does and doesn't do. Collapsing adjectival forms is the only worrisome behavior when it comes to sentiment.

Word	**Stemmed**

(exclaims, v)	exclaim
(exclaimed, v)	exclaim
(exclaiming, v)	exclaim
(exclamation, n)	exclamation
(proved, v)	prove
(proven, v)	prove
(proven, a)	proven
(happy, a)	happy
(happier, a)	happy
(happiest, a)	happy

5 Assessment

5.1 Classification accuracy

To assess the impact of the stemming algorithms, I use **the experimental set-up as I used when assessing tokenizers**. Here, though, I compare just the plain sentiment tokenizer with the sentiment tokenizer plus the Porter and Lancaster stemmers, applied to the output of sentiment tokenization. (Since the WordNet stemmer requires part-of-speech tagged data, and since its changes are minimal, I don't assess it here.)

The results of the experiment are given in **figure 1**. It looks like the stemmers benefit somewhat from their reduced vocabulary size when the amount of training data is small, though not enough to improve on the sentiment-aware tokenizer, and they are quickly out-paced as the training data grows.

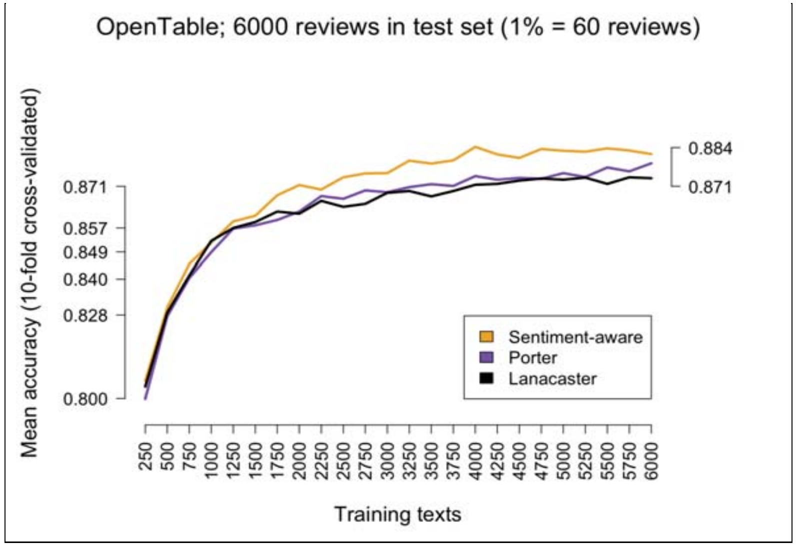


Figure 1 Assessing stemming algorithms via classification. (**Details on the experimental design**.)

5.2 Speed

Table 4 extends **the tokenizer speed assessment given earlier** with comparable numbers for the stemming algorithms.

Table 4 Tokenizer speed for 12,000 OpenTable reviews. The numbers are averages for 100 rounds. The average review length is about 50 words.

Tokenizer	Total time (secs)	Average secs/text
Whitespace	1.305	0.0001
Treebank	9.085	0.001
Sentiment	29.915	0.002
Sentiment + Porter stemming	49.471	0.004
Sentiment + Lancaster stemming	62.938	0.005

6 Summary of conclusions

1. My central conclusion is that, for sentiment analysis, running a stemmer is costly in terms of resources and performance accuracy.
2. I can imagine running the WordNet stemmer if matching against a restricted vocabulary became important, but in that case it would be better to run the algorithm in reverse to expand the word list.
3. I don't mean to suggest that stemming could never help with sentiment analysis, but rather only that these off-the-shelf stemming algorithms can weaken sentiment systems.

Home |

Sentiment tutorial home **Christopher Potts**, **Stanford Linguistics**

Sentiment Symposium Tutorial: Linguistic structure

1. **Overview**
2. **Negation**
    2.1 **The issue**
    2.2 **The approach**
    2.3 **Assessment**
3. **Other scope marking**
4. **Part-of-speech tagging**
5. **Dependency parsing**
6. **Summary of conclusions**

1 Overview

So far, the only structure we've imposed on our texts is to (carefully) **turn them into lists of tokens**. The present section explores practical methods for building on that basic structure by identifying semantic groupings and relationships that are relevant for sentiment.

Demo See the effects of negation marking on our own input texts: **http://sentiment.christopherpotts.net/tokenizing/**

Demo The Stanford parser has an online interface: **http://nlp.stanford.edu:8080/parser/**

2 Negation

Sentiment words behave very differently when under the semantic scope of negation. The goal of this section is to present and support a method for approximating this semantic influence.

2.1 The issue

The rules of thumb for how negation interacts with sentiment words are roughly as follows:

1. Weak (mild) words such as `good` and `bad` behave like their opposites when negated: `bad` ≈ `not good`; `good` ≈ `not bad`.
2. Strong (intense) words like `superb` and `terrible` have very general meanings under negation: `not superb` is consistent with everything from horrible to just-shy-of-superb, and different lexical items favor different senses.

These observations suggest that it would be difficult to have a general a priori rule for how to handle negation. It doesn't just turn good to bad and bad to good. Its effects depend on the words being negated.

An additional challenge for negation is that its expression is lexically diverse and its influences are far-reaching (syntactically speaking). In all of the following, for example, we have something like `neg-enjoy`:

1. I didn't enjoy it.
2. I never enjoy it.
3. No one enjoys it.
4. I have yet to enjoy it.
5. I don't think I will enjoy it.

Paying attention to bigrams like `not enjoy` would partially address the challenge, but it would leave out a lot of the effects of negation.

We could capture the effects of negation fairly exactly if we had accurate semantic parses and were willing to devote extensive time and energy to implementing the relevant principles of semantic scope-taking. Though I myself love such projects, I suspect that it would not be worth the effort. The approximate method I develop next does extremely well.

2.2 The approach

The method I favor for approximating the effects of negation is due to **Das and Chen 2001** and **Pang, Lee, and Vaithyanathan 2002**. It works as as follows:

Definition: Negation marking

> Append a `_NEG` suffix to every word appearing between a negation and a clause-level punctuation mark.

Negation and clause-level punctuation are defined as follows:

Definition: Negation

> A negation is any word matching the following regular expression:

```
(?:
    ^(?:never|no|nothing|nowhere|noone|none|not|
        havent|hasnt|hadnt|cant|couldnt|shouldnt|
        wont|wouldnt|dont|doesnt|didnt|isnt|arent|aint
    )$
)
|
n't
```

Definition: Clause-level punctuation

> A clause-level punctuation mark is any word matching the following regular expression:

```
^[.:;!?]$
```

The sentiment tokenization strategy we've been using makes this straightforward, since it isolates the clausal punctation from the word-internal punctuation.

The algorithm captures simple negations:

```
No one enjoys it.
```

```
no
one_NEG
enjoys_NEG
it_NEG
.
```

It also handles long-distance effects:

```
I don't think I will enjoy it: it might be too spicy.
```

```
i
don't
think_NEG
i_NEG
will_NEG
```

```
enjoy_NEG
it_NEG
:
it
might
be
too
spicy
.
```

The algorithm has literally turned `enjoy` into two tokens: `enjoy` outside of the scope of negation, `enjoy_NEG` inside the scope of negation. Thus, we needn't stipulate a relationship between the negated and un-negated forms; the sentiment analysis system can learn how the two behave.

The above regular expression for clause boundaries does not include commas, which are often used for sub-clausal boundaries. However, some applications might benefit from the more conservative behavior that comes from treating the comma as a boundary. This would prevent neg-marking of `but I might` in the following example:

```
I don't think I will enjoy it, but I might.
```

Demo See the effects of negation marking on our own input texts: **http://sentiment.christopherpotts.net/tokenizing/**

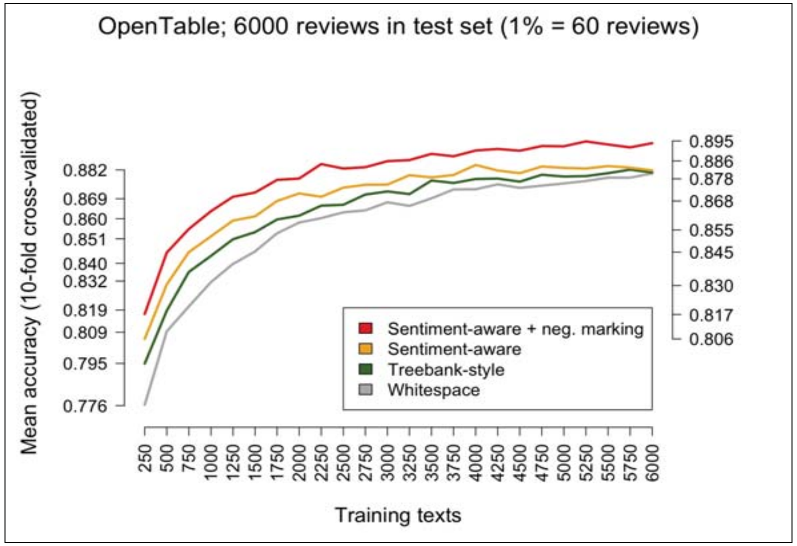
2.3 Assessment

Figure 1 extends **the tokenizer assessment we did earlier** with data on the sentiment-aware tokenizer plus negation marking. As you can see, negation marking results in substantial gains at all amounts of training data.

Negation marking is very difficult to implement for the other tokenizer strategies, since they either fail to reliably separate punctuation (**Whitespace**) or they separate too much of it (**Treebank**). Thus, I've not tried to do negation marking on their output.

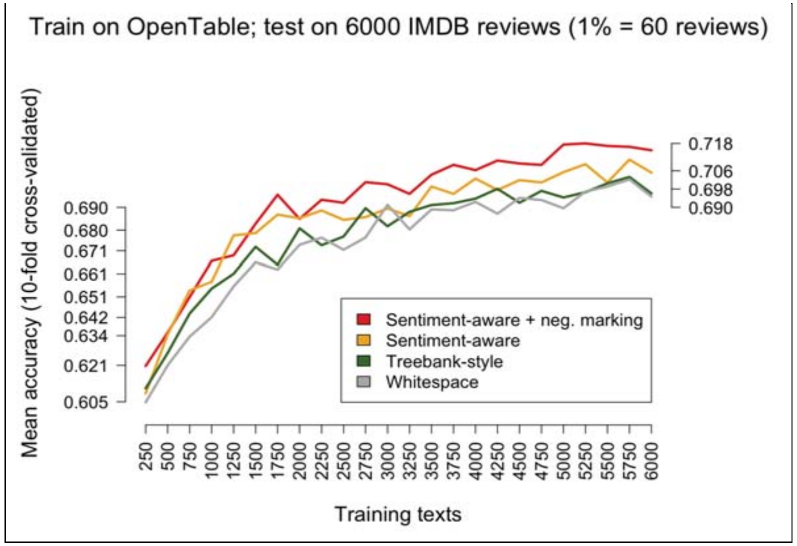
I should add also that negation marking is most important when the texts are short. For them, a single `never very good` might be the only indication of negativity, whereas longer texts tend to contain more sentiment cues and are thus less dependent on any single one.

Figure 1 Assessing the value of negation marking.



Negation marking also helps substantially with out-of-domain testing, as we see in **Figure 2**. Whereas the sentiment tokenizer alone only barely improves on the simpler methods, it leaps ahead of them when negation marking is added.

Figure 2 Assessing the value of negation marking by training on OpenTable data and testing on IMDB user-supplied movie reviews.



3 Other scope marking

Scope marking is also effective for marking quotation and the effects of attitude reports like **say**, **claim**, etc., which are often used to create distance between the speaker's commitments and those of others:

1. They said it would be horrible, but they were wrong: I loved it!!!
2. This "wonderful" car turned out to be a piece of junk.

For quotation, the strategy is to turn it on and off at quotation marks. To account for nesting, one can keep a counter (though nesting is rare). For attitude verbs, the strategy is the same as the one for negation: **_REPORT** marking between the relevant predicates and clause-level punctuation.

The only concern is that small data sets might not support the very large increase in vocabulary size that this marking will produce.

4 Part-of-speech tagging

There are many cases in which a sentiment contrast exists between words that have the same string representation but different parts of speech. **Table 1** provides a sample, using the **Harvard General Inquirer**'s disjoint Positiv/Negativ classes to assess the polarity values.

Table 1 Sentiment contrasts between identical strings with differing parts of speech.

Word	Tag1	Polarity	Tag2	Polarity
arrest	jj	Positiv	vb	Negativ
board	jj	Positiv	rb	Negativ
deal	intj	Negativ	nn	Positiv
even	jj	Positiv	vb	Negativ
even	rb	Positiv	vb	Negativ
fine	jj	Positiv	nn	Negativ
fine	jj	Positiv	vb	Negativ
fine	nn	Negativ	rb	Positiv
fine	rb	Positiv	vb	Negativ
fun	nn	Positiv	vb	Negativ
help	jj	Positiv	vbn	Negativ
help	nn	Positiv	vbn	Negativ
help	vb	Positiv	vbn	Negativ
hit	jj	Negativ	vb	Positiv
matter	jj	Negativ	vb	Positiv
mind	nn	Positiv	vb	Negativ
order	jj	Positiv	vb	Negativ
order	nn	Positiv	vb	Negativ
pass	nn	Negativ	vb	Positiv

This suggests that it might be worth the effort to run a part-of-speech tagger on your sentiment data and then use the resulting word–tag pairs as features or components of features.

I look to the **Stanford Log-Linear Part-of-Speech Tagger** (**Toutanova, Klein, Manning, and Singer**) for all my tagging needs. It is extremely well documented, it comes with sample Java code to facilitate incorporating it into other Java programs, it has a flexible command-line interface in case your existing system is in another programming language, and it is impressively fast (15000 words/second on a standard 2008 desktop). It is free for academic research and has very reasonably priced, ready-to-sign **commercial licenses**.

The only challenge is that the default tokenization is **Treebank style**. We saw in **the tokenizing unit** that this is a sub-optimal tokenizing strategy for sentiment applications.

However, it is straightforward to write new tokenizers to feed the POS-tagger. If you do this, there will be tokens that it handles incorrectly, but these errors might not affect your sentiment analysis.

To illustrate, let's work a bit with the following file, which I'll call `foo.txt`. The format is one sentence per line:

```
It is fine if you fine me, but don't rub it in!
The play was a hit until someone hit the lead actor. :-(
I found it to be booooring!
```

Suppose we tokenize this using the sentiment tokenizer and then store it on a single line, space-separated, in a file called `foo.tok`. The output:

```
It is fine if you fine me , but don't rub it in !
The play was a hit until someone hit the lead actor . :-(
I found it to be booooring !
```

Then the command-line call we want is as follows (I assume that you are in the same directory as the Part-of-Speech Tagger):

```
java -mx3000m -cp stanford-postagger.jar
edu.stanford.nlp.tagger.maxent.MaxentTagger -model models/bidirectional-distsim-
wsj-0-18.tagger -outputFormat slashTags -tagSeparator / -tokenize false -
textFile foo.txt
```

The output (in the actual output, each example is on a single line with no blank lines between):

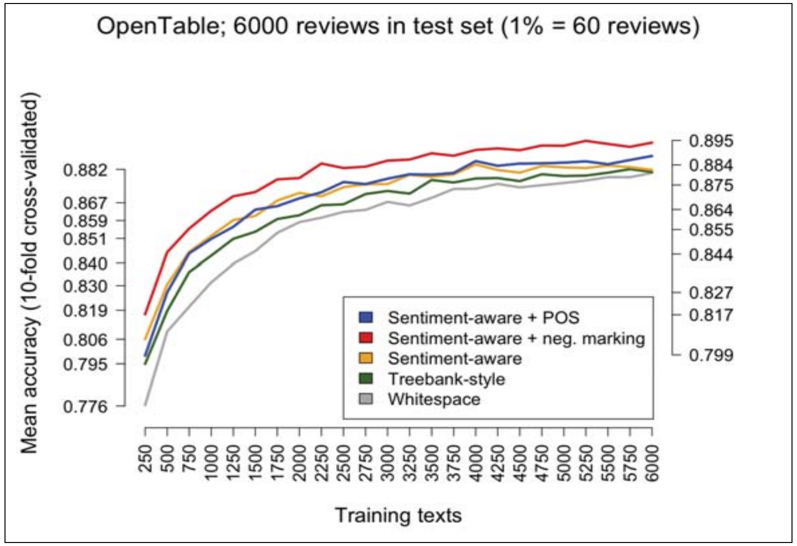
```
It/PRP is/VBZ fine/JJ if/IN you/PRP fine/VBP
me/PRP ,/, but/CC don't/NN rub/VBP it/PRP in/IN !/.

The/DT play/NN was/VBD a/DT hit/NN until/IN someone/NN hit/VBD
the/DT lead/JJ actor/NN ./. :-(/NN

I/PRP found/VBD it/PRP to/TO be/VB booooring/VBG !/.
```

Because the Treebank-style turns contractions into two tokens, the tagger gets the tag for `can't` wrong (it wants to see `ca` and `n't`). This can have detrimental effects on the surrounding tags, so it is perhaps worth pursuing a blend of the sentiment and Treebank styles if POS-tagging is going to be a large part of your system. However, the errors might not be problematic, since they seem to be very consistent.

For more on using the tagger efficiently from the command-line, see **Matt Jockers' tutorial**.

Figure 3 uses **our familiar classifier assessment** to see what POS-tagging contributes. The performance is not all that different from the regular sentiment-aware tokenization, but we shouldn't conclude from this that POS-tagging is not valuable. Wider testing might reveal that there are situations in which it helps substantially.

Figure 3 Assessing POS-tagging.



5 Dependency parsing

Dependency parsing transforms a sentence into a quasi-semantic structure that can be extremely useful for extracting sentiment information, particularly where the goal is to relativize the sentiment information to particular entities or topics.

The **Stanford Parser** (**Klein and Manning 2003a, b**) can map raw strings, tokenized strings, or POS-tagged strings, to dependency structures (and others). I focus here on the Stanford Dependencies (**de Marneffe, Manning, and MacCartney 2006**).

There isn't space here to review dependency parsing in detail, but a few simple examples probably suffice to convey how this tool can be used in sentiment analysis.

I put the output of **our tagging sample above** into a file called `foo.tagged` and then ran the following command from inside the parser's distribution directory:

```
java -mx3000m -cp stanford-parser.jar
edu.stanford.nlp.parser.lexparser.LexicalizedParser -outputFormat
"typedDependencies" -tokenized -tagSeparator / englishPCFG.ser.gz foo.tagged
```

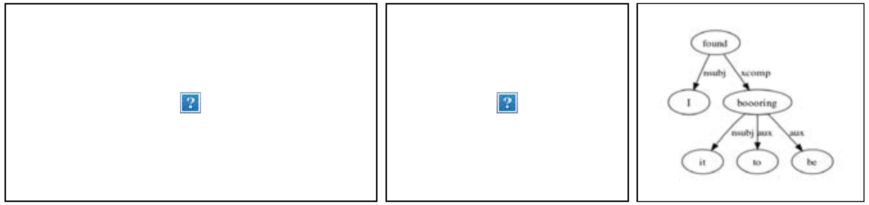
The output is a list of sequences of dependecy-graph edges, with indices keeping track of linear order:

```
nsubj(fine-3, It-1)
cop(fine-3, is-2)
mark(fine-6, if-4)
nsubj(fine-6, you-5)
advcl(fine-3, fine-6)
dobj(fine-6, me-7)
nsubj(rub-11, don't-10)
conj_but(fine-3, rub-11)
dobj(rub-11, it-12)
prep(rub-11, in-13)

det(play-2, The-1)
nsubj(hit-5, play-2)
cop(hit-5, was-3)
det(hit-5, a-4)
mark(hit-8, until-6)
nsubj(hit-8, someone-7)
advcl(hit-5, hit-8)
det(actor-11, the-9)
amod(actor-11, lead-10)
dobj(hit-8, actor-11)

nsubj(found-2, I-1)
nsubj(boooring-6, it-3)
```

```
aux(boooring-6, to-4)
aux(boooring-6, be-5)
xcomp(found-2, boooring-6)
```

Figure 4 provides the graphical structure for these examples:

Figure 4 Stanford Dependency structures.



Such dependency can isolate not only what the sentiment of a text is but also where that sentiment is coming from and whom it is directed at.

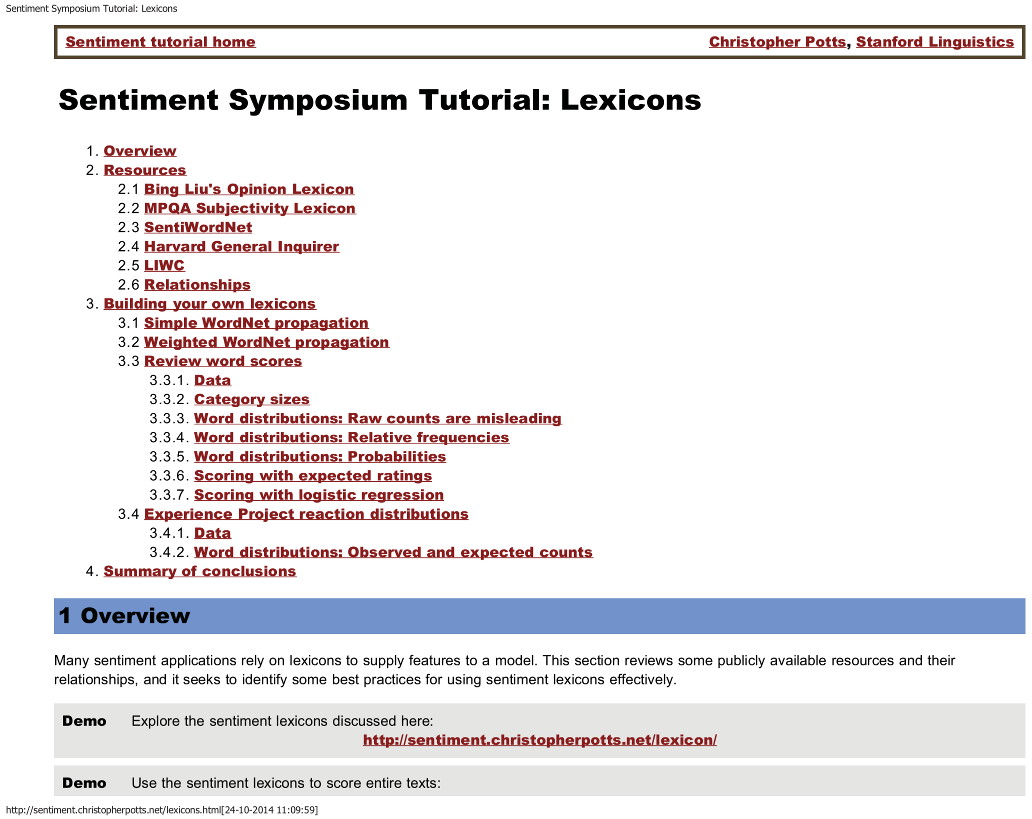
Demo The Stanford parser has an online interface:
http://nlp.stanford.edu:8080/parser/

Like the POS-tagger, the Parser is free for academic use and has reasonably priced, ready-to-sign **commercial licenses**.

6 Summary of conclusions

1. Negation marking is helpful at all amounts of training data, in both in-domain and out-of-domain testing.
2. Other kind of semantic marking (particularly for reported information) is also likely to be helpful where there is enough data to support the much larger vocabulary that this will create.
3. POS-tagging did not have much of an impact in our experimental design, but POS distinctions do matter for sentiment, so it might nonetheless be worth the resources. POS-tagging is often a step along the way to richer analyses as well.
4. Dependency structures identify useful semantic relationships.

Home | ©2011 **Christopher Potts**



Sentiment tutorial home **Christopher Potts**, **Stanford Linguistics**

Sentiment Symposium Tutorial: Lexicons

1. **Overview**
2. **Resources**
   - 2.1 **Bing Liu's Opinion Lexicon**
   - 2.2 **MPQA Subjectivity Lexicon**
   - 2.3 **SentiWordNet**
   - 2.4 **Harvard General Inquirer**
   - 2.5 **LIWC**
   - 2.6 **Relationships**
3. **Building your own lexicons**
   - 3.1 **Simple WordNet propagation**
   - 3.2 **Weighted WordNet propagation**
   - 3.3 **Review word scores**
     - 3.3.1. **Data**
     - 3.3.2. **Category sizes**
     - 3.3.3. **Word distributions: Raw counts are misleading**
     - 3.3.4. **Word distributions: Relative frequencies**
     - 3.3.5. **Word distributions: Probabilities**
     - 3.3.6. **Scoring with expected ratings**
     - 3.3.7. **Scoring with logistic regression**
   - 3.4 **Experience Project reaction distributions**
     - 3.4.1. **Data**
     - 3.4.2. **Word distributions: Observed and expected counts**
4. **Summary of conclusions**

1 Overview

Many sentiment applications rely on lexicons to supply features to a model. This section reviews some publicly available resources and their relationships, and it seeks to identify some best practices for using sentiment lexicons effectively.

Demo Explore the sentiment lexicons discussed here: **http://sentiment.christopherpotts.net/lexicon/**

Demo Use the sentiment lexicons to score entire texts:

http://sentiment.christopherpotts.net/textscores/

Demo Simple WordNet propagation:
http://sentiment.christopherpotts.net/wnpropagate/

Data and code
A variety of data sets and Python/NLTK implementations

2 Resources

2.1 Bing Liu's Opinion Lexicon

Bing Liu maintains and freely distributes a sentiment lexicon consisting of lists of strings.

- Distribution page (direct link to rar archive)
- Positive words: 2006
- Negative words: 4783
- Useful properties: includes mis-spellings, morphological variants, slang, and social-media mark-up

2.2 MPQA Subjectivity Lexicon

The MPQA (Multi-Perspective Question Answering) Subjectivity Lexicon is maintained by Theresa Wilson, Janyce Wiebe, and Paul Hoffmann (Wiebe, Wilson, and Cardie 2005). It is distributed under a GNU Public License. Table 1 shows what its structure is like.

Table 1 A fragment of the MPQA subjectivity lexicon.

	Strength	Length	Word	Part-of-speech	Stemmed	Polarity
1.	type=weaksubj	len=1	word1=abandoned	pos1=adj	stemmed1=n	priorpolarity=negative
2.	type=weaksubj	len=1	word1=abandonment	pos1=noun	stemmed1=n	priorpolarity=negative
3.	type=weaksubj	len=1	word1=abandon	pos1=verb	stemmed1=y	priorpolarity=negative
4.	type=strongsubj	len=1	word1=abase	pos1=verb	stemmed1=y	priorpolarity=negative
5.	type=strongsubj	len=1	word1=abasement	pos1=anypos	stemmed1=y	priorpolarity=negative
6.	type=strongsubj	len=1	word1=abash	pos1=verb	stemmed1=y	priorpolarity=negative
7.	type=weaksubj	len=1	word1=abate	pos1=verb	stemmed1=y	priorpolarity=negative
8.	type=weaksubj	len=1	word1=abdicate	pos1=verb	stemmed1=y	priorpolarity=negative
9.	type=strongsubj	len=1	word1=aberration	pos1=adj	stemmed1=n	priorpolarity=negative
10.	type=strongsubj	len=1	word1=aberration	pos1=noun	stemmed1=n	priorpolarity=negative
...						
8221.	type=strongsubj	len=1	word1=zest	pos1=noun	stemmed1=n	priorpolarity=positive

2.3 SentiWordNet

SentiWordNet (*note: this site was hacked recently; take care when visiting it*) (Baccianella, Esuli, and Sebastiani 2010) attaches positive and negative real-valued sentiment scores to WordNet synsets (Fellbaum1998). It is freely distributed for noncommercial use, and licensed are available for commercial applications. (See the website for details.) Table 2 summarizes its structure. (For extensive discussion of WordNet synsets and related objects, see this introduction).

Table 2 A fragment of the SentiWordNet database.

POS	ID	PosScore	NegScore	SynsetTerms	Gloss
a	00001740	0.125	0	able#1	(usually followed by `to') having the necessary means or [...]
a	00002098	0	0.75	unable#1	(usually followed by `to') not having the necessary means or [...]
a	00002312	0	0	dorsal#2 abaxial#1	facing away from the axis of an organ or organism; [...]
a	00002527	0	0	ventral#2 adaxial#1	nearest to or facing toward the axis of an organ or organism; [...]
a	00002730	0	0	acroscopic#1	facing or on the side toward the apex
a	00002843	0	0	basiscopic#1	facing or on the side toward the base
a	00002956	0	0	abducting#1 abducent#1	especially of muscles; [...]
a	00003131	0	0	adductive#1 adducting#1 adducent#1	especially of muscles; [...]
a	00003356	0	0	nascent#1	being born or beginning; [...]
a	00003553	0	0	emerging#2 emergent#2	coming into existence; [...]

2.4 Harvard General Inquirer

The **Harvard General Inquirer** is a lexicon attaching syntactic, semantic, and pragmatic information to part-of-speech tagged words (**Stone, Dunphry, Smith, and Ogilvie 1966**). The **spreadsheet format** is the easiest one to work with for most computational applications. **Table 3** provides a glimpse of the richness and complexity of this resource.

Table 3 A fragment of the Harvard General Inquirer spreadsheet file.

	Entry	Positiv	Negativ	Hostile	...184 classes ...	Othtags	Defined
1	A					DET ART	...
2	ABANDON		Negativ			SUPV	
3	ABANDONMENT		Negativ			Noun	
4	ABATE		Negativ			SUPV	
5	ABATEMENT					Noun	
...							
35	ABSENT#1		Negativ			Modif	
36	ABSENT#2					SUPV	
...							
11788	ZONE					Noun	

2.5 LIWC

Linguistic Inquiry and Word Counts (LIWC) is a propriety database consisting of a lot of categorized regular expressions. It costs about $90. Its classifications are highly correlated with those of the Harvard General Inquirer. **Table 4** gives some of its sentiment-relevant categories with example regular expressions.

Table 4 A fragment of the LIWC database.

Category	Examples
Negate	aint, ain't, arent, aren't, cannot, cant, can't, couldnt, ...
Swear	arse, arsehole*, arses, ass, asses, asshole*, bastard*, ...
Social	acquainta*, admit, admits, admitted, admitting, adult, adults, advice, advis*
Affect	abandon*, abuse*, abusi*, accept, accepta*, accepted, accepting, accepts, ache*
Posemo	accept, accepta*, accepted, accepting, accepts, active*, admir*, ador*, advantag*
Negemo	abandon*, abuse*, abusi*, ache*, aching, advers*, afraid, aggravat*, aggress*,
Anx	afraid, alarm*, anguish*, anxi*, apprehens*, asham*, aversi*, avoid*, awkward*
Anger	jealous*, jerk, jerked, jerks, kill*, liar*, lied, lies, lous*, ludicrous*, lying, mad

2.6 Relationships

All of the above lexicons provide basic polarity classifications. Their underlying vocabularies are different, so it is difficult to compare them comprehensively, but we can see how often they explicitly disagree with each other in that they supply opposite polarity values for a given word. **Table 5** reports on the results of such comparisons.

(Where a lexicon had part-of-speech tags, I removed them and selected the most sentiment-rich sense available for the resulting string. For SentiWordNet, I counted a word as positive if its positive score was larger than its negative score; negative if its negative score was larger than its positive score; else neutral, which means that words with equal non-0 positive and negative scores are neutral.)

Table 5 Disagreement levels for the sentiment lexicons reviewed above.

	MPQA	Opinion Lexicon	Inquirer	SentiWordNet	LIWC
MPQA	–	33/5402 (0.6%)	49/2867 (2%)	1127/4214 (27%)	12/363 (3%)
Opinion Lexicon		–	32/2411 (1%)	1004/3994 (25%)	9/403 (2%)
Inquirer			–	520/2306 (23%)	1/204 (0.5%)
SentiWordNet				–	174/694 (25%)
LIWC					–

I can imagine two equally reasonable reaction to the disagreements. The first would be to resolve them in favor of some particular sense. The second would be to combine the values derived from theses resources, thereby allowing the conflicts to persist, as a way of capturing the fact that the disagreements arise from genuine sense ambiguities.

Demo Explore the sentiment lexicons discussed here: **http://sentiment.christopherpotts.net/lexicon/**

Demo Use the sentiment lexicons to score entire texts: **http://sentiment.christopherpotts.net/textscores/**

3 Building your own lexicons

The above lexicons are useful for a wide range of tasks, but they are fixed resources. This section is devoted to developing new resources. This can have three benefits, which we will see in various combinations:

1. Much larger lexicons can be developed inferentially.
2. We can capture different dimensions of sentiment that might be pressing for specific tasks.
3. We can develop lexicons that are sensitive to the norms of specific domains.

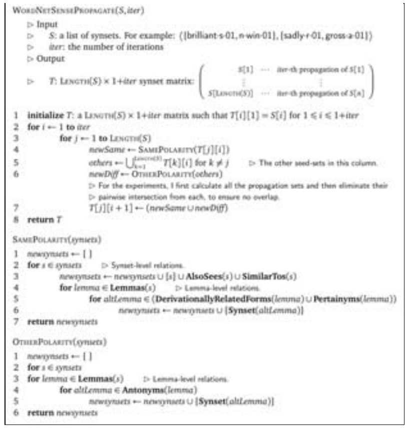
3.1 Simple WordNet propagation

The guiding idea behind simple WordNet propagation is the properties of some hand-selected seed-sets will be preserved as we travel strategically through WordNet (**Hu and Liu 2004** **Andreevskaia and Bergler 2006** **Esuli and Sebastiani 2006** **Kim and Hovy 2006** **Godbole, Srinivasaiah, and Skiena 2007** **Rao and Ravichandran 2009**).

The algorithm begins with *n* small, hand-crafted seed-sets and then follows WordNet relations from them, thereby expanding their size. The expanded sets of iteration *i* are used as seed-sets for iteration *i+1*, generally after pruning any pairwise overlap between them.

The algorithm is spelled out in full in **figure 1.**

Figure 1 Free hyper parameters: the seed-sets, the WordNet relations called in SamePolarity and OtherPolarity, the number of iterations, the decision to remove overlap.



```
WordNetSenseFropagate(S, iter)
    ▷ Input
    ▷     S: a list of synsets. For example: ([brilliant-s-01, n-win-01], [sadly-r-01, gross-a-01])
    ▷     iter: the number of iterations
    ▷ Output
    ▷     T: Length(S) × 1+iter synset matrix: ( S[1] ··· iter-th propagation of S[1] ; ⋮ ⋮ ; S[Length(S)] ··· iter-th propagation of S[n] )

initialize T: a Length(S) × 1+iter matrix such that $T[i][1] = S[i]$ for $1 \leqslant i \leqslant 1+iter$
for i ← 1 to iter
    for j ← 1 to Length(S)
        newSame ← SamePolarity(T[j][i])
        others ← $\bigcup_{k=1}^{\mathrm{Length}(S)} T[k][i]$ for $k \neq j$        ▷ The other seed-sets in this column.
        newDiff ← OtherPolarity(others)
        ▷ For the experiments, I first calculate all the propagation sets and then eliminate their
        ▷ pairwise intersection from each, to ensure no overlap.
        T[j][i + 1] ← (newSame ∪ newDiff)
return T

SamePolarity(synsets)
newsynsets ← [ ]
for s ∈ synsets        ▷ Synset-level relations.
    newsynsets ← newsynsets ∪ [s] ∪ AlsoSees(s) ∪ SimilarTos(s)
    for lemma ∈ Lemmas(s)        ▷ Lemma-level relations.
        for altLemma ∈ (DerivationallyRelatedForms(lemma) ∪ Pertainyms(lemma))
            newsynsets ← newsynsets ∪ [Synset(altLemma)]
return newsynsets

OtherPolarity(synsets)
newsynsets ← [ ]
for s ∈ synsets
for lemma ∈ Lemmas(s)        ▷ Lemma-level relations.
    for altLemma ∈ Antonyms(lemma)
        newsynsets ← newsynsets ∪ [Synset(altLemma)]
return newsynsets
```

The algorithm has a number of free parameters: the seed-sets, the WordNet relations called in SamePolarity and OtherPolarity, the number of iterations, the decision to remove overlap. The demo allows you to try out different combinations of values:

Demo Simple WordNet propagation:
http://sentiment.christopherpotts.net/wnpropagate/

Table 6 provides some additional seed-sets, drawing from other distinctions found in the Harvard Inquirer. These can be pasted into the demo if one wants a sense for how well new lexical classes propagate.

Table 6 Propagation example seed-sets to try.

Category	Seed set
Pleasur	amuse, calm, ecstasy, enjoy, joy
Pain	agony, disconcerted, fearful, regret, remorse
Strong	illustrious, rich, control, perseverance

Weak	lowly, poor, sorry, sluggish, weak
MALE	boy, brother, gentleman, male, guy
Female	girl, sister, bride, female, lady

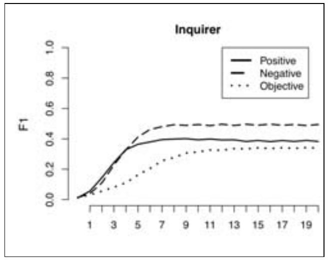
To assess the algorithm for polarity sense-preservation, I began with the seed-sets in **table 7** and then allowed the propagation algorithm to run for 20 iterations, checking each for its effectiveness at reproducing the Positiv/Negativ/Neither distinctions in the subset of Harvard General Inquirer that is also in WordNet.

Table 7 Seed sets used to evaluate the WordNet propagation algorithm against the Harvard General Inquirer.

Positive	excellent, good, nice, positive, fortunate, correct, superior
Negative	nasty, bad, poor, negative, unfortunate, wrong, inferior
Objective	administrative, financial, geographic, constitute, analogy, ponder, material, public, department, measurement, visual

Figure 2 summarizes the results of this experiment, which are decidedly mixed.

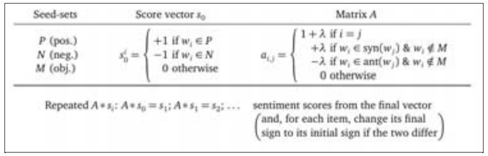
Figure 2 Assessing how well simple Wordnet propagation is able to recover the Harvard Inquirer Positiv/Negativ/Neither classes using the seed sets of **table 7**.



3.2 Weighted WordNet propagation

Blair-Goldensohn, Hannan, McDonald, Ryan, Reis, and Reynar (2008) developed an algorithm that propagates not only the senses of the original seed set but also attaches scores to words, reflecting their intensity, which here is given by the strength of their graphical connections to the seed words. The algorithm is stated in **figure 3**.

Figure 3 The WordNet score propagation algorithm.

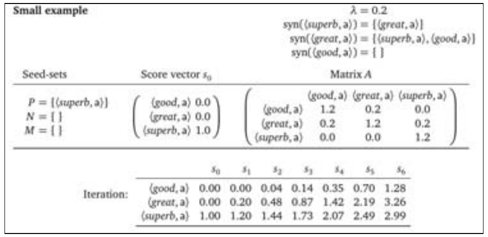


Seed-sets	Score vector s_0	Matrix A
P (pos.) N (neg.) M (obj.)	$s_0^i = \begin{cases} +1 \text{ if } w_i \in P \\ -1 \text{ if } w_i \in N \\ 0 \text{ otherwise} \end{cases}$	$a_{i,j} = \begin{cases} 1+\lambda \text{ if } i=j \\ +\lambda \text{ if } w_i \in \text{syn}(w_j) \;\&\; w_i \notin M \\ -\lambda \text{ if } w_i \in \text{ant}(w_j) \;\&\; w_i \notin M \\ 0 \text{ otherwise} \end{cases}$

Repeated $A * s_i$: $A * s_0 = s_1; A * s_1 = s_2; \ldots$ sentiment scores from the final vector $\left(\text{and, for each item, change its final sign to its initial sign if the two differ}\right)$

Figure 4 works through an example.

Figure 4 WordNet score propagation example. The authors propose a further rescaling of the scores: `log(abs(s)) * sign(s) if abs(s) > 1, else 0`. However, in the example, we would lose the sentiment score for `good` if we stopped before iteration 6. In my experiments, rescaling resulted in dramatically fewer non-0 values.



Small example

$$\lambda = 0.2$$
$$\text{syn}(\langle superb, \text{a}\rangle) = [\langle great, \text{a}\rangle]$$
$$\text{syn}(\langle great, \text{a}\rangle) = [\langle superb, \text{a}\rangle, \langle good, \text{a}\rangle]$$
$$\text{syn}(\langle good, \text{a}\rangle) = [\ ]$$

Seed-sets	Score vector s_0	Matrix A
$P = [\langle superb, \text{a}\rangle]$ $N = [\ ]$ $M = [\ ]$	$\begin{pmatrix} \langle good, \text{a}\rangle & 0.0 \\ \langle great, \text{a}\rangle & 0.0 \\ \langle superb, \text{a}\rangle & 1.0 \end{pmatrix}$	$\begin{pmatrix} & \langle good, \text{a}\rangle & \langle great, \text{a}\rangle & \langle superb, \text{a}\rangle \\ \langle good, \text{a}\rangle & 1.2 & 0.2 & 0.0 \\ \langle great, \text{a}\rangle & 0.2 & 1.2 & 0.2 \\ \langle superb, \text{a}\rangle & 0.0 & 0.0 & 1.2 \end{pmatrix}$

Iteration:		s_0	s_1	s_2	s_3	s_4	s_5	s_6
	⟨good, a⟩	0.00	0.00	0.04	0.14	0.35	0.70	1.28
	⟨great, a⟩	0.00	0.20	0.48	0.87	1.42	2.19	3.26
	⟨superb, a⟩	1.00	1.20	1.44	1.73	2.07	2.49	2.99

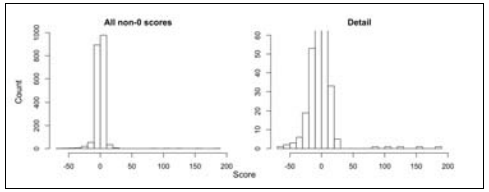
I ran the algorithm using the full Harvard General Inquirer Positiv/Negative/Neither classes as seeds-sets. The output in archived CSV format:

- **WordNet scores lexicon** (zip file)
- **Associated readme file** (txt file)

You can view the results at **the lexicon demo**.

In my informal assessment, the positive and negative scores it assigns tend to be accurate. The disappointment is that so many of the scores are 0, as see in **figure 5**. I think this could be addressed by following more relations that just the basic synset one, as we do for the simple WordNet propagation algorithm, but I've not tried it yet.

Figure 5 WordNet score propagation score distribution.



3.3 Review word scores

In this section, I make use of the CSV-formatted data here:

- **http://compprag.christopherpotts.net/code-data/imdb-words.csv.zip**

This is a tightly controlled, POS-tagged dataset. Even more carefully curated ones are here, drawing from a wider range of corpora:

- **http://www.stanford.edu/~cgpotts/data/wordnetscales/**

And for more naturalistic, non-POS-tagged data in this format from a variety of sources:

- **http://www.stanford.edu/~cgpotts/data/salt20/potts-salt20-data-and-code.zip**

The methods are discussed and motivated in **Constant, Davis, Potts, and Schwarz 2008** and **Potts and Schwarz 2010**, and **this page provides a more extended discussion with associated R code**.

3.3.1 Data

The file `http://compprag.christopherpotts.net/code-data/imdb-words.csv.zip` consists of data gathered from the user-supplied reviews at the **IMDB**. I suggest that you take a moment right now to browse around the site a bit to get a feel for the nature of the reviews — their style, tone, and

so forth.

The focus of this section is the relationship between the review authors' language and the star ratings they choose to assign, from the range 1-10 stars (with the exception of **This is Spinal Tap**, which goes to 11). Intuitively, the idea is that the author's chosen star rating affects, and is affected by, the text she produces. The star rating is a particular kind of high-level summary of the evaluative aspects of the review text, and thus we can use that high-level summary to get a grip on what's happening linguistically.

The data I'll be working with are all in the format described in **table 8**. Each row represents a star-rating category. Thus, for example, in these data, `(bad, a)` is used 122,232 in 1-star reviews, and the total token count for 1-star reviews is 25,395,214.

Table 8 The data format. Some of the files linked above do not have the Tag column, and most of them are based in 5 stars rather than 10 stars.

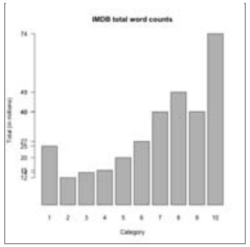
Word	Tag	Category	Count	Total
bad	a	1	122232	25395214
bad	a	2	40491	11755132
bad	a	3	37787	13995838
bad	a	4	33070	14963866
bad	a	5	39205	20390515
bad	a	6	43101	27420036
bad	a	7	46696	40192077
bad	a	8	42228	48723444
bad	a	9	29588	40277743
bad	a	10	51778	73948447

The next few sections describe methods for deriving sentiment lexicons from such data. The methods should generalize to other kinds of ordered sentiment metadata (e.g., helpfulness ratings, confidence ratings).

3.3.2 Category sizes

A common feature of online user-supplied reviews is that the positive reviews vastly out-number the negative ones; see **figure 6**.

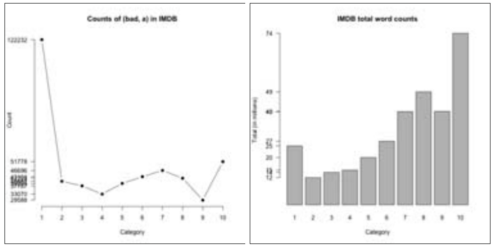
Figure 6 The highly imbalanced category sizes.



3.3.3 Word distributions: Raw counts are misleading

As we saw above, the raw Count values are likely to be misleading due to the very large size imbalances among the categories. For example, there are more tokens of `(bad, a)` in 10-star reviews than in 2-star ones, which seems highly counter-intuitive. Plotting the values reveals that the Count distribution is very heavily influenced by the overall distribution of words (**figure 7**).

Figure 7 Count distribution for `(bad, a)` (left) and the overall category size (right; repeated from **figure 6**). The distribution is heavily influenced by the category sizes.



The source of this odd picture is clear: the 10-star category is 7 times bigger than the 1-star category, so the absolute counts do not necessarily reflect

the rate of usage.

3.3.4 Word distributions: Relative frequencies

To get a better read on the usage patterns, we use relative frequencies:

Definition: Relative Frequencies (RelFreq)

Count/Total

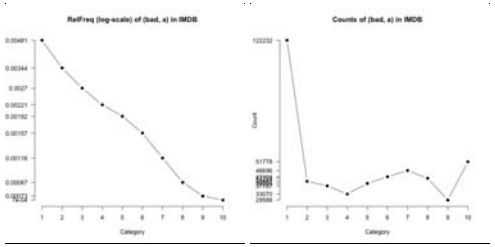
Table 9 extends **table 8** with these RelFreq values.

Table 9 The data extended with relative frequencies (RelFreq) values (= Count / Total).

Word	Tag	Category	Count	Total	RelFreq
bad	a	1	122232	25395214	0.0048
bad	a	2	40491	11755132	0.0034
bad	a	3	37787	13995838	0.0027
bad	a	4	33070	14963866	0.0022
bad	a	5	39205	20390515	0.0019
bad	a	6	43101	27420036	0.0016
bad	a	7	46696	40192077	0.0012
bad	a	8	42228	48723444	0.0009
bad	a	9	29588	40277743	0.0007
bad	a	10	51778	73948447	0.0007

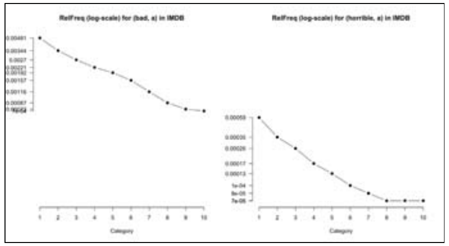
Relative frequency values are hard to get a grip on intuitively because they are so small. Plotting helps bring out the relationships between the values, as in **figure 8**.

Figure 8 RelFreq distribution for `(bad, a)` (left), alongside the Count distribution (right; repeated from **figure 7**). RelFreq values show little or no influence from the underlying category sizes.



One drawback to RelFreq values is that they are highly sensitive to overall frequency. For example, `(bad, a)` is significantly more frequent than `(horrible, a)`, which means that the RelFreq values for the two words are hard to directly compare. **Figure 9** nonetheless attempts a comparison.

Figure 9 Comparing words via their RelFreq distributions.



It is possible to discern that `(bad, a)` is less extreme in its negativity than `(horrible, a)`. However, the effect looks subtle. The next measure we look at abstracts away from overall frequency, which facilitates this kind of direct comparison.

3.3.5 Word distributions: Probabilities

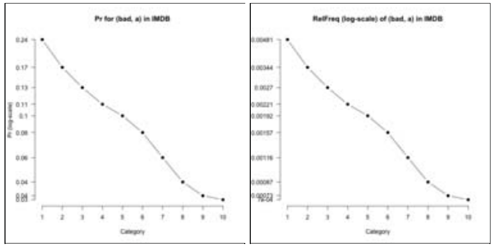
A drawback to RelFreq values, at least for present purposes, is that they are extremely sensitive to the overall frequency of the word in question. There is a comparable value that is insensitive to this quantity:

Definition: Pr values

RelFreq / sum(RelFreq)

Pr values are just rescaled RelFreq values: we divide by a constant to get from RelFreq to Pr. As a result, the distributions have exactly the same shape, as we see in **figure 10**.

Figure 10 Comparing Pr values (left) with RelFreq values (right; repeated from **figure 8**). The shapes are exactly the same (Pr is a rescaling of RelFreq).



A technical note: The move from RelFreq to Pr involves an application of Bayes Rule.

1. RelFreq Values can be thought of as estimates of the conditional distribution `P(word|rating)`: given that I am in rating category `rating`, how likely am I to produce `word`?
2. Bayes Rule allows us to obtain the inverse distribution `P(rating|word)`:

   `P(rating|word) = P(word|rating)P(rating) / P(word)`

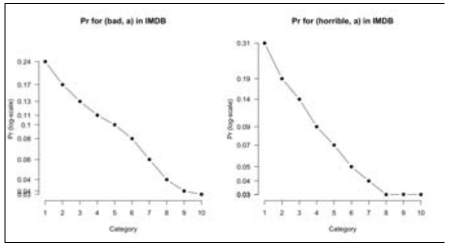
3. However, we would not want to directly apply this rule, because of the term `P(rating)` in the numerator. That would naturally be approximated by the distribution given by Total, as in **figure 6**, which would simply re-introduce all of those unwanted biases.
4. Thus, we keep `P(rating)` constant, which is just to say that we leave it out:

   `P(word|rating) / P(word)`

   where `P(word) = sum(RelFreq)`.

Pr values greatly facilitate comparisons between words (figure 11).

Figure 11 Comparing the Pr distributions of `(bad, a)` and `(horrible, a)`. The comparison is easier than it was with RelFreq values (figure 8).



I think these plots clearly convey that `(bad, a)` is less intensely negative than `(horrible, a)`. For example, whereas `(bad, a)` is at least used throughout the scale, even at the top, `(horrible, a)` is effectively never used at the top of the scale.

(For methods that rigorously compare word distributions of this sort, see this write-up, this talk, and Davis 2011.)

3.3.6 Scoring with expected ratings

We are now in a position to assign polarity scores to words. A first method for doing this uses expected ratings:

Definition: Expected ratings

sum((Category-5.5) * Pr)

Subtracting 5.5 from the Category values centers them at 0, so that we can treat scores below 0 as negative and scores above 0 as positive.

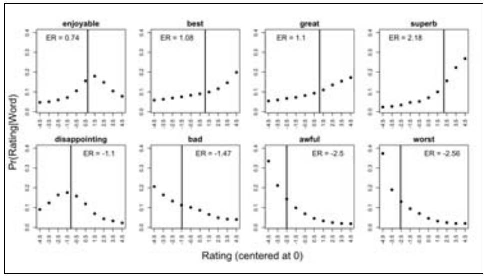
Expected ratings calculations are used by de Marneffe et al. 2010 to summarize Pr-based distributions. The expected rating calculation is just a weighted average of Pr values.

To get a feel for these values, it helps to work through some examples:

1. The rating vector is R = [-4.5 .. 4.5]

2. If the Pr is P = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1] (all 10-star), then sum(R * P) = 4.5
3. If the rating vector is R = [0, 0, 0.2, 0, 0, 0, 0, 0, 0, 0.8] (all 10-star), then we do sum(R * P) = 3.1
4. If the rating vector is R = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0] (all 10-star), then we do sum(R * P) = -4.5
5. If the rating vector is R = [0.35, 0.26, 0.1, 0.05, 0.05, 0.05, 0.02, 0.02, 0.05, 0.05] (all 10-star), then we do sum(R * P) = -2.33

Figure 12 Pr plots with added expected rating.



To get sentiment classification and intensity, we treat words with ER values below 0 as negative, those with ER valus above 0 as positive, and then use the absolute values as measures of intensity:

Definition: Sentiment lexicon via ER values.

A word *w* is positive if ER(*w*) ≥ 0, else negative.

A word *w*'s intensity is abs(ER(*w*)).

3.3.7 Scoring with logistic regression

Expected ratings are easy to calculate and quite intuitive, but it is hard to know how confident we can be in them, because they are insensitive to the amount and kind of data that went into them. Suppose the ER for words `v` and `w` are both 10, but we have 500 tokens of `v` and just 10 tokens of `w`. This suggests that we can have a high degree of confidence in our ER for `v`, but not for `w`. However, ER values don't encode this uncertainty, nor is there an obvious way to capture it.

Logistic regression provides a useful way to do the work of ERs but with the added benefits of having a model and associated test statistics and measures of confidence. For our purposes, we can stick to a simple model that uses Category values to predict word usage. The intuition here is just the one that we have been working with so far: the star-ratings are correlated with the usage of some words. For a word like `(bad, a)`, the correlation is negative: usage drops as the ratings get higher. For a word like `(amazing, a)`, the correlation is positive.

With our logistic regression models, we will essentially fit lines through our RelFreq data points, just as one would with a linear regression involving one predictor. However, the logistic regression model fits these values in log-odds space and uses the inverse logit function (plogis in R) to ensure that all the predicted values lie in [0,1], i.e., that they are all true probability values. Unfortunately, there is not enough time to go into much more detail about the nature of this kind of modeling. I refer to **Gelman and Hill 2008, §5-6** for an accessible, empirically-driven overview. Instead, let's simply fit a model and try to build up intuitions about what it does and says.

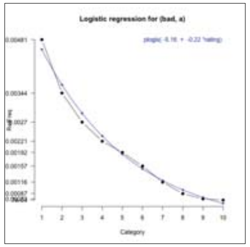
The simple linear regression model for `bad` is given in **table 10**. The model simply uses the rating values to predict the usage (log-odds) of the word in each category.

Table 10 Logistic regression fit for `(bad, a)`.

	Coefficient Estimate	Standard Error	t value	p
Intercept	-5.16	0.046	-112.49	< 0.00001
Category	-0.22	0.008	-27.96	< 0.00001

This model is plotted on **figure 13**.

Figure 13 RelFreq view of `(bad, a)` with logistic regression.

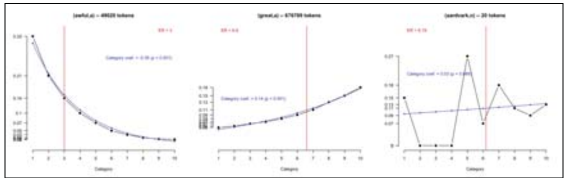


Here, we can use the coefficient for Category as our sentiment score. Where the value is negative (negative slope), the word is negative. Where it is

positive, the word is positive. Informally, we can also use the size of the coefficient as a measure of its intensity.

The great strength of this approach is that we can use the p-values to determine whether a score is trustworthy. **Figure 14** helps to convey why this is an important new power. (Here and in later plots, I've rescaled the values into Pr space to facilitate comparisons.)

Figure 14 Comparing words using our assessment values.



This leads to the following method for inducing a sentiment lexicon from these data:

Definition: Sentiment lexicon via logistic regression

Let Coef(*w*) be the Category coefficient for if that coefficient is significant at the chosen level, else 0

If Coef(*w*) = 0, then *w* is objective/neutral

If Coef(*w*) > 0, then *w* is positive

If Coef(*w*) < 0, then *w* is negative

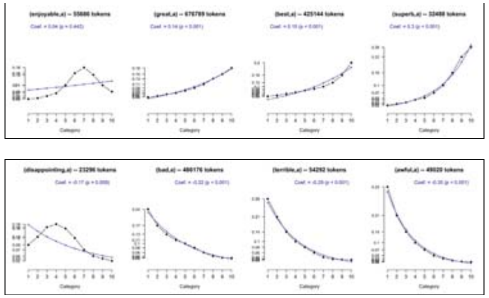
A word's intensity is abs(Coef(*w*))

Depending on where the significance value is set, this can learn conservative lexicons of a few thousand words or very liberal lexicons of tens of thousands.

This method of comparing coefficient values is likely to irk statisticians, but it works well in practice. For a more exact and careful method, as well as a proposal for how to compare words with non-linear relationships to the ratings, see **this talk I gave recently on creating lexical scales**.

Figure 15 shows off this new method of lexicon induction.

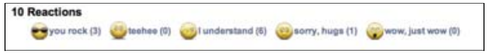
Figure 15 Some scalars in the IMDB.



3.4 Experience Project reaction distributions

The **Experience Project** is a social networking website that allows users to share stories about their own personal experiences. At the **confessions** portion of the site, users write typically very emotional stories about themselves, and readers can then chose from among five reaction categories to the story, but clicking on one of the five icons in **figure 16**. The categories provide rich new dimensions of sentiment, ones that are generally orthogonal to the positive/negative one that most people study but that nonetheless models important aspects of sentiment expression and social interaction (**Potts 2010b**, **Socher, Pennington, Huang, Ng and Manning 2011**).

Figure 16 **Experience Project** categories. "You rock" is a positive exclamative category. "Teehee" is a playful, lighthearted category. "I understand" is an expression of solidarity. "Sorry, hugs" is a sympathetic category. And "Wow, just wow" is negative exclamative, the least used category on the site.



This section presents a simple method for using these data to develop sentiment lexicons.

3.4.1 Data

As with the IMDB data above, I've put the word-level information into an easy-to-use CSV format, as in **table 11**. Thus, as long as you require only word-level statistics, you needn't scrape the site again.

- Use the file epconfessions-unigrams.csv in **http://www.stanford.edu/~cgpotts/data/salt20/potts-salt20-data-and-code.zip**

Table 11 Experience Project word-level data.

Word	Category	Count	Total
bad	hugs	11612	18038374
bad	rock	5711	14066087
bad	teehee	3987	8167037
bad	understand	12577	20466744
bad	wow	5993	12550603

3.4.2 Word distributions: Observed and expected counts

The basic scoring method contrasts observed click rates with expected click rates on the assumption that all word–click combinations are equally likely:

Definition: Observed/Expected values

Expected: sum(Count) / (Total/sum(Total))

The O/E values for w are Count/Expected

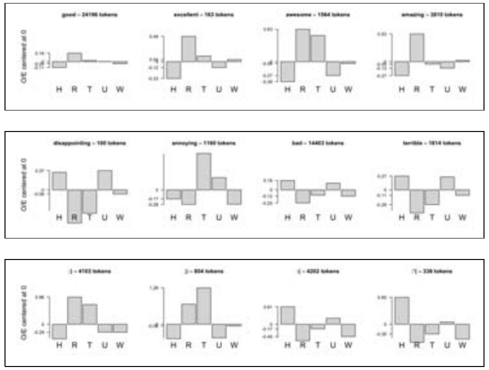
Table 12 extends **table 11** with Expected and O/E values.

Table 12 Experience Project word-level data.

Word	Category	Count	Total	Expected	O/E
bad	hugs	11612	18038374	9816.527	1.1829030
bad	rock	5711	14066087	7652.981	0.7462452
bad	teehee	3987	8167037	4443.831	0.8971989
bad	understand	12577	20466744	11137.727	1.1292250
bad	wow	5993	12550603	6828.934	0.8775893

Some representative cases:

Figure 17 O/E values for representative words.



These scores give rise to a multidimensional lexical entry via the following definition:

Definition: Multidimensional lexicon

EP(*w*) is a five dimensional vector of O/E values.

The Chi-squared test or the G-test (log-likelihood test) can be used to reduce these vectors to all-0 based on significance testing.

The lexicon demos include both IMDB and EP scores as well:

Demo Explore the sentiment lexicons discussed here: **http://sentiment.christopherpotts.net/lexicon/**

Demo Use the sentiment lexicons to score entire texts: **http://sentiment.christopherpotts.net/textscores/**

4 Summary of conclusions

1. There are a number of good fixed lexicons for sentiment. They are negligible to high levels of disagreement with each other. These can be exploited strategically — resolve the conflicts somehow or allow them to persist as genuine points of uncertainty.
2. WordNet can be used to derive interesting lexicons from small seeds sets, even for distinctions that are not directly encoded in WordNet's structure.
3. Naturally occurring metadata are a rich source of lexical entries. Statistical models are valuable for such lexicon induction.
4. A major advantage of inducing a lexicon directly from data is that one can then capture domain specific effects, which are very common in sentiment. (See also the discussion of vector-space models for lexicon induction methods that don't any metadata.)

Home |

Sentiment Symposium Tutorial: Classifiers

1. **Overview**
2. **Models**
   - 2.1 **Naive Bayes**
   - 2.2 **Maximum Entropy**
   - 2.3 **Others**
3. **Assessing classifier models**
   - 3.1 **Accuracy and its limitations**
   - 3.2 **Precision**
   - 3.3 **Recall**
   - 3.4 **Derived effectiveness measures**
   - 3.5 **Training data assessment**
   - 3.6 **The importance of out-of-domain testing**
4. **Comparisons: Naive Bayes vs. MaxEnt**
5. **Feature selection**
   - 5.1 **A priori method cut-offs**
   - 5.2 **Lexicon induction as feature set induction**
   - 5.3 **Mutual information**
   - 5.4 **A cautionary note**
6. **Back the model competition**
7. **Tools**
   - 7.1 **The Stanford Classifier**
   - 7.2 **Others**
8. **A lightweight, accurate classifier**
9. **Summary of conclusions**

1 Overview

This section introduces two classifier models, Naive Bayes and Maximum Entropy, and evaluates them in the context of a variety of sentiment analysis problems. Throughout, I emphasize methods for evaluating classifier models fairly and meaningfully, so that you can get an accurate read on what your systems and others' systems are really capturing.

Demo Trained classifier models to experiment with: **http://sentiment.christopherpotts.net/classify/**

2 Models

I concentrate on two closely related probabilistic models: Naive Bayes and MaxEnt. Some other classifier models are reviewed briefly below as well.

2.1 Naive Bayes

The Naive Bayes classifier is perhaps the simplest trained, probabilistic classifier model. It is remarkably effective in many situations.

I start by giving a recipe for training a Naive Bayes classifier using just the words as features:

1. Estimate the probability *P*(*c*) of each class *c* C by dividing the number of words in documents in *c* by the total number of words in the corpus.
2. Estimate the probability distribution *P*(*w* | *c*) for all words *w* and classes *c*. This can be done by dividing the number of tokens of *w* in documents in *c* by the total number of words in *c*.
3. To score a document *d* for class *c*, calculate

$$\textbf{score}(d,c) \overset{def}{=} P(c) * \prod_{i=1}^{n} P(w_i|c)$$

4. If you simply want to predict the most likely class label, then you can just pick the c with the highest **score** value. To get a probability distribution, calculate

$$P(c|d) \overset{def}{=} \frac{\textbf{score}(d,c)}{\sum_{c' \in C} \textbf{score}(d,c')}$$

The last step is important but often overlooked. The model predicts a full distribution over classes. Where the task is to predict a single label, one chooses the label with the highest probability. It should be recognized, though, that this means losing a lot of structure. For example, where the max label only narrowly beats the runner-up, we might want to know that.

The chief drawback to the Naive Bayes model is that it assumes each feature to be independent of all other features. This is the "naive" assumption seen in the multiplication of $P(w_i \mid c)$ in the definition of **score**. Thus, for example, if you had a feature `best` and another `world's best`, then their probabilities would be multiplied as though independent, even though the two are overlapping. The same issues arise for words that are highly correlated with other words (idioms, common titles, etc.).

2.2 Maximum Entropy

The Maximum Entropy (MaxEnt) classifier is closely related to a Naive Bayes classifier, except that, rather than allowing each feature to have its say independently, the model uses search-based optimization to find weights for the features that maximize the likelihood of the training data.

The features you define for a Naive Bayes classifier are easily ported to a MaxEnt setting, but the MaxEnt model can also handle mixtures of boolean, integer, and real-valued features.

I now briefly sketch a general recipe for building a MaxEnt classifier, assuming that the only features are word-level features:

1. For each word *w* and class *c* *C*, define a joint feature *f*(*w*, *c*) = *N* where *N* is the number of times that *w* occurs in a document in class *c*. (*N* could also be boolean, registering presence vs. absence.)
2. Via iterative optimization, assign a weight to each joint feature so as to maximize the log-likelihood of the training data.
3. The probability of class *c* given a document *d* and weights λ is

$$P(c|d,\lambda) \overset{def}{=} \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c' \in C} \exp \sum_i \lambda_i f_i(c',d)}$$

Because of the search procedures involved in step 2, MaxEnt models are more difficult to implement than Naive Bayes model, but this needn't be an obstacle to using them, since there are excellent software packages available.

The features for a MaxEnt model can be correlated. The model will do a good job of distributing the weight between correlated features. (This is not to say, though, that you should be indifferent to correlated features. They can make the model hard to interpret and reduce its portability.)

In general, I think MaxEnt is a better choice than Naive Bayes. Before supporting this with systematic evidence, I first mention a few other classifier models and pause to discuss assessment metrics.

2.3 Others

1. Support Vector Machines (likely to be competitive with MaxEnt; see **Pang, Lee, and Vaithyanathan 2002**).
2. Decision Trees (valuable in situations in which you can intuitively define a sequence of interdependent choices, though I've not seen them used for sentiment).
3. Generalized Expectation Criteria (a generalization of MaxEnt that facilitates bringing in expert labels; see **Mann and McCallum 2010**).
4. AdaBoost (**Wilson, Wiebe, and Hoffmann (2005)** use AdaBoost in the context of polarity lexicon construction).

3 Assessing classifier models

This section reviews classifier assessment techniques. The basis for the discussion is the confusion matrix. An illustrative example is given in **table 1**.

Table 1 Confusion matrix.

		Predicted		
		Pos	**Neg**	**Obj**
	Pos	15	10	100
Observed	**Neg**	10	15	10
	Obj	10	100	1000

3.1 Accuracy and its limitations

The most intuitive and widely used assessment method is accuracy: correct guesses divided by all guesses. That is, divide the boxed (diagonal) cells in **table 2** by the sum of all cells.

How to cheat: if the categories are highly imbalanced one can get high accuracy by always or often guessing the largest category. Most real world tasks involving highly imbalanced category sizes, so accuracy is mostly useless on its own. One should at least always ask what the accuracy would be of a classifier that guessed randomly based on the class distribution.

Table 2 Accuracy: divide the diagonal cells by the sum of all cells.

		Predicted		
		Pos	**Neg**	**Obj**

Observed	Pos	15	10	100
	Neg	10	15	10
	Obj	10	100	1000

3.2 Precision

Precision is the correct guesses penalized by the number of incorrect guesses. With the current confusion matrix set-up, the calculations are column-based (**table 3**).

How to cheat: You can often get high precision for a category C by rarely guessing C, but this will ruin your recall.

Table 3 Precision: true positive / (true positive + false positive).

		Predicted		
		Pos	Neg	Obj
Observed	Pos	15	10	100
	Neg	10	15	10
	Obj	10	100	1000

3.3 Recall

Recall is correct guesses penalized by the num ber of missed items. With the current confusion matrix set-up, the calculations are row-based (**table 4**).

How to cheat: You can often get high recall for a category C by always guessing C, but this will ruin your precision.

Table 4 Recall: true positive / (true positive + false negative).

		Predicted		
		Pos	Neg	Obj
Observed	Pos	15	10	100
	Neg	10	15	10
	Obj	10	100	1000

3.4 Derived effectiveness measures

Precision and recall values can be combined in various ways:

1. F1: 2((precision*recall) / (precision+recall))
2. Macro-average: Average precision, recall, or F1 over the classes of interest.
3. Micro-average: Sum corresponding cells to create a 2 x 2 confusion matrix, and calculate precision in terms of the new matrix. (In this set-up, precision, recall, and F1 are all the same.)

Both F1 and micro-averaging assume that precision and recall are equally weighted. This is often untrue of real-world situations. If we are deathly afraid of missing something, we favor recall. If we require pristine data, we favor precision.

Similarly, both of the averaging procedures assume we care equally about all categories, and they also implicitly assume that no category is an outlier in one direction or another.

3.5 Training data assessment

Now that we have some effectiveness measures, we want to do some experiments. The experiments themselves are likely not of interest, though (unless we are in a competition). Rather, we want to know how well our model is going to generalize to unseen data.

The standard thing to do is train on one portion of the corpus and then evaluate on a disjoint subset of that corpus. We've done this a number of times already.

Here, I'd like to push for the idea that you should also test how well your model does *on the very data it was trained on*. Of course, it should do well. However, it shouldn't do too much better than it does on the testing data, else it is probably over-fitting. This is a pressing concern for MaxEnt models, which have the power to fully memorize even large training sets if they have enough features.

So, in sum, what we're looking for is harmony between train-set performance and test-set performance. If they are both 100% and 98%, respectively, then we can be happy. If they are 100% and 65%, respectively, then we should be concerned.

3.6 The importance of out-of-domain testing

Wherever possible, you should assess your model on out-of-doman data, which we've also already done here and there. Sometimes, such data is not available. In that case, you might consider annotating 200 or so examples yourself. It probably won't take long, and it will be really illuminating.

4 Comparisons: Naive Bayes vs. MaxEnt

Now that we have assessment ideas under our belts, let's compare Naive Bayes and MaxEnt systematically.

I work with balanced sets throughout, which means that we can safely use accuracy (which in turn greatly simplifies the assessments in general)

On in-domain testing, the MaxEnt model is substantially better than the Naive Bayes model for all of the tokenization schemes we explored earlier. **Figure 1** and **Figure 2** support this claim. The first uses OpenTable data, and the second uses a subset of the Experience Project data for which there is a majority label, which is then treated as the true label for the text.

Figure 1 MaxEnt and Naive Bayes using the tokenization algorithms discussed earlier.

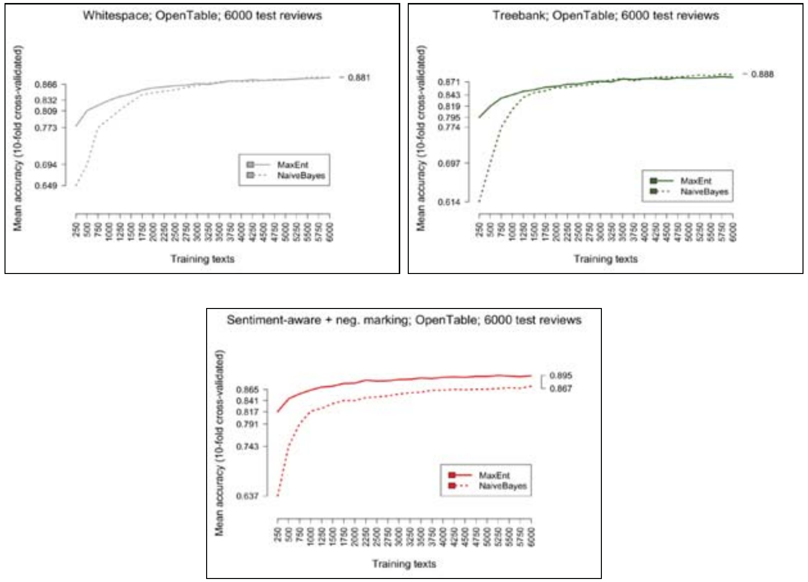
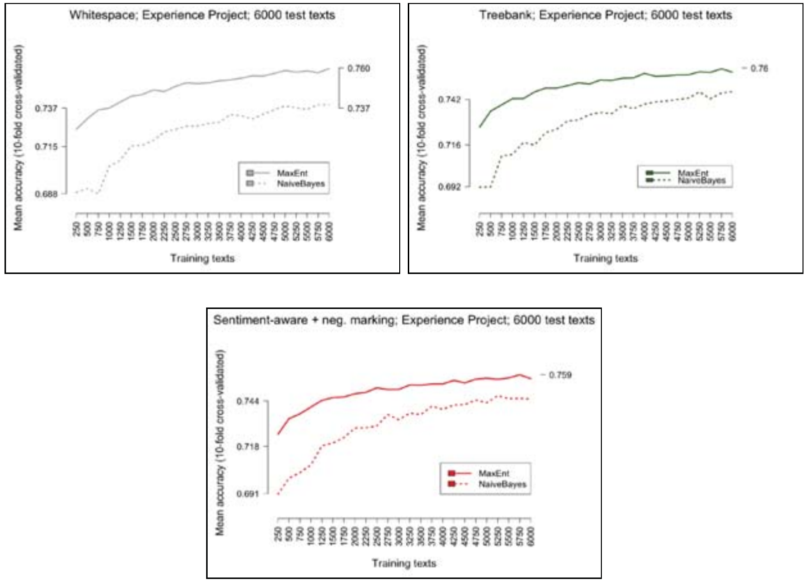


Figure 2 MaxEnt and Naive Bayes using the tokenization algorithms discussed earlier, training and testing Experience Project confessions.



However, he picture is less clear when we move to experiments involving out-of-domain testing. **Figure 3** and **figure 4** report on such comparisons. In both cases, we train on OpenTable data. **Figure 3** tests the model on IMDB data, and **figure 4** tests it on Amazon reviews. In both cases, though MaxEnt does really well with small amounts of data, it is quickly outdone by the Naive Bayes model.

Figure 3 MaxEnt and Naive Bayes using the tokenization algorithms discussed earlier, training and testing on different domains.

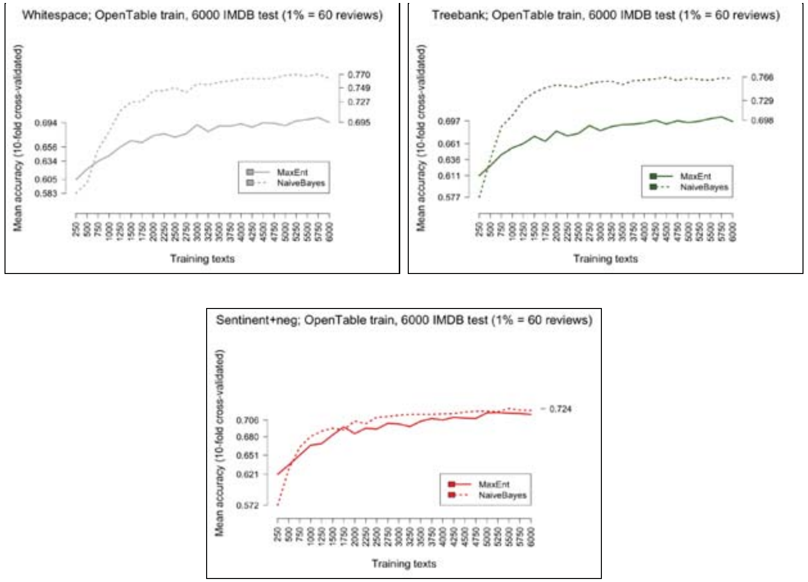
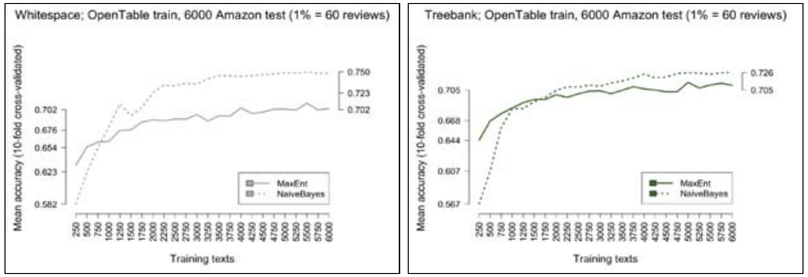
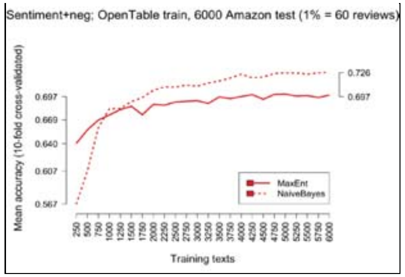


Figure 4 MaxEnt and Naive Bayes using the tokenization algorithms discussed earlier, training and testing on different domains.



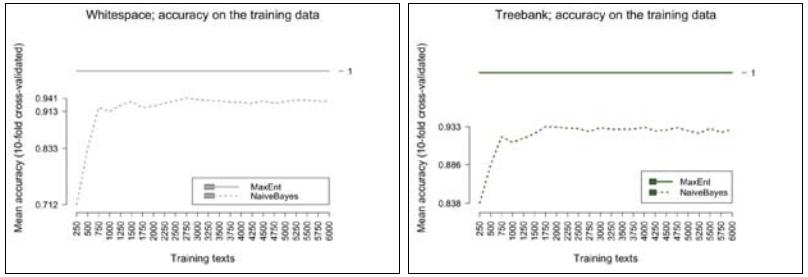


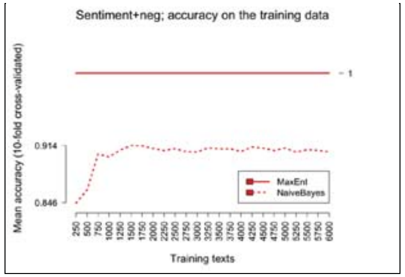
Why is this so? Does it indicate that the Naive Bayes model is actually superior? After all, we typically do not want to make predictions about data that we already have labels for, but rather we want to project those labels onto new data, so we should favor the best out-of-domain model.

Before we make this conclusion, I think we need to do a bit more investigation.

The first step is to see how the models perform on their own training data. **Figure 5** presents these experiments. Strinkingly, the MaxEnt model invariably gets 100% of the training data right. Since we are nowhere near that number even on in-domain test sets, it is likely that we are massively overfitting to the training data. This suggests that we should look to sparser feature sets, so that the model generalizes better.

Figure 5 MaxEnt and Naive Bayes using the tokenization algorithms discussed earlier, testing on the training data.





5 Feature selection

5.1 A priori method cut-offs

Probably the most common method for reducing the size of the feature space is to pick the top *N*(%) of the features as sorted by overall frequency. The top *J* features might be removed as well, on the assumption that they are all stopwords that say little about a document's class.

The major drawback to this method is that is can can lose a lot of important sentiment information. Below the *N* cut-off, there might be infrequent but valuable words like `lackluster` and `hesitant`. In the top *J* are likely to be negation, demonstratives, and personal pronouns that carry subtle but extremely reliable sentiment information (Constant et al. 2008, Potts, Chung and Pennebaker 2007).

5.2 Lexicon induction as feature set induction

Where there is metadata, methods like those we employed in the lexicon section for review data and socia networking data can be employed. This basically means restricting attention to the features that have an interesting relationship to the metadata in question. This can be extremely effective in practice. The drawback I see is that you might be limiting the classifier's ability to generalize to out-of-domain data, where the associations might be different. It is possible to address this using the unsupervised methods discussed in the vector-space section, which can enable you to expand your initial lexicon to include more general associations.

5.3 Mutual information

Another way to leverage one's labels to good effect is to pick the top *N* features as ranked not

by frequency but rather by mutual information with the class labels (**McCallum and Nigam 1998**):

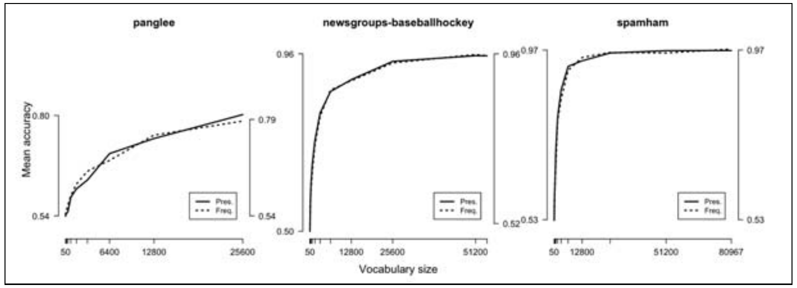
Definition: Mutual information between features and class labels

$$\begin{aligned} I(C;\pi) &= H(C) - H(C|\pi) \\ &= \sum_{c \in C} \sum_{x \in \{0,1\}} P(c, f_\pi(x)) \log_2 \left(\frac{P(c, f_\pi(x))}{P(c)P(f_\pi(x))} \right) \end{aligned}$$

5.4 A cautionary note

Pang and Lee present evidence that sentiment classifier accuracy increases steadily with the size of the vocabulary. We actually already saw a glimpse of this in the Language and Cognition section; the relevant figure is repeated as **figure 6**. However, the ever-better models we get are likely to have trouble generalizing to new data. Thus, despite the risks, some kind of reduction in the feature space is likely to pay off.

Figure 6 A single classifier model (MaxEnt) applied to three different domains at various vocabulary sizes, with vocabulary selection done by mutual information with the class labels.

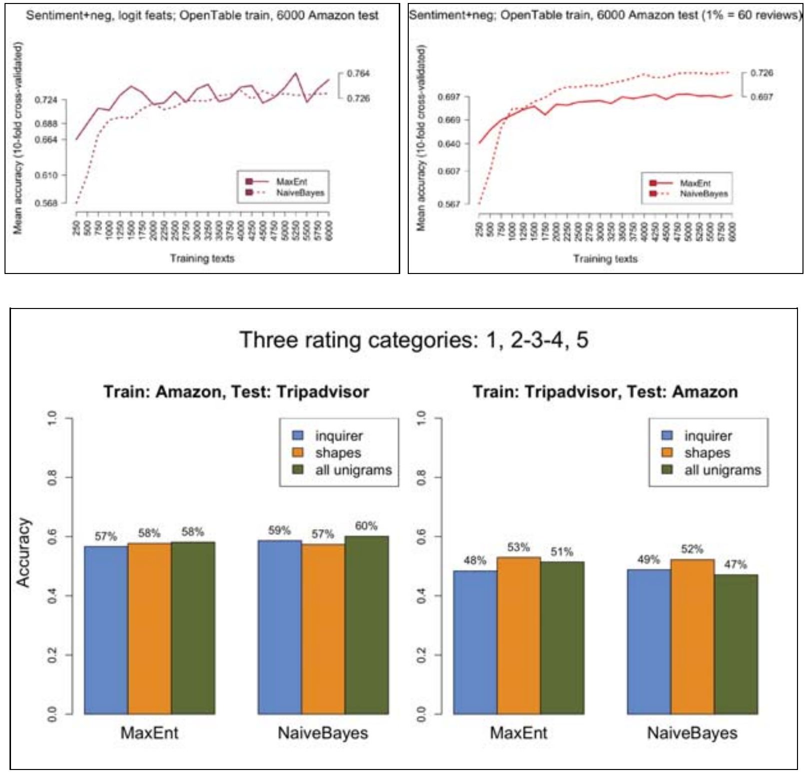


6 Back the model competition

To recap, we've discovered that the MaxEnt model is over-fitting to the training data, which is hurting its out-of-domain performance. The hypothesis is that if we reduce the feature space, the MaxEnt model will pull ahead of the Naive Bayes model even for out-of-domain testing. (Recall that it is already significantly better for in-domain testing.)

Figure 7 reports on experiments along these lines. Both models are largely improved by reduced features space, but MaxEnt really pulls ahead pretty decisively when one looks at the overall performance numbers.

Figure 7 Model comparison for a feature set derived via the lexicon induction method of the IMDB lexicon section. The smaller model addresses the over-fitting problem for MaxEnt, allowing it to pull ahead of Naive Bayes. I am not sure why the MaxEnt performance is so volatile, though. I've included results from earlier, different experiments to try to make the overall picture clearer.



7 Tools

There are excellent software packages available for fitting models like these. This section focusses on the Stanford Classifier but links to others that are also extremely useful.

7.1 The Stanford Classifier

The **Stanford Classifier** is written in Java, and the distribution provides example code for integrating it into your own projects. It is free for academic use, and the **licensing fees for commerical use** are very reasonable.

The classifier also has an excellent command-line interface. One simply creates tab-separated training and testing files. One column (usually the leftmost, or 0th) provides the true labels. The remaining columns specify features, which can be boolean, integer, real-valued, or textual. For textual columns, one can specify how to break it up into string-based boolean features.

The command-line interface is fully documented in the javadoc/ directory of the classifier distribution: open index.html in a Web browser and find ColumnDataClassifier in the lower-left navigation bar.

Figure 8 contains a prop (properties) file for the classifier.

Figure 8 A sample prop file for the Stanford classifier. The strings marked with preceding $ would be filled in by you, based on the files you wanted to feed the classifier and those that you wanted it to produce.

```
# USAGE
# java -mx3000m -jar stanford-classifier.jar -prop $THISFILENAME
#
##### Training and testing source, as tab-separated-values files:
#
trainFile=$TRAINFILE.tsv
testFile=$TESTFILE.tsv
#
# Model (useNB=false is MaxEnt; useNB=true is NaiveBayes)
useNB=false
#
##### Features
#
# Use the class feature to capture any biases coming
# from imbalanced class sizes:
#
useClassFeature=true
#
# The true class labels are given in column 0 (leftmost)
#
```

```
goldAnswerColumn=0
#
# Suppose that column 1 contains something like the frequency
# of sentiment words overall. We tell the classifier to treat
# the values as numeric and to log-transform them:
1.realValued=true
1.logTransform=true
#
# Column 2 contains our textual features, already tokenized.
# This says that column 1 is text in which the tokens are
# separated by |||. use splitWordsRegexp=\\s+ to split on
# whitespace
#
2.useSplitWords=true
2.splitWordsRegexp=\\|\\|\\|
#
# Don't display a column to avoid screen clutter;
# use positive numbers to pick out columns you want displayed:
#
displayedColumn=-1
#
# Output a complete representation of the model to a file:
#
printClassifier=AllWeights
printTo=$CLASSIFIERFILE
#
# These are the recommended optimization parameters for MaxEnt;
# if useNB=true, they are ignored:
#
prior=quadratic
intern=true
sigma=1.0
useQN=true
QNsize=15
tolerance=1e-4
```

7.2 Others

- **MALLET** (Java): flexible command-line interface; fully extendible; implements a wide range of classifier; excellent facilities for cross-validation and out-of-domain testing; also provides many other NLP tools, including topic models and optimization methods.
- **NLTK** (Python): very flexible Naive Bayes and MaxEnt implementations.

- **LingPipe** (Java): lots of different classifier models, some of them highly tuned for specific linguistic applications (including sentiment)

8 A lightweight, accurate classifier

The models discussed above tend to be costly in terms of the disk space, memory, and time they require for both training and prediction.

Here's a proposal for how to build a lightweight, accurate classifier that is quick to train on even hundreds of thousands of instances, and that makes predictions for new texts in well under a second:

1. Begin with a set of *N* fixed sentiment lexicons *L*. For my experiments, I used **the fixed polarity lexicons**, **the IMDB scores**, the **Experience Project O/E vectors**, and the sentiment-rich classes from the **Harvard General Inquirer** and **LIWC**. The total number of predictors was 39, all of them numeric.
2. The feature function tokenizes each text using **the sentiment-aware tokenizer** with **negation marking**. _NEG marked phrases have scores in the IMDB and EP data, and are treated like their unmarked counterparts by the other lexicons.
3. For a given text, the feature function simply sums up all the words' scores for each of the 39 predictors and then normalizes them by the length of the text. Thus, each text is modeled as a vector of 39 numbers.
4. Trained and evaluate MaxEnt models to assess performance.

A summary of the experiments I did is given in **table 5**. Overall, the performance is highly competitive with state-of-the-art classifiers, and the model does almost no-overfitting. (Indeed, we could probably stand to fit more tightly to the training data.)

Table 5 Lightweight numeric classifier results.

Train	Test	Test Accuracy	Train Accuracy
IMDB	IMDB	0.877	0.877
OpenTable	OpenTable	0.88	0.884
Opentable + IMDB	PangLee	0.849	0.873

Demo Experiment with classifiers of this form, trained on a variety of data:

http://sentiment.christopherpotts.net/classify/

9 Summary of conclusions

1. Accuracy numbers are meaningful only to the extent that the test set is balanced. Where it is imbalanced, precision and recall number are better.
2. Watch out for over-fitting. A model that does beautifully in-domain might fall apart out of domain. The problem is especially pressing for MaxEnt, which has the power to memorize large training sets.
3. Favor feature selection functions that are sensitive to what is happening in the training data.
4. On the whole, MaxEnt is an improvement over Naive Bayes: MaxEnt is more flexible in terms of its features, and it deals better with correlated features.

Home |

Sentiment tutorial home   Christopher Potts, Stanford Linguistics

Sentiment Symposium Tutorial: Vector-space models

1. Overview
2. Word–word matrices
3. Other matrix designs
4. Sentiment associates
5. Vector-based sentiment propagation
6. Related approaches
7. Summary of conclusions

1 Overview

This section introduces some basic techniques from the study of vector-space models. In the present setting, the guiding idea behind such models is that important aspects of a word's meanings are latent in its distribution, that is, in its patterns of co-occurrence with other words.

I focus on word–word matrices, rather than the more familiar and widely used word–document matrices of information retrieval and much of computational semantics. The reason for this is simply that, on the data I was experimenting with, this design seemed to be delivering better results with fewer resources.

Turney and Pantel 2010 is a highly readable introduction to vector-space models, covering a wide variety of different design choices and applications. I highly recommend it for anyone aiming to implement and deploy models like this.

Demo Explore word-vector similarity in diverse corpora: http://sentiment.christopherpotts.net/vecsim/

2 Word–word matrices

A word–word for a vocabulary *V* of size *n* is an *n* × *n* matrix in which both the rows and columns represent words in *V*. The value in cell (*i*, *j*) is the number of times that word *i* and word *j* appear together in the same context.

Table 1 depicts a fragment of a word–word matrix derived from 12,000 OpenTable reviews, using the sentiment + negation tokenization scheme motivated earlier. Here, the counts for (*i*, *j*) are the number of times that word *i* and word *j* appear together in the same review. With this matrix design, the rows and columns are the same, though they might be very different with other matrix designs.

Table 1 Fragment of a word × word count matrix derived from 12,000 OpenTable reviews. The rows and columns are the same, and the diagonal gives the token counts for each word.

	a	a_neg	able	able_neg	about	about_neg	above
a	32620	4101	340	110	2066	551	122
a_neg	4101	3286	31	41	312	142	21
able	340	31	163	2	16	4	0
able_neg	110	41	2	63	6	7	2
about	2066	312	16	6	1177	37	9
about_neg	551	142	4	7	37	365	1
above	122	21	0	2	9	1	88

We now want to measure how similar the row vectors are to one another. If we compare the raw count vectors (say, by using **Euclidean distance**), then the dominating factor will be overall corpus frequency, which is unlikely to be sentiment relevant. Cosine similarity addresses this by comparing length normalized vectors:

Definition: Cosine similarity

The cosine similarity between vectors *A* and *B* of length *n* is

$$\mathbf{cosim}(A,B) \overset{def}{=} \frac{\sum_{i=1}^{n}(A_i * B_i)}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}}$$

In **table 2**, I picked two sentiment rich target words, `excellent` and `terrible`, and compared them for cosine similarity against the whole vocabulary. The figure provides the 20 closest words by this similarity measure. Though the lists contain more function words than we might like, there are also rich sentiment words, some of them specific to the domain of restaurants, which suggest that we might use this method to increase our lexicon size in a context-dependent manner.

Table 2 Top 20 neighbors of `excellent` and `terrible` as measured by cosine similarity.

excellent	terrible
excellent	terrible
service	.
food	was
and	bad
attentive	service
.	food
friendly	poor
atmosphere	over
delicious	there
prepared	completely
definitely	disappointed
ambiance	just
comfortable	about
with	eating
the	only
choice	horrible
well	almost
recommended	seemed
experience	even
outstanding	probably

Demo Explore word-vector similarity in diverse corpora: **http://sentiment.christopherpotts.net/vecsim/**

Singular value decomposition (SVD; see also the Latent Semantic Analysis of **Deerwester, Dumais, Furnas, Landauer, and Harshman 1990**) is a dimensionality reduction technique that has been shown to uncover a lot of underlying semantic structure. **Table 3** shows the result of length normalizing the word–word matrix and then applying SVD to the result.

Table 3 Table **table 1** after length normalization followed by singular value decomposition.

	a	a_neg	able	able_neg	about	about_neg	above
a	- 0.00012	- 0.00028	- 0.00109	-0.00139	- 0.00279	0.00115	0.00078
a_neg	- 0.00186	0.00077	0.00846	0.00823	0.00457	0.00099	0.00147
able	0.01790	- 0.01113	0.01530	-0.00409	- 0.01084	-0.01161	- 0.01337
able_neg	0.00560	0.00871	0.00003	0.00596	0.01057	-0.01821	- 0.00289
about	0.02486	0.01737	- 0.02056	-0.01453	0.01422	0.01922	0.05323
about_neg	0.00514	0.00212	0.00252	0.00847	0.01624	0.01034	0.00013
above	0.00663	- 0.02498	0.01634	0.00680	- 0.00898	0.00609	- 0.00667

If we compare the new vectors, the results are somewhat different. To my eye, they look messier from a sentiment perspective. Going forward, I'll stick with the simpler method of just using cosine similarity on the raw count vectors.

Table 4 Top 20 neighbors of `excellent` and `terrible` as measured by length normalization followed by singular value decomposition.

excellent	terrible
excellent	terrible
said	fish
about	friend
some	had_NEG
potatoes	left_NEG
ambiance	so-so
...	or_NEG
bar	dessert
i'm	especially
what	curry
you_NEG	saturday_NEG
little	again_NEG
got	pre
his	expectations
be_NEG	of
enough_NEG	inattentive
are	attentive
later_NEG	when
snapper	patrons
tables	you've

3 Other matrix designs

There are many, many design choices available for vector-space models. We can vary not only the matrix itself but also the kinds of dimensionality reduction we perform (if any) and the similarity measures we employ. I refer to **Turney and Pantel 2010** for a thorough overview. Here, I confine myself to mentioning a few other matrix designs that might be valuable for sentiment analysis:

- Word × document (very common)
- Word × word based on proximity in search engine results
- Word × word based on **dependency edges**
- Word × dependecy-edge
- Modifier × modified
- Twitter-based word × hashtag

4 Sentiment associates

Turney and Littman (2003) generalize the informal procedure used above, to achieve a general lexicon learning method. Their approach:

1. Given a word × context matrix, apply tf-idf weighting and then SVD.
2. Definite two seeds sets (positive vs. negative, or whatever contrast is of interest).
3. Rank words by their closeness to the seed sets, using the Semantic Orientation (SO) function defined just below.

Definition: Semantic orientation

Given opposing seeds sets S_1 and S_2, a matrix *M*, and a vector similarity measure **vecSim** defined for *M*, the semantic orientation of a word *w* is

$$SO(w) \overset{\text{def}}{=} \sum_{w' \in S_1} \mathbf{vecSim}(w, w') - \sum_{w' \in S_2} \mathbf{vecSim}(w, w')$$

Turney and Littman explore a number of different weighting schemes and similarity measures, paying special attention to how well they do on various corpus sizes. They conjecture that their approach can be generalized to a wide variety of sentiment contrasts.

I tried out a simple version of their approach on the OpenTable subset used above. The matrix was an unweighted word–word matrix, and the vector similarity measure was cosine similarity. The seed sets I used (**table 5**) are designed to aim for positive and negative words used to describe eating and good restaurant experiences. The top 20 positive and negative words are given in **table 6**. The results look extremely promising to me.

Table 5 Seed sets

positive words	good, excellent, delicious, tasty, pleasant
negative words	bad, spoiled, awful, rude, gross

Table 6 Top 20 words by the semantic orientation measure.

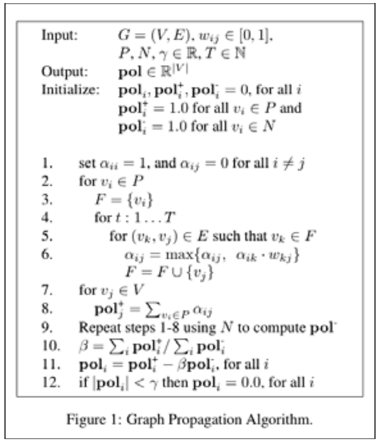
Positive	Negative
excellent	rude
delicious	awful
good	bad
tasty	worst
pleasant	acted
very	apologies_NEG
service	attitude
great	unprofessional
attentive	desserts_NEG
friendly	complained
restaurant	NEVER
all	recommending_NEG
my	customer_NEG
menu	terrible
recommend	HORRIBLE
experience	rude_NEG
nice	questions_NEG
atmosphere	ridiculous
wine	taking_NEG
well	watery

5 Vector-based sentiment propagation

The method of **Velikovich, Blair-Goldensohn, Hannan, and McDonald (2010)** is closely related to that of **Turney and Littman 2003**, except that it extends the idea with a sophisticated and somewhat mind-bending propagation component aimed at learning truly massive sentiment lexicons.

The algorithm is defined in **figure 1**.

Figure 1 Web propagation algorithm. *G* is a cosine similarity graph. *P* and *N* are seed sets. γ is a threshold; sentiment scores below it are rounded to 0. *T* is the number of iterations.



```
Input:       $G = (V, E), w_{ij} \in [0, 1]$,
             $P, N, \gamma \in \mathbb{R}, T \in \mathbb{N}$
Output:      $\mathbf{pol} \in \mathbb{R}^{|V|}$
Initialize:  $\mathbf{pol}_i, \mathbf{pol}_i^+, \mathbf{pol}_i^- = 0$, for all $i$
             $\mathbf{pol}_i^+ = 1.0$ for all $v_i \in P$ and
             $\mathbf{pol}_i^- = 1.0$ for all $v_i \in N$

1.   set $\alpha_{ii} = 1$, and $\alpha_{ij} = 0$ for all $i \neq j$
2.   for $v_i \in P$
3.     $F = \{v_i\}$
4.     for $t : 1 \ldots T$
5.       for $(v_k, v_j) \in E$ such that $v_k \in F$
6.         $\alpha_{ij} = \max\{\alpha_{ij},\ \alpha_{ik} \cdot w_{kj}\}$
           $F = F \cup \{v_j\}$
7.   for $v_j \in V$
8.     $\mathbf{pol}_j^+ = \sum_{v_i \in P} \alpha_{ij}$
9.   Repeat steps 1-8 using $N$ to compute $\mathbf{pol}^-$
10.  $\beta = \sum_i \mathbf{pol}_i^+ / \sum_i \mathbf{pol}_i^-$
11.  $\mathbf{pol}_i = \mathbf{pol}_i^+ - \beta \mathbf{pol}_i^-$, for all $i$
12.  if $|\mathbf{pol}_i| < \gamma$ then $\mathbf{pol}_i = 0.0$, for all $i$
```

Figure 1: Graph Propagation Algorithm.

I use the simple example in **table 7** illustrate how the algorithm works.

Table 7 A simple example to show the action of the propagation algorithm. The corpus at left gives rise to the count matrix (top right) which is then converted into a cosine similarity matrix (bottom right), which is the graph *G* that is the central input to the algorithm.

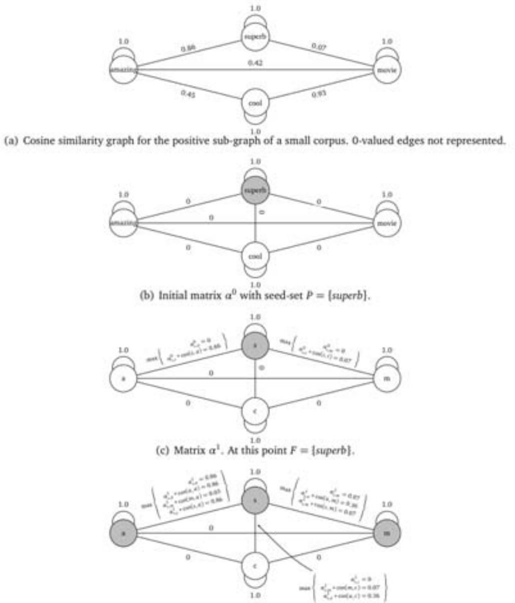
Corpus
superb amazing
superb movie
superb movie
superb movie
superb movie
superb movie
amazing movie
amazing movie
cool superb

	amazing	cool	movie	superb
amazing	0	0	2	1
cool	0	0	0	1
movie	2	0	0	5
superb	1	1	5	0

	amazing	cool	movie	superb
amazing	1.0	0.45	0.42	0.86
cool	0.45	1.0	0.93	0.0
movie	0.42	0.93	1.0	0.07
superb	0.86	0.0	0.07	1.0

Figure 2 continues the above example by showing how propagation works.

Figure 2 The iterative propagation method.



(a) Cosine similarity graph for the positive sub-graph of a small corpus. 0-valued edges not represented.

(b) Initial matrix α^0 with seed-set $P = \{superb\}$.

(c) Matrix α^1. At this point $F = \{superb\}$.

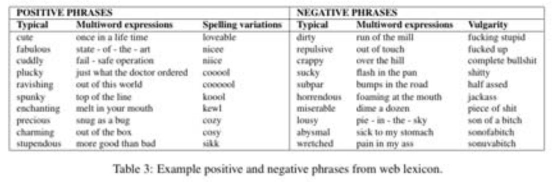
Velikovich et al. build a truly massive graph:

> For this study, we used an English graph where the node set *V* was based on all *n*-grams up to length 10 extracted from 4 billion web pages. This list was filtered to 20 million candidate phrases using a number of heuristics including frequency and mutual information of word boundaries. A context vector for each candidate phrase was then constructed based on a window of size six aggregated over all mentions of the phrase in the 4 billion documents. The edge set $E$ was constructed by first, for each potential edge (v ,v), computing the cosine similarity value between context vectors. All edges

> i j
> (v_i,v_j) were then discarded if they were not one of the 25 highest weighted edges adjacent to either node v_i or v_j.

Some highlights of their lexicon are given in **figure 3**.

Figure 3 Web prop lexicon



POSITIVE PHRASES			NEGATIVE PHRASES		
Typical	**Multiword expressions**	**Spelling variations**	**Typical**	**Multiword expressions**	**Vulgarity**
cute	once in a life time	loveable	dirty	run of the mill	fucking stupid
fabulous	state - of - the - art	nicee	repulsive	out of touch	fucked up
cuddly	fail - safe operation	niice	crappy	over the hill	complete bullshit
plucky	just what the doctor ordered	cooool	sucky	flash in the pan	shitty
ravishing	out of this world	coooool	subpar	bumps in the road	half assed
spunky	top of the line	koool	horrendous	foaming at the mouth	jackass
enchanting	melt in your mouth	kewl	miserable	dime a dozen	piece of shit
precious	snug as a bug	cozy	lousy	pie - in - the - sky	son of a bitch
charming	out of the box	cosy	abysmal	sick to my stomach	sonofabitch
stupendous	more good than bad	sikk	wretched	pain in my ass	sonuvabitch

Table 3: Example positive and negative phrases from web lexicon.

I think this approach is extremely promising, but I have not yet had the chance to try it out on sub-Google-sized corpora. However, I do have a small Python implementation for reference:

Code Implementation of the vector-based propagation algorithm: **webpropagate.py**

6 Related approaches

Once you have a distributional matrix, there are lots of things you can do with it:

1. Clustering
2. Topic modeling
3. Visualizations via 2d embedding

The central challenge for using these approaches in sentiment analysis is that they all tend to favor content-level associations over sentiment associations. That is, if you feed them the full vocabulary of your corpus, or the top *n* words from it, then you are unlikely to get sentiment-like groupings back. Some methods for addressing this challenge:

1. Filter the vocabulary to known sentiment words. (The drawback is that you won't learn any new ones.)
2. Filter the vocabulary to adjectives and adverbs. (The drawback is that you'll miss sentiment elsewhere.)
3. Bring in sentiment metadata to guide the unsupervised clustering model. This could be done with Labeled LDA (**Ramage, Hall, Nallapati, and Manning 2009**) or the related approach of **Maas, Daly, Pham, Huang, Ng and Potts (2010)**, which is focused explicitly on learning sentiment-rich word vectors.

7 Summary of conclusions

1. Vector-space models seem to be capable of learning large and powerful feature sets capturing a variety of sentiment contrasts.
2. The number of design choices is dizzying, and there are relatively few proven guidelines, but this could be freeing, right?

Home | ©2011 **Christopher Potts**

Christopher Potts, **Stanford Linguistics**

Sentiment Symposium Tutorial: Context

1. **Overview**
2. **Understanding the metadata**
    - 2.1 **Star rating imbalances**
    - 2.2 **Category relationships**
    - 2.3 **Metadata interpretation**
    - 2.4 **Perspective**
3. **Factors correlated with overall assessment**
    - 3.1 **The more reviews the better**
    - 3.2 **Helpfulness ratings**
    - 3.3 **Text length**
4. **Users**
    - 4.1 **Intra-author consistency**
    - 4.2 **Inter-author diversity**
    - 4.3 **User style**
5. **Topic-relative analysis**
6. **Long-suffering fans and thwarted expectations**
7. **Additional predictive information**
8. **Summary of conclusions**

1 Overview

Sentiment is highly variable and context-dependent. The goal of this section is to highlight some ways in which you can improve system performance by embracing this:

- Bring contextual features into your models.
- Understand the social dynamics of your domain.

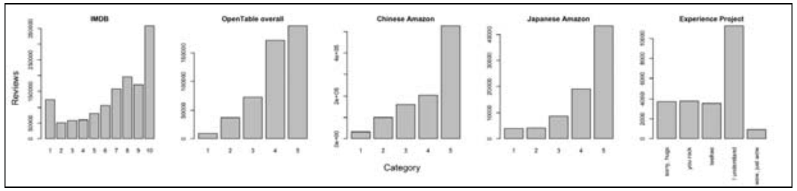
2 Understanding the metadata

There can be enormous benefits to devoting time and resources to understanding your data and associated annotations before you start to build a sentiment model.

2.1 Star rating imbalances

Figure 1 summarizes one of the major challenges to sentiment systems trained on naturalistic annotations like star-ratings: it is almost invariably the case that some categories are vastly over-represented. (**This section** is largely about getting around this problem when building lexicons.)

Figure 1 Imbalances in the distribution of texts relative to categories. For the ratings corpora (panels 1-4), positive dominates. For **Experience Project**, sympathy and solidarity dominate.



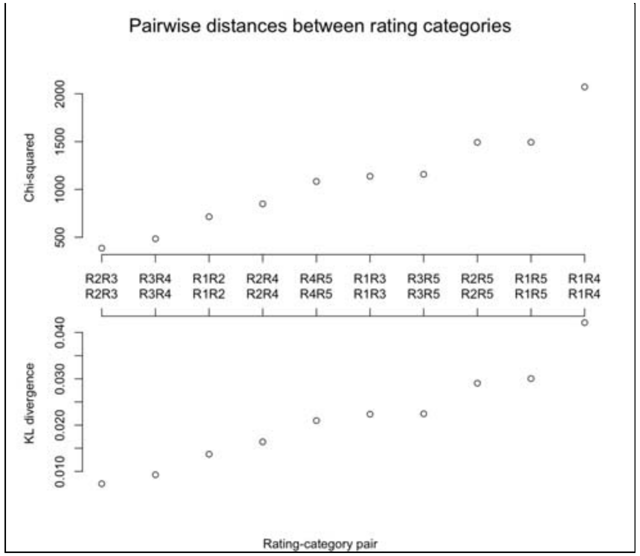
2.2 Category relationships

For many applications, you will want to reduce the dimensionality of your label set.

If you have hundreds of labels, then consider building a word × label matrix using the techniques in the **vectors section** and then seeing if you can effectively combine columns.

For smaller category sets like star ratings, you might want to directly measure the distance between them, to see if natural divisions emerge. **Figure 2** uses two probabilistic methods for making such comparisons for a five-star rating system. Strikingly, it looks like a natural division would be a three-way one grouping the entire middle of the scale against its edges.

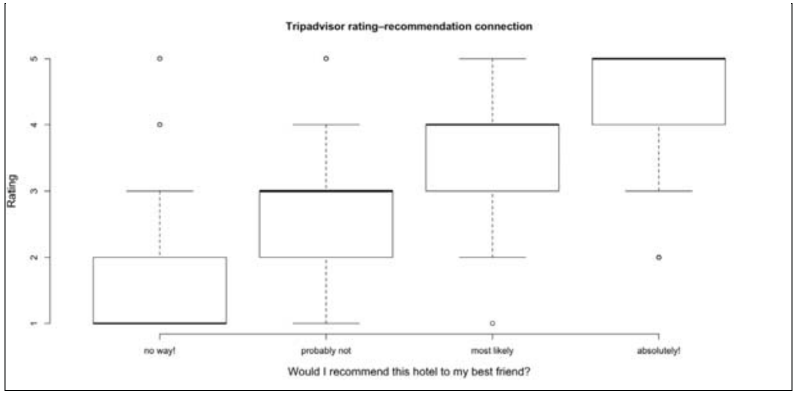
Figure 2 Category distances. The top panel gives the distances according to their chi-squared statistics (Kilgarriff and Rose 1998; Manning and Schütze 1999:171), and the bottom panel gives their KL-divergences. The orderings are the same in both cases, and they suggest that we might do well to treat the ratings scale as having three parts: the lowest of the low (1 star), the mushy middle (3-4 stars), and the best of the best (5 star).



2.3 Metadata interpretation

Although star-rating systems are by now widely understood, we might still worry that some users are confused. For example, in many parts of Europe, 1 is the best mark one can get in school, and this seems to lead some users to pick 1-star when giving very positive reviews. **Figure 3** is reassuring, though: it shows a high correlation between intuitive natural language phrases and star ratings.

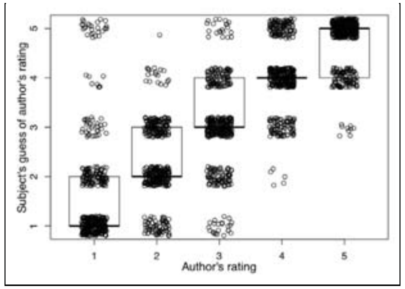
Figure 3 Users who both answered the question on the x-axis and gave a rating (y-axis) provide a window into how people conceptualize the rating scale. The dark horizontal lines indicate the median ratings, with boxes surrounding 50% of the ratings. The outliers could trace to people who treat the stars as a ranking system where 1 is the best.



2.4 Perspective

The star rating is primarily an indicator of the speaker/author vantage point. However, there is good reason to believe that readers/hearers are able to accurately recover the chosen star rating based on their reading of the text. **Potts (2011)** reports on an experiment conducted with Amazon's Mechanical Turk in which subjects were presented with 130-character reviews from OpenTable and asked to guess which rating the author of the text assigned (1-5 stars). **Figure 4** summarizes the results: the author's actual rating is on the x-axis, and the participants' guesses on the y-axis. The responses have been jittered around so that they don't lie atop each other. The plot also includes median responses (the black horizontal lines) and boxes surrounding 50% of the responses. The figure reveals that participants were able to guess with high accuracy which rating the author assigned; the median value is always the actual value, with nearly all subjects guessing within one star rating.

Figure 4 Experimental results suggesting that texts reliably convey to readers which star rating the author assigned.



3 Factors correlated with overall assessment

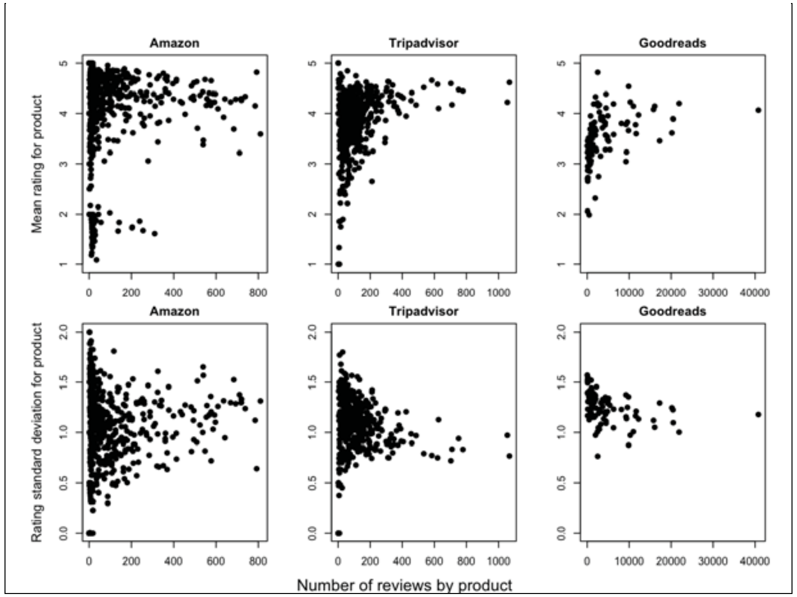
The environment in which the text was produced can impact sentiment in complex ways.

3.1 The more reviews the better

The observational fact is that the more reviews a product gets, the more consistent the ratings, and the more positive those ratings are. **Figure 5** summarizes for a few different corpora.

There is an intuitive explanation for this: a few bad reviews is enough to stop people from buying, which in turn leads them to stop reviewing. Conversely, positive reviews stimulate people to buy, which in turn increases the reviewer pool. (Movies and video games are somewhat exceptional in this regard. Both are products that many people buy immediately, sight unseen.)

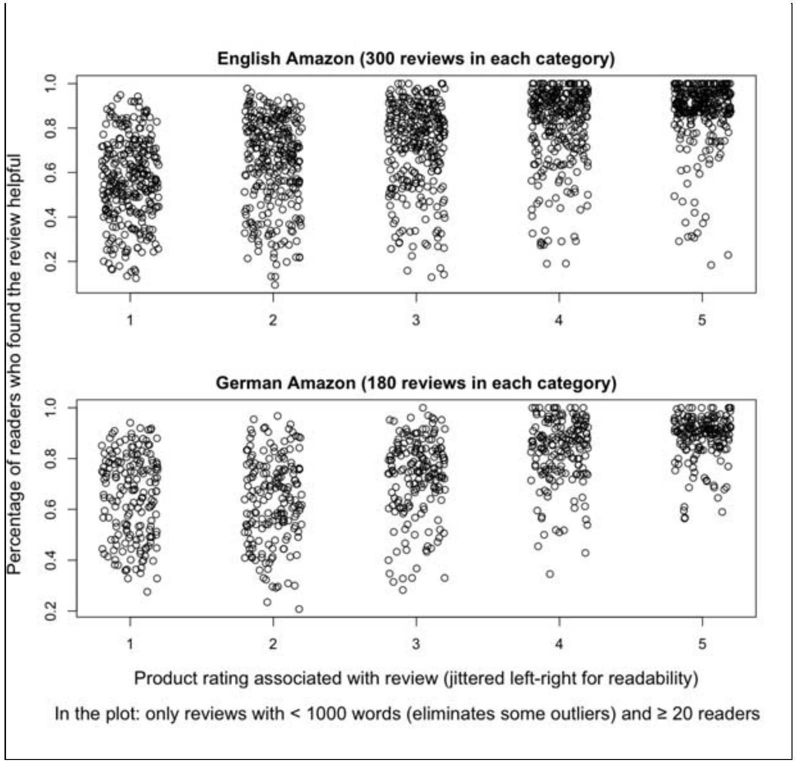
Figure 5 Star ratings and product review counts. High ratings and high review counts go hand-in-hand (top panels), and this results in a narrowing of opinions (bottom panels). This is presumably because products with a few negative reviews are purchased less and hence reviewed less.



3.2 Helpfulness ratings

Helpfulness ratings are predictors of sentiment, as seen in **figure 6**. I am not sure why this is, exactly, but its seems to be a very robust effect (**Ghose, Ipeirotis, and Sundararajan 2007**; **Danescu-Niculescu-Mizil, Kossinets, Kleinberg, and Lee 2009**). One factor that likely contributes is that online retailers tend to promote very positive reviews, which means they are read more than others.

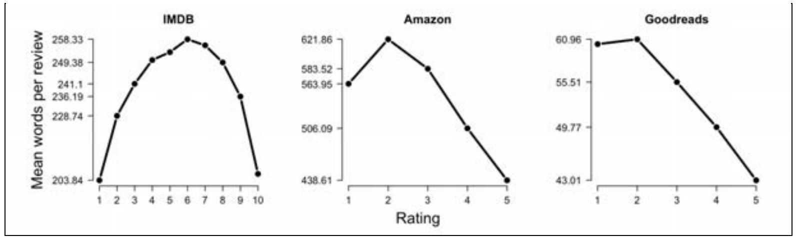
Figure 6 Many sites have meta-data saying "X of Y people found this review helpful", where Y is not the number of views, but rather the total number of people who selected "helpful" or "unhelpful". These plots show that there is a correlation between star ratings and helpfulness ratings: the higher the star rating, the more helpful people find the review.



3.3 Text length

Text length is another useful predictor (**figure 7**). Short texts tend to be highly emotive (positive or negative). Longer texts tend to be balancing perspectives, so they are longer on average.

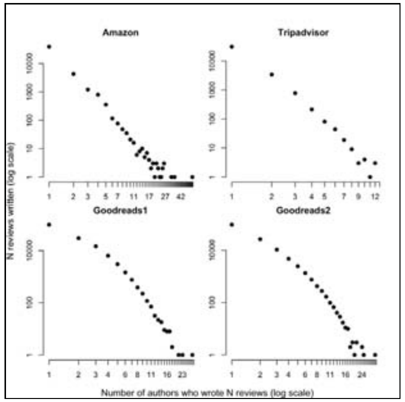
Figure 7 Mean review length for three corpora. Middle of the road reviews tend to be longer, presumably because they likely to balance evidence rather than just broadcastings a strongly held opinion.



4 Users

Authorship follows the a **Zipfian pattern**: most authors contribute one or two texts, and a handful contribute huge numbers of texts. Overall, user-level modeling can sharpen the overall picture.

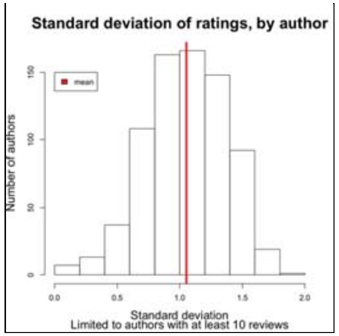
Figure 8 Authorship frequency spectrum.



4.1 Intra-author consistency

Individual reviewers tend to give the same ratings repeatedly (**figure 9**).

Figure 9 Reviewer rating standard deviations



This pattern probably arises because reviewers simply use different parts of the scale: some very kind reviewers never dip below three stars no matter how much they dislike the product, whereas give our stars much more grudgingly. If this is the case, then it might pay off to z-score normalize the ratings of individual authors before you use them:

Definition: z-scores

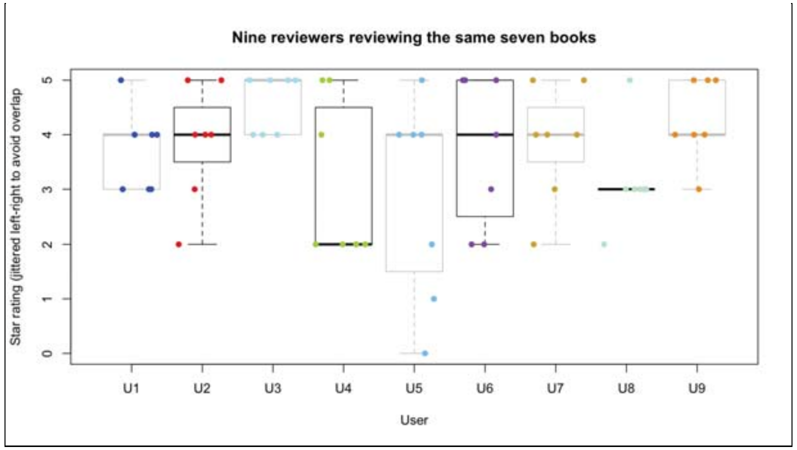
The z-score for a score *x* is (*x* - μ)/σ, where μ is the mean of the population for *x* and σ is the standard deviation of the population for *x*.

The population for a score could be the set of a reviewer's scores, or the scores for a product or product class, or the entire corpus.

4.2 Inter-author diversity

Though individual raters tend to confine themselves to a small part of the scale, raters differ from each other quite considerably. **Figure 10** supports this claim in a rather controlled way, by comparing nine users who all rated the same seven books on Amazon.

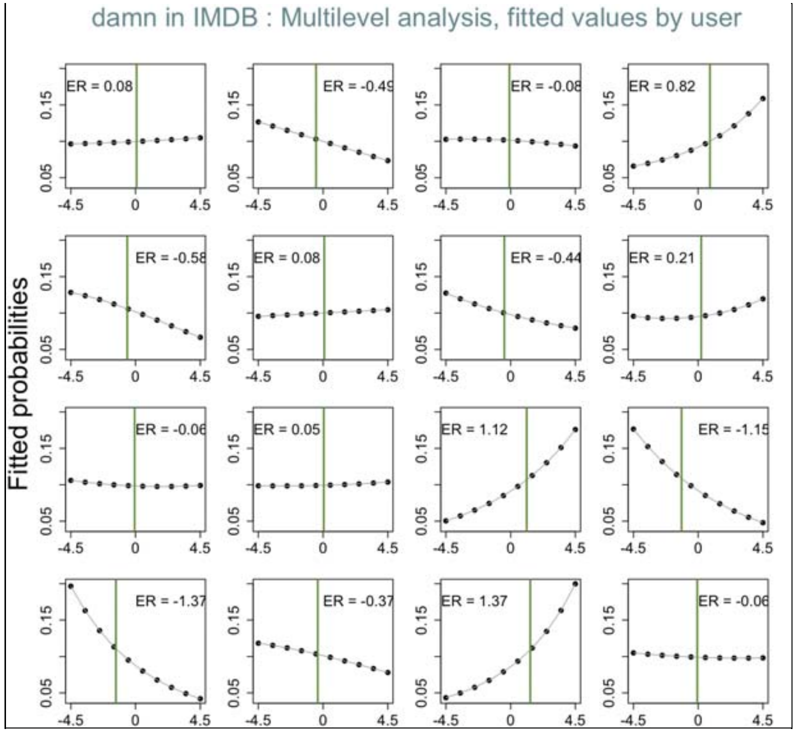
Figure 10 Reviewer comparisons for nine reviewers who rated the same seven books on Amazon. They tend to agree on which half of the scale is the right one, but there is a lot of variation within that space.



4.3 User style

Reviewers also use language differently from each other, so the more you can model these differences, the more accurate your analysis will be. **Figure 11** illustrates with a rather intuitive example concerning how people use curses.

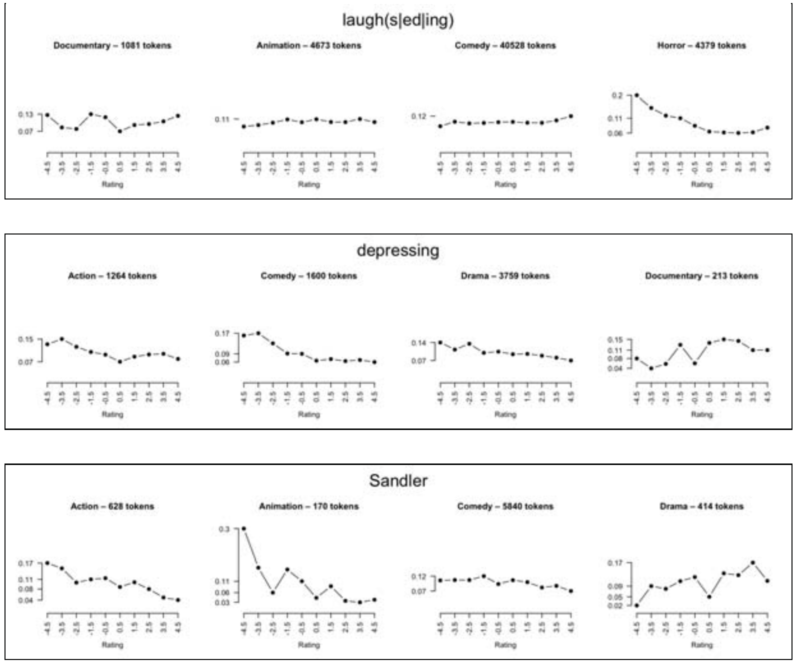
Figure 11 A portrait of individual variation: `damn` as used by 16 different users in a collection of reviews from IMDB. The panels depict the estimates from a fitted multi-level model in which the intercept and all the predictors are allowed to vary by user. Some users swear only when happy, others only when sad, others at either extreme, and still others just whenever.



5 Topic-relative analysis

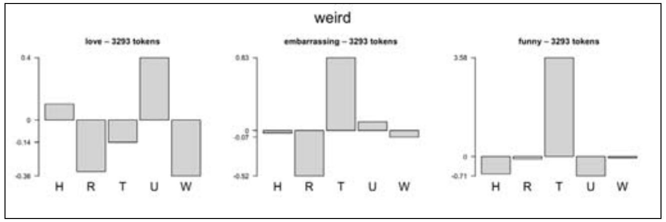
What you're talking about has a profound effect on the language you use. Thus, if you have topical information (forum name, product, product class, etc.) it should be included in your model. **Figure 12** shows some quick, intuitive examples of variation by movie genre in IMDB reviews, using the techniques described in the **review lexicon section**.

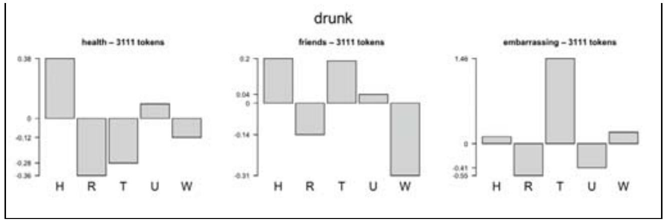
Figure 12 Genre effects from IMDB reviews.



Similarly, **confessions at the Experience Project** are tagged with group information, and these provide valuable clues as to how to understand the language (**figure 13**).

Figure 13 Group effects in Experience Project confessions.





Some sites have aspect-level metadata which can facilitate learning specific topic-level associations. Unfortunately, in my experience, the aspect-level ratings tend to be highly correlated with each other and with the overall rating, which detracts from their usefulness. The situation on OpenTable, summarized in **figure 14** and **figure 15**, is typical.

Figure 14 OpenTable rating distributions. Positive reviews dominate in all categories. Noise is fundamentally different, since it doesn't have a standard preference ordering.

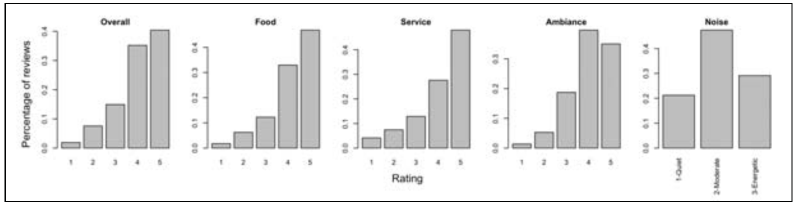
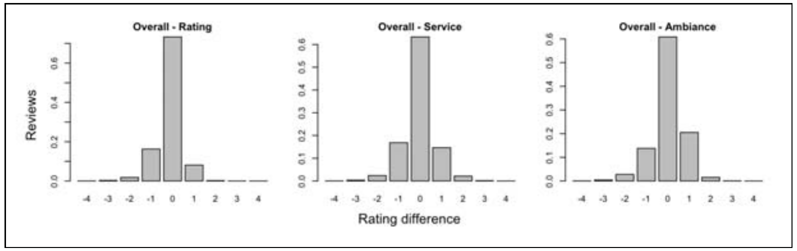


Figure 15 Comparisons with Overall. In each panel, the overall rating value is subtracted from the other rating value. Thus, a value of 0 indicates agreement between the two ratings for the review in question.



Demo Compare aspect-level scores from Open Table: **http://sentiment.christopherpotts.net/classify/**

6 Long-suffering fans and thwarted expectations

The ratio of positive to negative words as defined by a quality sentiment lexicon might be an indicator of the true sentiment of the text, in that extreme values are likely to be instances of thwarted expectations.

Table 1 An example of thwarted expectations. This is a negative review. Inquirer positive terms are in `blue`, and Inquirer negative terms are in `red`. There are 20 positive terms and six negative ones, for a Pos:Neg ratio of 3.33.

> i had been looking `forward` to this film since i heard about it early last year , when matthew perry had `just` signed on . i'm big fan of perry's `subtle sense` of `humor` , and in addition , i think chris farley's on-edge , extreme acting was a riot . so naturally , when the trailer for " almost heroes " `hit` theaters , i almost jumped up and down . a soda in `hand` , the lights dimming , i was ready to be blown away by farley's final starring role and what was supposed to be matthew perry's big breakthrough . i was ready to be `just` amazed ; for this to be among farley's `best` , in `spite` of david spade's `absence` . i was ready to be laughing my head off the minute the credits ran . sadly , none of this came to `pass` . the `humor` is spotty at `best` , with `good` moments and laughable one-liners few and far between . perry and farley have no chemistry ; the role that perry was cast in seems obviously written for spade , for it's his type of `humor` , and not at all what perry is associated with . and the movie tries to be `smart` , a subject `best` left alone when it's a farley flick . the movie is a `major` dissapointment , with only a few scenes `worth` a first look , `let` alone a second . perry delivers not one `humorous` line the whole movie , and not surprisingly ; the only reason the movie made the top ten grossing list opening week was because it was advertised with farley . and farley's `classic humor` is widespread , `too` . almost heroes almost works , but misses the wagon-train by quite a longshot . guys , let's leave the exploring to lewis and clark , huh ? `stick` to " tommy boy " , and we'll all be " friends " .

Suggestion: create a real-valued feature that is the Pos:Neg ratio if that ratio is below 1 (lower quartile for the whole PangLee data set) or above 1.76 (upper quartile), else 1.31 (the median). The goal is to single out "imbalanced" reviews as potentially untrustworthy at the level of their unigrams. (For similar idea, see Pang, Lee, and Vaithyanathan 2002.)

7 Additional predictive information

The above is just a sample. Any contextual information you can get your hands on will be valuable. Here's a sample of others that you're likely to be able to bring into a real-world system with relative ease.

1. Length of time on the market
2. The number of posts and the distribution of posting times
3. Price relative to competitions/comparables
4. Demographics (age, location, education and other carriers of social meaning)
5. The product's depiction in advertising and official descriptions

8 Summary of conclusions

1. Devote some time to understanding the properties of your annotations. This is always important for naturalistic data. But even if it's your own annotation scheme, it might harbor some significant latent structure.
2. General meta-properties like the number of reviews for a product and the length of time the product has been on the market are likely to be sentiment indicators, due to social effects.
3. Basic properties of the text (length, vocab size, etc.) can be significant predictors of sentiment.
4. Bring in usernames as features. They are likely to carry highly predictive information.
5. In a similar vein, if your data come from multiple corpora, that high-level source information should be included too.
6. In a similar vein, any demographic information you have about the author will be valuable.
7. Bring in topic/aspect level features (genre, topical group, etc.), as this is likely to profoundly affect sentiment information.

Home |

Sentiment tutorial home **Christopher Potts**, **Stanford Linguistics**

Sentiment Symposium Tutorial: Summarization

1. **Overview**
2. **Why visualize?**
3. **Visualization best practices**
4. **Words and lexicons**
5. **Products and services**
6. **Tools**
7. **Summary of conclusions**

1 Overview

This section focusses on sentiment summarization via visualization. While there is work on textual sentiment summarization, I think high-level visual summaries are better in this area. Any linguistic summary will leave out important nuances of the original source texts, which could be misleading. Of course, visual summaries can make such mistakes too, but we expect them to be high-level and approximate, so we are less likely to be misled.

The central online demos all summarize their results visually in addition to providing numerical information:

Demo Lexicon visualization **http://sentiment.christopherpotts.net/lexicon/**

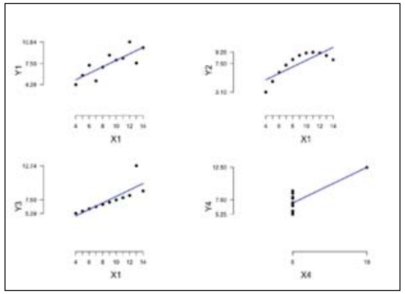
Demo Text scoring: **http://sentiment.christopherpotts.net/textscores/**

Demo Trained model predictions: **http://sentiment.christopherpotts.net/classify/**

2 Why visualize?

It's often the case that a visualization can capture nuances in the data that numerical or linguistic summaries cannot easily capture. **Figure 1** is a famous example involving datasets that might be summarized in the same way but nonetheless have very different properties.

Figure 1 Anscombe's Quartet (**Anscombe 1973**), via **Tufte (2001)**: four dramatically different data-sets with the same mean (7.50), standard deviation (2.03), and least-squares fit (3 + 0.5x).



3 Visualization best practices

Visualization is an art and a science in its own right, so I cannot hope to do justice to it here. The following advice from **Tufte (2001, 2006)** is easy to keep in mind (if only so that your violations of it are conscious and motivated):

1. Draw attention to the data, not the visualization.
2. Use a minimum of ink.
3. Avoid creating graphical puzzles.
4. Use tables where possible.

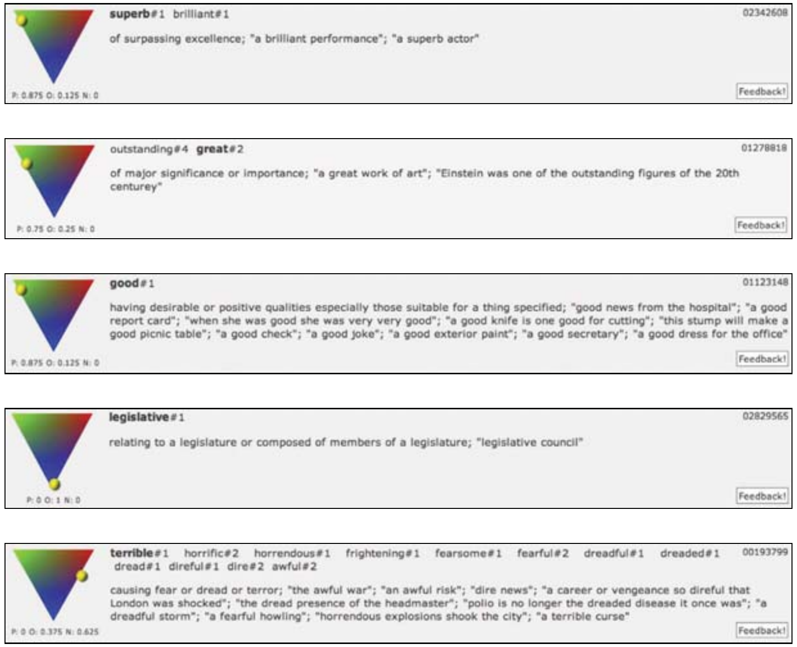
And some basic experimental evidence concerning effective visualization:

1. Proportion judgments: highest accuracy with side-by-side bar charts (**Cleveland and McGill 1984**; **Heer and Bostock 2010**).
2. For Web-based displays: gridlines improve accuracy; for a 0-100 scale, at least 80 pixels (no evidence that increasing height beyond that helps; **Heer and Bostock 2010**).

4 Words and lexicons

The online interface to **SentiWordNet** uses colored triangles to place words in a space defined by positive, negative, and neutral sentiment (**figure 2**).

Figure 2 **SentiWordNet** lexical visualizations.



superb#1 brilliant#1 02342608

of surpassing excellence; "a brilliant performance"; "a superb actor"

P: 0.875 O: 0.125 N: 0

outstanding#4 **great**#2 01278818

of major significance or importance; "a great work of art"; "Einstein was one of the outstanding figures of the 20th centurey"

P: 0.75 O: 0.25 N: 0

good#1 01123148

having desirable or positive qualities especially those suitable for a thing specified; "good news from the hospital"; "a good report card"; "when she was good she was very very good"; "a good knife is one good for cutting"; "this stump will make a good picnic table"; "a good check"; "a good joke"; "a good exterior paint"; "a good secretary"; "a good dress for the office"

P: 0.875 O: 0.125 N: 0

legislative#1 02829565

relating to a legislature or composed of members of a legislature; "legislative council"

P: 0 O: 1 N: 0

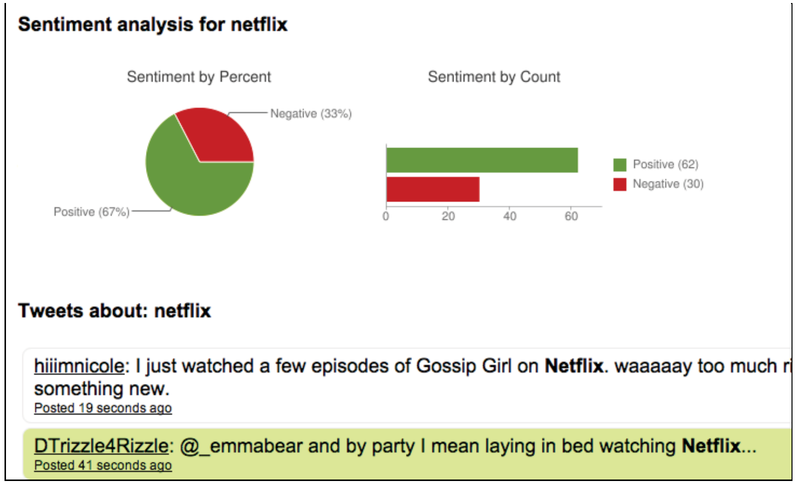
terrible#1 horrific#2 horrendous#1 frightening#1 fearsome#1 fearful#2 dreadful#1 dreaded#1 dread#1 direful#1 dire#2 awful#2 00193799

causing fear or dread or terror; "the awful war"; "an awful risk"; "dire news"; "a career or vengeance so direful that London was shocked"; "the dread presence of the headmaster"; "polio is no longer the dreaded disease it once was"; "a dreadful storm"; "a fearful howling"; "horrendous explosions shook the city"; "a terrible curse"

P: 0 O: 0.375 N: 0.625

Twitter Sentiment uses **Google Charts** to summarize its search results, and it also provides the raw data so that users can probe more deeply (**figure 3**).

Figure 3 **Twitter Sentiment** results for `Netflix`.



Sentiment analysis for netflix

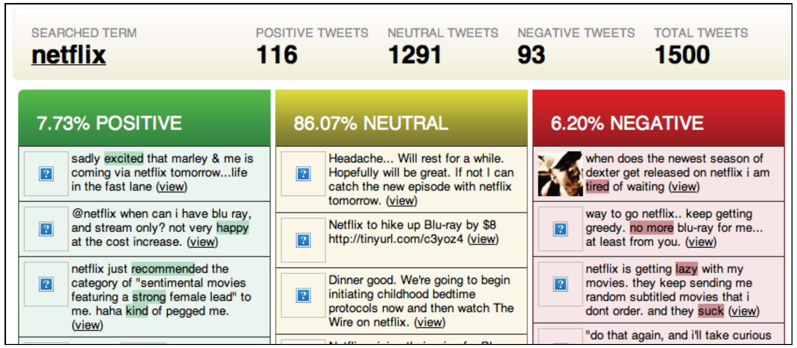
Tweets about: netflix

hiiimnicole: I just watched a few episodes of Gossip Girl on **Netflix**. waaaaay too much ri something new.
Posted 19 seconds ago

DTrizzle4Rizzle: @_emmabear and by party I mean laying in bed watching **Netflix**...
Posted 41 seconds ago

Twitrratr blends the data and summarization together (**figure 4**).

Figure 4 **Twitrratr** results for `Netflix`.



SEARCHED TERM	POSITIVE TWEETS	NEUTRAL TWEETS	NEGATIVE TWEETS	TOTAL TWEETS
netflix	116	1291	93	1500

7.73% POSITIVE	86.07% NEUTRAL	6.20% NEGATIVE
sadly excited that marley & me is coming via netflix tomorrow...life in the fast lane (view)	Headache... Will rest for a while. Hopefully will be great. If not I can catch the new episode with netflix tomorrow. (view)	when does the newest season of dexter get released on netflix i am tired of waiting (view)
@netflix when can i have blu ray, and stream only? not very happy at the cost increase. (view)	Netflix to hike up Blu-ray by $8 http://tinyurl.com/c3yoz4 (view)	way to go netflix.. keep getting greedy. no more blu-ray for me... at least from you. (view)
netflix just recommended the category of "sentimental movies featuring a strong female lead" to me. haha kind of pegged me. (view)	Dinner good. We're going to begin initiating childhood bedtime protocols now and then watch The Wire on netflix. (view)	netflix is getting lazy with my movies. they keep sending me random subtitled movies that i dont order. and they suck (view)
		"do that again, and i'll take curious

We Feel Fine aggregates enormous amounts of data and then visualizes the results for strings of the form `we feel X` (**figure 5**).

Figure 5 Visualizations from **We Feel Fine**.

Figure 6 uses **the t-SNE algorithm** to embed a very high dimensional lexicon into a 2d space.

Figure 6 A 2d embedding of a lexicon derived from **Experience Project data** using an extension of the model from **Maas, Daly, Pham, Huang, Ng and Potts 2010**.

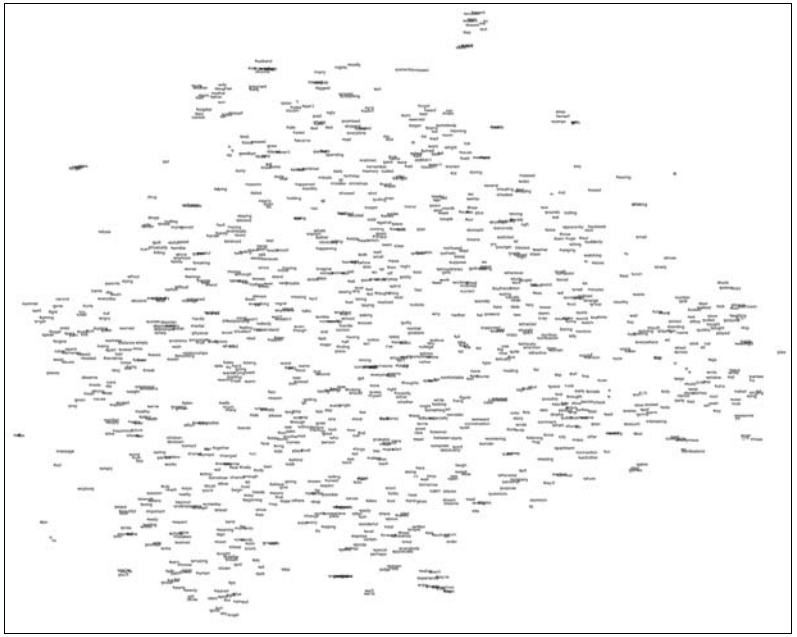


Figure 7 visualizes scores derived from the IMDB and Experience Project websites using the methods described in the lexicons section.

Figure 7 Merged IMDB and EP lexicons. The x-axis represents attenuation and emphasis. The y-axis represents sentiment polarity. The colors represent the (largely orthogonal) Experience Project scores.

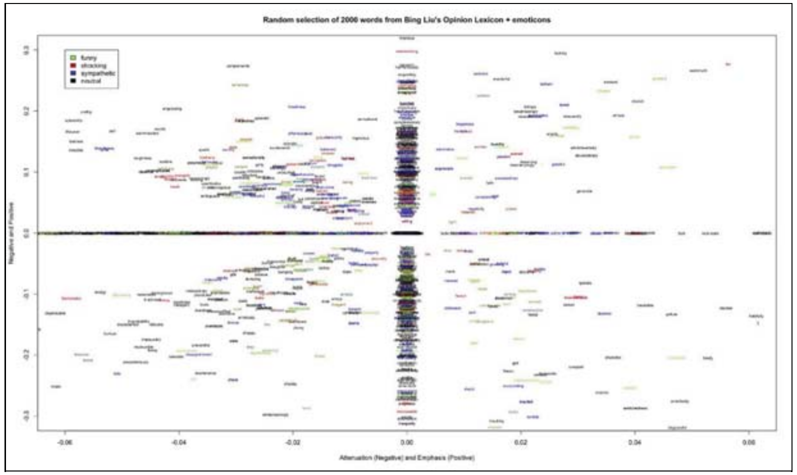
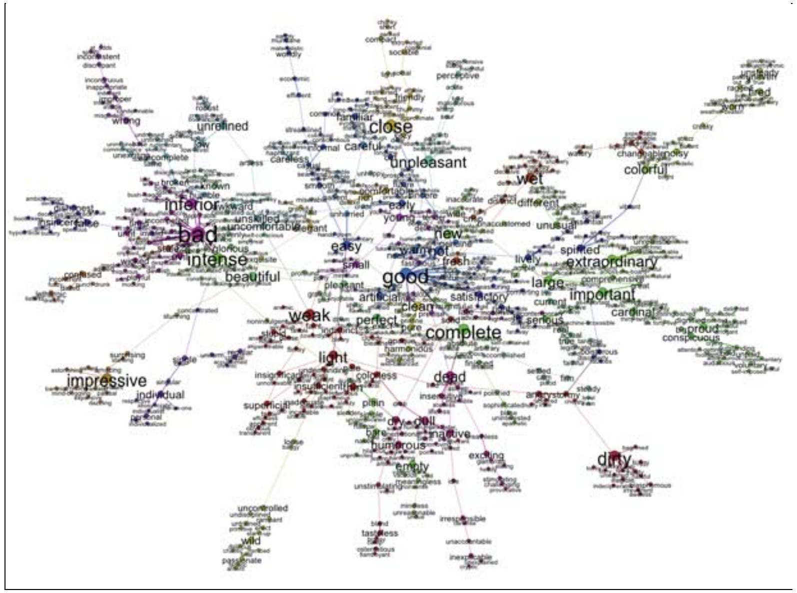


Figure 8 uses the **Gephi** social networking program to graph the relationships between modifiers in **WordNet** as given by the similar-to graph.

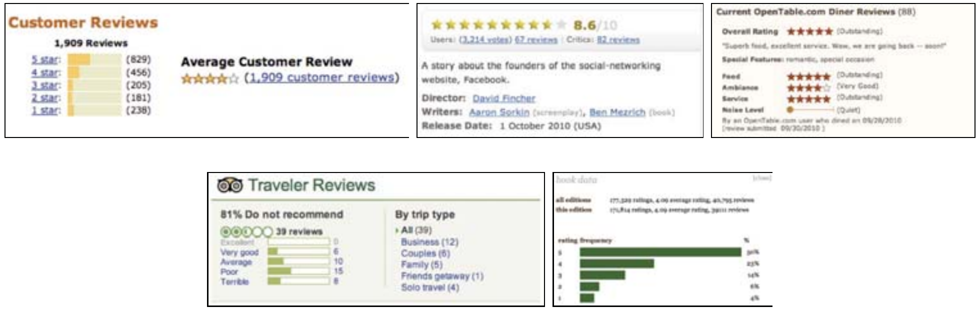
Figure 8 WordNet modifier relationships visualized using **Gephi**.



5 Products and services

Many online retailers and social networking sites do an excellent job of summarizing rating information about specific products and services. I think the summaries in **figure 9** work particularly well.

Figure 9 Effective rating summaries of products and services.


Customer Reviews
1,909 Reviews
5 star: (829)
4 star: (456)
3 star: (205)
2 star: (181)
1 star: (238)
Average Customer Review
☆☆☆☆☆ (1,909 customer reviews)

★★★★★★★★★★ 8.6/10
Users: (3,214 votes) 67 reviews | Critics: 82 reviews
A story about the founders of the social-networking website, Facebook.
Director: David Fincher
Writers: Aaron Sorkin (screenplay), Ben Mezrich (book)
Release Date: 1 October 2010 (USA)

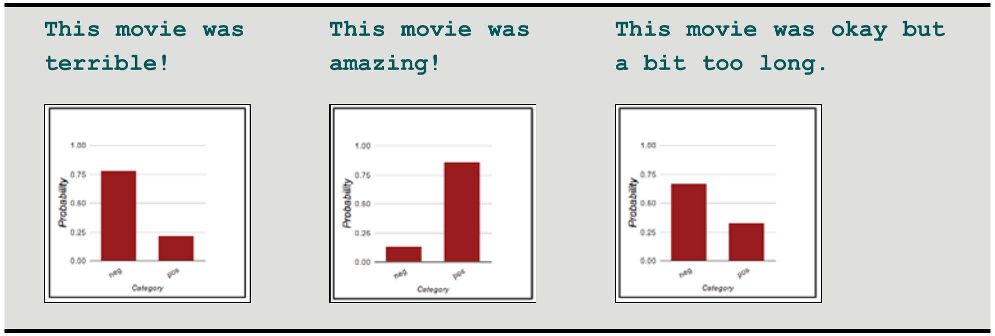
Current OpenTable.com Diner Reviews (88)
Overall Rating ★★★★★ (Outstanding)
"Superb food, excellent service. Wow, we are going back -- soon!"
Special Features: romantic, special occasion
Food ★★★★★ (Outstanding)
Ambiance ★★★★☆ (Very Good)
Service ★★★★★ (Outstanding)
Noise Level ● (Quiet)
By an OpenTable.com user who dined on 09/28/2010
[review submitted 09/30/2010]

Traveler Reviews
81% Do not recommend
●●●○○ 39 reviews
Excellent 0
Very good 6
Average 10
Poor 15
Terrible 8
By trip type
All (39)
Business (12)
Couples (6)
Family (5)
Friends getaway (1)
Solo travel (4)

book data [close]
all editions [illegible]
this edition [illegible]
rating frequency %
5
4
3
2
1

If you build a classifier model, I think it makes sense to provide similar distributional information, so that it is apparent not only what predictions your system makes but also where it is particularly certain or uncertain. **Figure 10** provides some examples from **the classifier demo**.

Figure 10 Classifier predictions. The rightmost case is unclear, and this is reflected in the relatively slim margin by which neg wins our over pos, as compared with the more certain judgments of the other short reviews.



This movie was terrible!

This movie was amazing!

This movie was okay but a bit too long.

Bing Liu often uses boxplot-like visualizations to compare products along a variety of dimensions. In **figure 11**, this is particularly valuable, since reducing the comparison to a single number might be misleading, as each product has its own strengths and weaknesses (which individual users might care about at different levels).

Figure 11 Comparing two products along multiple dimensions (**Liu, Hu, and Cheng 2005**).

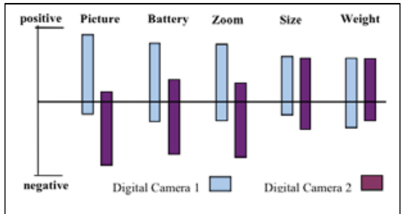
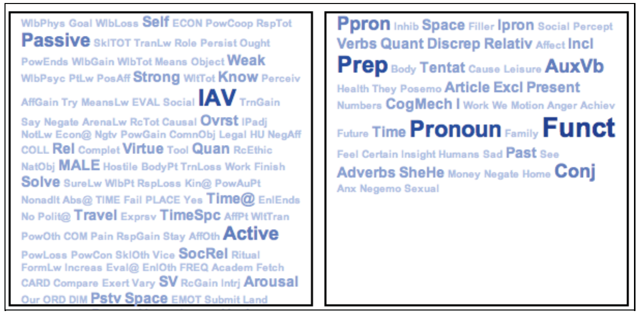


Figure 1: Visual comparison of consumer opinions on two products.

Finally, **Wordle** graphics are extremely popular these days. They usually represent the words in a text, with size corresponding to frequency (and the colors often randomly assigned). **Figure 12** does something slightly different: it visualizes a text as a cloud of semantic classes from the **Harvard General Inquirer** (left) and **LIWC** (right).

A cautionary note about Wordle: naive users tend to assume that the color choices are deliberate and that size corresponds to **importance**, a cognitively much deeper notion that frequency. (The practice of filtering very high-frequency function words encourages this misconception.)

Figure 12 Wordle-like visualizations of a text from the Experience Project. At left, the text is reduced to its Harvard General Inquirer classes, with the size of a class name given by the number of words from the text in that class. At right is the same kind of visualization using LIWC semantic classes.



WlbPhys Goal WlbLoss Self ECON PowCoop RspTot
Passive SklTOT TranLw Role Persist Ought
PowEnds WlbGain WlbTot Means Object Weak
WlbPsyc PtLw PosAff Strong WltTot Know Perceiv
AffGain Try MeansLw EVAL Social IAV TrnGain
Say Negate ArenaLw RcTot Causal Ovrst IPadj
NotLw Econ@ Ngtv PowGain ComnObj Legal HU NegAff
COLL Rel Complet Virtue Tool Quan RcEthic
NatObj MALE Hostile BodyPt TrnLoss Work Finish
Solve SureLw WlbPt RspLoss Kin@ PowAuPt
Nonadlt Abs@ TIME Fail PLACE Yes Time@ EnlEnds
No Polit@ Travel Exprsv TimeSpc AffPt WltTran
PowOth COM Pain RspGain Stay AffOth Active
PowLoss PowCon SklOth Vice SocRel Ritual
FormLw Increas Eval@ EnlOth FREQ Academ Fetch
CARD Compare Exert Vary SV RcGain Intrj Arousal
Our ORD DIM Pstv Space EMOT Submit Land
Ppron Inhib Space Filler Ipron Social Percept
Verbs Quant Discrep Relativ Affect Incl
Prep Body Tentat Cause Leisure AuxVb
Health They Posemo Article Excl Present
Numbers CogMech I Work We Motion Anger Achiev
Future Time Pronoun Family Funct
Feel Certain Insight Humans Sad Past See
Adverbs SheHe Money Negate Home Conj
Anx Negemo Sexual

6 Tools

Some accessible, open-source visualization toolkits:

- Prefuse and Prefuse Flare: **http://prefuse.org/**
- Many Eyes: **http://many-eyes.com/**
- Google Visualization APIs: **http://code.google.com/apis/chart/**
- T-distributed Stochastic Neighbor Embedding (t-SNE): **http://homepage.tudelft.nl/19j49/t-SNE.html**
- Gephi: **http://gephi.org/**

7 Summary of conclusions

1. Visualization is often the best way to summarize sentiment information.
2. Colors and shapes can provide a quick mental framework for sentiment analysis.
3. Dimensionality reduction is generally very important, but it should be done with caution.
4. Favor visualizations that convey the degree of certainty you have in your conclusions.
5. Try to make the raw data supporting your conclusions easily accessible, so that users can drill down to gain a better understanding.

Sentiment Symposium Tutorial: Bibliography

Andreevskaia, Alina and Sabine Bergler. 2006. Mining WordNet for a fuzzy sentiment: Sentiment tag extraction from WordNet glosses. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

Anscombe, Francis John. 1973. Graphs in statistical analysis. *American Statistician* 27: 17-21.

Sitaram Asur and Bernardo A. Huberman. 2010. Predicting the future with social media. arXiv:1003.5699v1.

Baccianella, Stefano, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation*, 2200-2204. European Language Resources Association.

Blair-Goldensohn, Sasha; Kerry Hannan; Ryan McDonald; Tyler Neylon; George A. Reis; and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era*. Beijing, China.

Bollen, Johan; Huina Mao; and Xiao-Jun Zeng . 2010. Twitter mood predicts the stock market. arXiv:1010.3003v1.

Cleveland, W. S. and R. McGill. 1984. Graphical perception: theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association* 79: 531-554

Constant, Noah; Christopher Davis; Christopher Potts; and Florian Schwarz. 2009. The pragmatics of expressive content: Evidence from large corpora. *Sprache und Datenverarbeitung* 33: 5-21.

Danescu-Niculescu-Mizil, Cristian; Gueorgi Kossinets; Jon Kleinberg; and Lillian Lee. 2009. How opinions are received by online communities: a case study on Amazon.com helpfulness votes. In *Proceedings of WWW*. ACL.

Sanjiv Das and Mike Chen. 2001. Yahoo! for Amazon: extracting market sentiment from stock message boards. In *Proceedings of the 8th Asia Pacific Finance Association Annual Conference*.

Deerwester, S.; S. T. Dumais; G. W. Furnas; T. K. Landauer; and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41: 391-407.

de Marneffe, Marie-Catherine and Christopher D. Manning. 2006. Stanford typed dependencies manual.

Eckert, Penelope. 2008. Variation and the indexical field. *Journal of Sociolinguistics* 12: 453-476.

Ekman, Paul. 1985. *Telling Lies: Clues to Deceit in the Marketplace, Politics, and Marriage*. New York: Norton.

Esuli, Andrea and Fabrizio Sebastiani. 2006. SentiWordNet: a publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation*, 417-422 Genova.

Fellbaum, Christiane. 1998. *WordNet: An Electronic Database*. Cambridge, MA: MIT Press.

Ghose, Anindya; Panagiotis Ipeirotis; and Arun Sundararajan. 2007. Opinion mining using econometrics: a case study on reputation systems. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague: ACL.

Godbole, Namrata; Manjunath Srinivasaiah; and Steven Skiena. 2007. Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media*.

Harris, Jesse A. and Christopher Potts. 2009. Perspective-shifting with appositives and expressives. *Linguistics and Philosophy* 32: 523-552.

Heer, Jeffrey and Michael Bostock. 2010. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *ACM Human Factors in Computing Systems*, 203-212.

Hu, Minqing and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 168-177. ACL.

Kaplan, David. 1999. What is meaning? Explorations in the theory of Meaning as Use. Brief version --- draft 1. Ms., UCLA.

Kim, Soo-Min and Eduard H. Hovy. 2006. Identifying and analyzing judgment opinions. In Robert C. Moore; Jeff A. Bilmes; Jennifer Chu-Carroll; and Mark Sanderson, eds., *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 200-207. ACL.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. ACL.

Klein, Dan and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In Suzanna Becker, Sebastian Thrun, Klaus Obermayer, ed., *Advances in Neural Information Processing Systems* 15, 3-10. Cambridge, MA: MIT Press.

Lasersohn, Peter. 2005. Context dependence, disagreement, and predicates of personal taste. *Linguistics and Philosophy* 28: 643-686.

Liu, Bing; Minqing Hu; and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th International World Wide Web Conference*. ACM.

Maas, Andrew L.; Raymond E. Daly; Peter T. Pham; Dan Huang; Andrew Y. Ng; and Christopher Potts. 2011. Learning word vectors for sentiment analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. ACL.

Mann, Gideon and Andrew McCallum. 2010. Generalized expectation criteria with application to semi-supervised classification and sequence modeling. *Journal of Machine Learning Research* 11: 955-984

Manning, Christopher D.; Dan Klein; William Tseng; Morgan Huihsin; and Anna N. Rafferty. 2008. Stanford log-linear part-of-speech tagger, version 1.6. Electronic document, **http://nlp.stanford.edu/software/tagger.shtml**.

McCallum, Andrew and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, 41-48. AAAI Press.

O'Connor, Brendan; Ramnath Balasubramanyan; Bryan R. Routledge, and Noah A. Smith. 2010. From Tweets to polls: linking text sentiment to public opinion time series. *Proceedings of the International AAAI Conference on Weblogs and Social Media*, 122-129. AAAI Press.

Pang, Bo; Lillian Lee; and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. ACL.

Plutchik, Robert. 2002. *Emotions and Life*. Washington, D.C.: American Psychological Association.

Potts, Christopher and Florian Schwarz. 2010. Affective 'this'. *Linguistic Issues in Language Technology* 3: 1-30.

Potts, Christopher. 2011. On the negativity of negation. In Nan Li and David Lutz, eds., *Proceedings of Semantics and Linguistic Theory 20*, 636-659. Ithaca, NY: CLC Publications.

Ramage, Daniel; David Hall; Ramesh Nallapati; and Christopher D. Manning. 2009. Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 248-256. ACL.

Rao, Delip and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, 675-682. ACL.

Scherer, Klaus R. 1984. Emotion as a Multicomponent Process: A model and some cross-cultural data. In P. Shaver, ed., *Review of Personality and Social Psych* 5: 37-63.

Socher, Richard; Jeffrey Pennington; Eric H. Huang; Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 151-161. ACL.

Stone, Philip J; Dexter C. Dunphry; Marshall S. Smith; and Daniel M. Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. Cambridge, MA: MIT Press.

Toutanova, Kristina; Dan Klein; Christopher D. Manning; and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, 252-259.

Tufte, Edward R. 2001. *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press.

Tufte, Edward R. 2006. *Beautiful Evidence*. Cheshire, CT: Graphics Press.

Turney, Peter D. and Michael L. Littman. 2003. Measuring praise and criticism: inference of semantic orientation from association. *ACM Transactions on Information Systems* 21: 315-346.

Turney, Peter D. and Patrick Pantel. 2010. From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research* 37: 141-188.

Valitutti, Alessandro; Carlo Strapparava; and Oliviero Stock. 2004. Developing affective lexical resources. *PsychNology Journal* 2: 61-83.

Velikovich, Leonid; Sasha Blair-Goldensohn; Kerry Hannan; and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 777-785. ACL.

Wiebe, Janyce; Theresa Wilson; and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation* 39: 165-210.

Wilson, Theresa; Jancye Wiebe; and Philip Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. ACL.

Home |