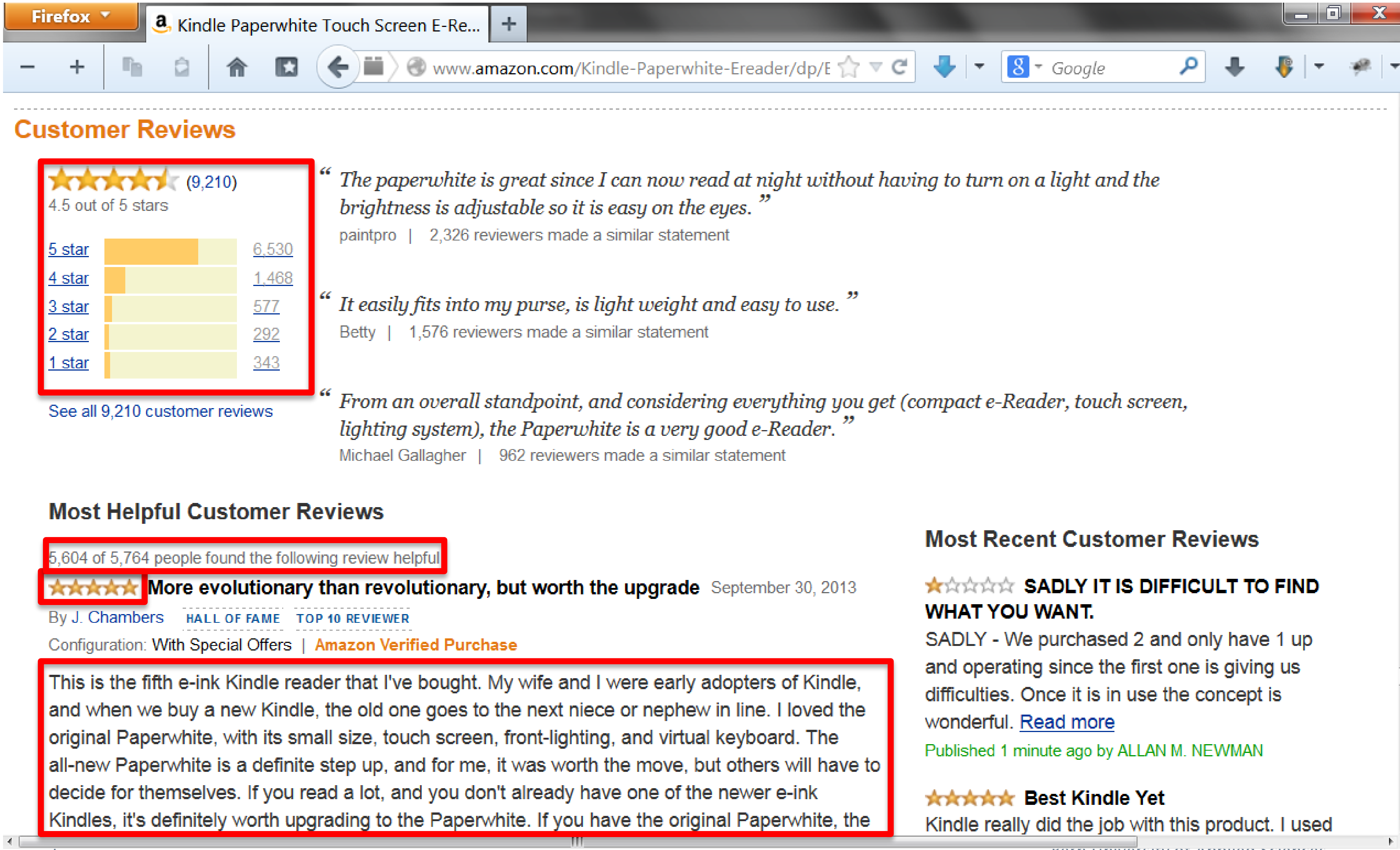
Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

BTI7535 Data Science

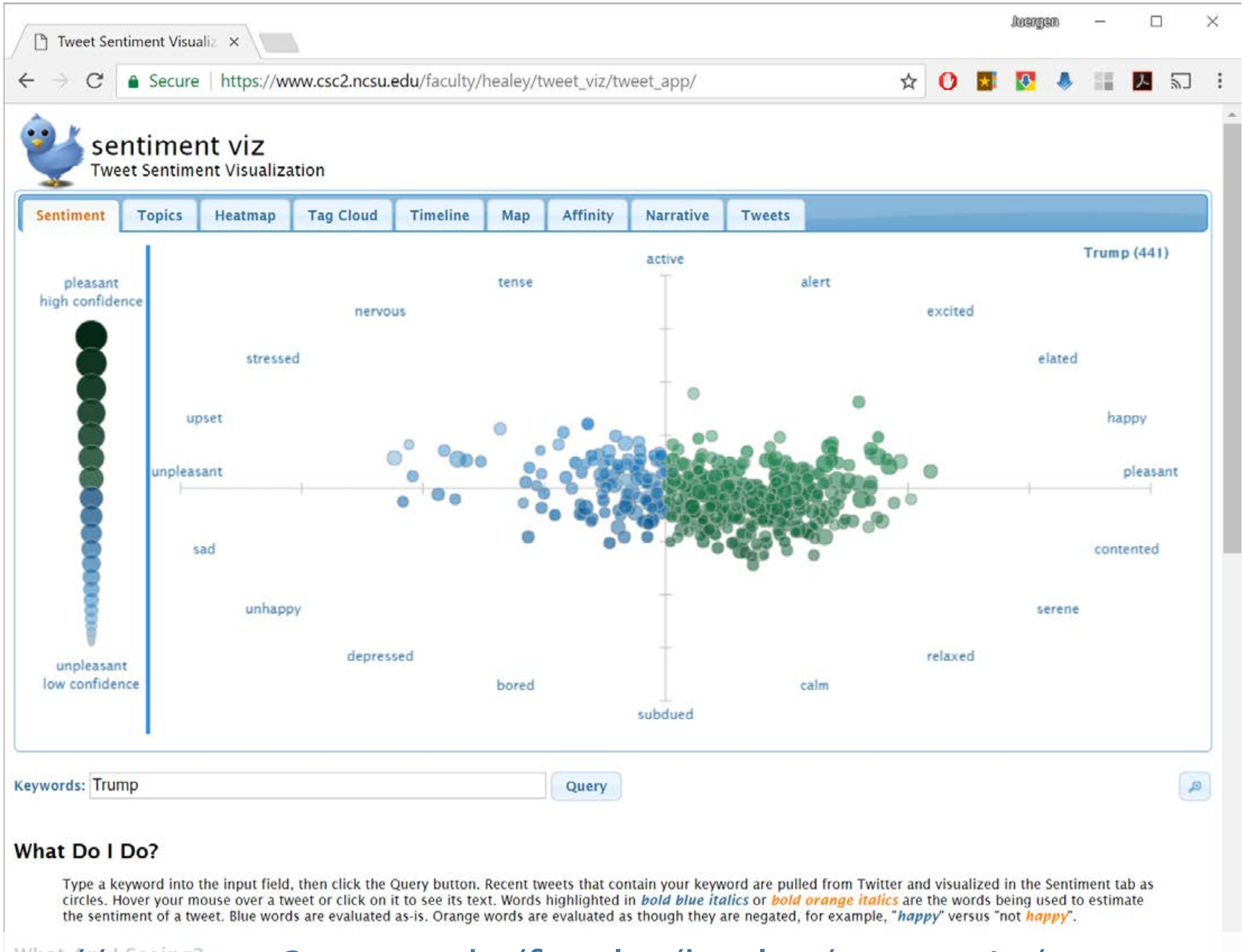
Sentiment Analysis

Prof. Dr. Jürgen Vogel (juergen.vogel@bfh.ch)

Amazon Product Reviews



Twitter



https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/

Sentiment Analysis

Sentiment Analysis

- Opinion/Sentiment analysis/extraction/mining/
- *Products, movies, events, …*: is this review positive or negative?
- *Politics*: what do people think about this candidate or issue?
- *Public sentiment*: how is consumer confidence? Is despair increasing?

Aspects for Sentiment Analysis

1. data source: social networks (facebook, twitter), shops, blogs, forums, ...
2. features
   - explicit rating
   - meta data: user, community, ...
     - graph analysis for social networks
   - text: using the connotation of words
     - the noun *hate* is defined as *very strong feeling of dislike* and thus as a *very negative* connotation
3. method: assessment of (combinations of) these features
   - (human-written) rules
   - machine learning

Sentiment Analysis Goals

1. Is the attitude of this text positive or negative?
   - "Simple"
   - *Of all the natural world style documentaries that have been done this is surely the best.*
   - *The movie Battlefield Earth is a disaster of epic proportions!*
2. Rank the attitude of this text from 1 to 5
   - "Complex"
   - *mediocre viewing. lacks depth and focus with too many people talking rather than allowing the viewer to learn about or experience nature*
3. Detect the target, source, or complex attitude types
   - "Tough"
   - *The ride was very rough but the attendants did an excellent job of making us comfortable.*

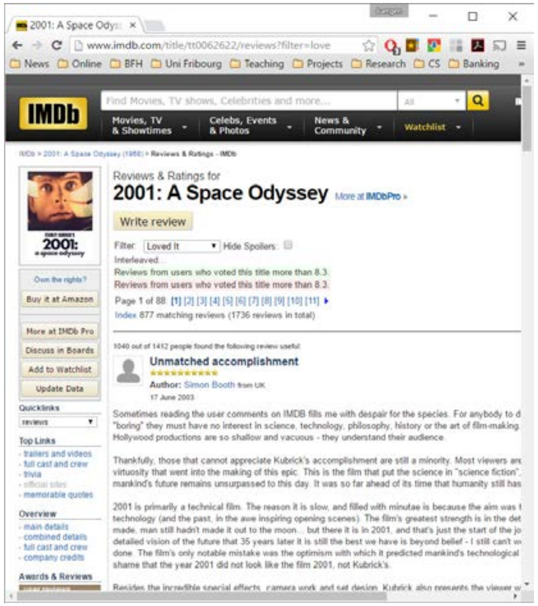
Baseline Method: Counting Sentiments

Sentiment Analysis

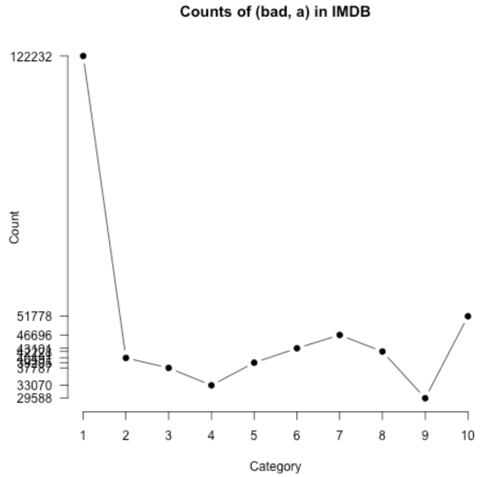
Word Sentiments

- How to map words to sentiments?
  - *like, love, want, fantastic* → positive sentiment
  - *dislike, unnecessary, defect* → negative sentiment
- Sentiment Lexicon
  - manually labeled
  - e.g., The General Inquirer (http://www.wjh.harvard.edu/~inquirer)
    - categories
      - Positiv (1915 words) and Negativ (2291 words)
      - Strong vs Weak, Active vs Passive, Overstated versus Understated
      - Pleasure, Pain, Virtue, Vice, Motivation, Cognitive Orientation, etc
    - free for research purposes
  - researchers also try to create (or extend) lexicons automatically

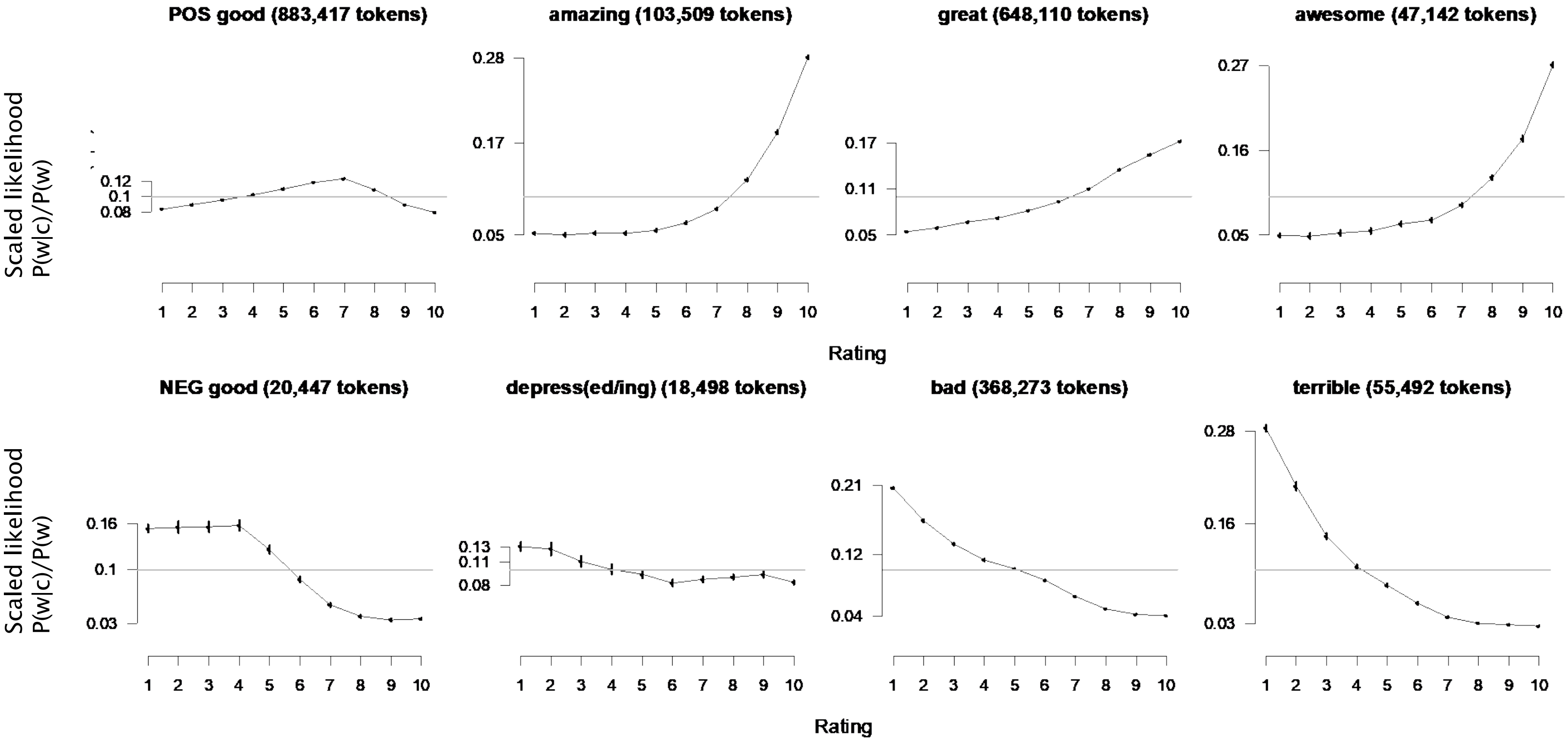
From Word Sentiments to Text Sentiment



- Can we use the appearance of "sentiment words" to determine the overall sentiment of a text?
- Study: Word sentiments in the movie review database IMDB (http://www.imdb.com/)
  - huge collection of movie reviews complete with rating from 1 to 10
  - how likely is each word to appear in each sentiment class?
  - approach: count("*bad*") in 1-star, 2-star, etc.
  - but can't use raw counts
  - instead, likelihood: $P(w|c) = \dfrac{f(w,c)}{\sum_{w \in c} f(w,c)}$
  - scale to make the likelihood comparable between words $\dfrac{P(w|c)}{P(w)}$



Analyzing the Polarity of Words in IMDB



Potts, Christopher. 2011. On the negativity of negation. SALT 20, 636-659.

Counting Word Sentiments

This is the fifth e-ink Kindle reader that I've bought. My wife and I were early adopters of Kindle, and when we buy a new Kindle, the old one goes to the next niece or nephew in line. I loved the original Paperwhite, with its small size, touch screen, front-lighting, and virtual keyboard. The all-new Paperwhite is a definite step up, and for me, it was worth the move, but others will have to decide for themselves. If you read a lot, and you don't already have one of the newer e-ink Kindles, it's definitely worth upgrading to the Paperwhite. If you have the original Paperwhite, the upgrade is well worth considering. Although I've only had the new Paperwhite a few hours, I'm already glad I upgraded.

Algorithm 1

- count words with pos and neg sentiments
- sentiment(text) = positive if #pos / (#pos + #neg) > 50%
  - here: 7 pos / (7 pos + 3 neg) → positive sentiment
- to increase the result confidence: > 60% - 80%

Counting Word Sentiments with Weights

Firefox | Kindle Paperwhite Touch Screen E-Re...

This is the fifth e-ink Kindle reader that I've bought. My wife and I were early adopters of Kindle, and when we buy a new Kindle, the **old** 1 *e goes to the next niece or nephew in line. I* **loved** 3 *original Paperwhite, with its* **small** 1 *e, touch screen, front-lighting, and virtual keyboard. The all-new Paperwhite is a* **definite step up,** 4 *d for me, it was* **worth** 1 *e move, but others will have to decide for themselves. If you read a lot, and you* **don't** 1 *eady have one of the* **newer** 1 *nk Kindles, it's* **definitely worth** 4 *upgrading to the Paperwhite. If you have the original Paperwhite, the upgrade is* **well worth** 4 *nsidering. Although I've only had the new Paperwhite a few hours, I'm already* **glad** 1 *pgraded.*

Most Recent Customer Reviews

Algorithm 2

- count weighted pos/neg sentiments of words or phrases
- sentiment(text) = positive if Σ pos / (Σ pos + Σ neg) > 50%
  - here: 18 pos / (18 pos + 3 neg) = 86% → highly pos sentiment

decide for themselves. If you read a lot, and you don't already have one of the newer e-ink Kindles, it's definitely worth upgrading to the Paperwhite. If you have the original Paperwhite, the

Best Kindle Yet
Kindle really did the job with this product. I used

Discussion

- algorithms like these may achieve up to 70% accuracy
- 30% wrong classifications?
  - compare to "just" 50% error rate of a random generator assigning pos of neg sentiment to a text

Sentiments

Sentiment Analysis

Generalized: Scherer Typology of Affective States

- **Emotion**
  - relatively brief episode of synchronized response
  - in response to the evaluation of an event as being of major significance
  - *angry, sad, joyful, fearful, ashamed, proud, elated*
- **Mood**
  - diffuse affect state, change in subjective feeling
  - of low intensity but relatively long duration, often without apparent cause
  - *cheerful, gloomy, irritable, listless, depressed, buoyant*
- **Interpersonal stances**
  - affective stance toward another person in a specific interaction
  - *friendly, flirtatious, distant, cold, warm, supportive, contemptuous*
- **Attitudes**
  - relatively enduring, affectively colored beliefs
  - preferences towards objects or persons
  - *liking, loving, hating, valuing, desiring*
- **Personality traits**
  - emotionally laden, stable personality and behavior tendencies
  - typical for a person
  - *nervous, anxious, reckless, morose, hostile, jealous*

Sentiments

Sentiment analysis is the detection of **attitudes**

= "enduring, affectively colored beliefs, dispositions towards objects or persons"

1. **Holder (source)** of attitude
2. **Target (aspect)** of attitude
3. **Type** of attitude
   - from a set of types
     - *Like, love, hate, value, desire,* etc.
   - or a weighted **polarity**:
     - *positive, negative, neutral,* together with *strength*
4. **Text** containing the attitude
   - Sentence or entire document

Advanced Method: Classification

Sentiment Analysis

Classification via Machine Learning

Classification in General

1. given
   - a representation of a document d
     - usually not the document itself but some features (i.e., properties) of the document
   - a fixed set of classes: $C = \{c_1, c_2, \ldots, c_J\}$
   - a training set *D* of documents each with a label in *C*
2. determine
   - a learning method/algorithm to learn a classifier $\gamma: D \rightarrow C$
3. result
   - for a test document *d*, we assign it the class $\gamma(d) = c \in C$

Classification-Based Sentiment Analysis

1. Text Preprocessing
   - Sentence Detection
   - Tokenization
   - POS
   - Phrase Chunking
   - ...
2. Feature Extraction
   - sentiment of individual words ("like")
   - negation ("don't like")
   - …
3. (Supervised) Classification
   - MaxEnt
   - Naïve Bayes
   - SVM
   - ...

Feature: Negation

- How to handle negation?
  - *I didn't like this movie* vs. *I really like this movie*
- Simple but effective: add *NOT_* to every word between negation and following punctuation
  - *didn't like this movie, but I*

    → *didn't NOT_like NOT_this NOT_movie, but I*

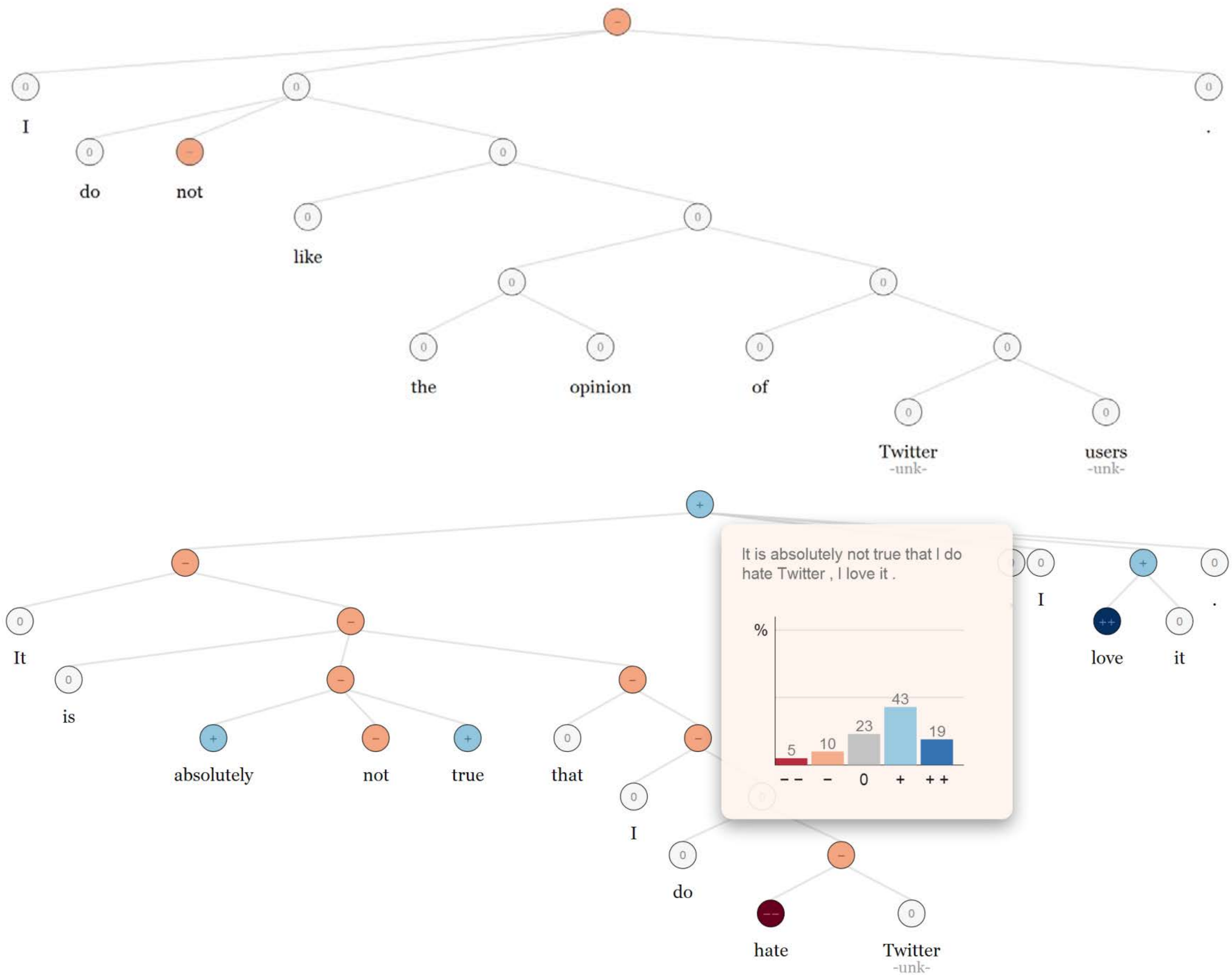
Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA).

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79—86.

Feature: Classifying Complete Sentences

- documents may consist of multiple parts with different sentiments
  - *As usual Keanu Reeves is nothing special. But Laurence Fishbourne is very talented.*
  - classify sentences individually
    - may also want to calculate overall document classification
  - but: sentences may also depend on each other
    - *This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it can't hold up.*
- sentences may consist of multiple parts with different sentiments
  - *The food was great but the service was awful*
  - here the sentiments have different targets (*food* and *service*)
  - differentiate via deep syntactic parsing or phrase chunking

Word Sentiments plus Sentence Structure

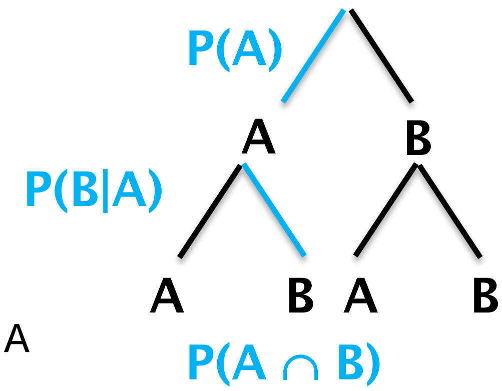


source: http://nlp.stanford.edu/sentiment/index.html

Naïve Bayes

Supervised Classification

Bayes' Theorem



- mathematical law about conditional probabilities
- given two events A and B, then the conditional probability $P(B|A)$ relates to the probability that event B occurs after A has occurred
  - applies, e.g., when we are blindly drawing samples from a bag containing red and black balls without returning the balls
    - assume we start with 2 red and 2 black balls
      - then $P(red) = 2/4 = P(black)$ for the first draw
    - if we draw a red ball first, then we know upfront for the 2nd draw
      - $P(red|red) = 1/3$ and $P(black|red) = 2/3$
- probability that two conditionally-related events $P(A)$ and $P(B|A)$ occur one after another: $P(A \cap B) = P(A) * P(B|A)$
  - e.g., probability to first draw red and then red again: $P(red \cap red) = P(red) * P(red|red) = 2/4 * 1/3 = 1/6$
- $P(A|B)$, $P(B|A)$ may be calculated as
  - $P(A|B) = P(A \cap B) / P(B)$ and $P(B|A) = P(A \cap B) / P(A)$ (for $P(B) \neq 0$ and $P(A) \neq 0$)
  - in the drawing example, $P(red|red) = 2/4 * 1/3 / 2/4 = 1/3$
- from this we can derive Bayes' theorem
  - $P(A|B) = P(B|A)\ P(A) / P(B)$

Using Bayes' Theorem for Classification

- our events are: A = (c)ategory, B = (d)ocument
- we calculate P(c|d)
  - the probability that a given document belongs into a certain class
  - based on certain features of the document
- given d, our classfier thus
  1. calculates P(c|d) for all c
  2. selects c with the highest P(c|d)

Using Bayes' Theorem for Classification

Calculating $P(c|d) = P(d|c)\ P(c)\ /\ P(d)$

- on the basis of a representative (and large) training set
- $P(d|c)$ probability of a document d given the category c
  $= P(f_1,\ f_2, ...,\ f_n|c)$ where f_i are the document features
  - in the simplest case, f_i are the individual words that d contains
  - we assume that f_i are independent from each other given a class c
    - thus we can caculate $P(f_1,f_2,\ldots,f_n|c)=P(f_1|c)*P(f_2|c)*\ldots*P(f_n|c)$
    - where $P(f_i|c)$ = # of word f_i in c / # of all words in c
      - i.e., create one large document from all d in class c and then calculate the frequency of f_i
  - of course, in practice such an assumption is wrong
    - would expect that within c, certain words are often co-located
    - thus the name: **naive bayes** classifier (NB)
  - the NB classifier performs well in practive despite this "naive" assumption
- $P(c)$ probability of a category c
  = # of documents within c / # of all documents
- $P(d)$ probability of a document d
  = 1 / # of documents
  - is equal for all possible $P(c|d)$ and thus irrelevant (just a scaling factor)

Example

	Doc	Words	Class
Train	1	Chinese Beijing Chinese	cn
	2	Chinese Chinese Shanghai	cn
	3	Chinese Macao	cn
	4	Tokyo Japan Chinese	jp
Test	5	Chinese Chinese Chinese Tokyo Japan	?

- P(c) = # of training documents within c / # of all training documents
  - P(cn) = 3/4
  - P(jp) = 1/4
- $P(d|c) = P(f_1|c) * P(f_2|c) * ... * P(f_n|c)$
  - # of words in cn: 8; in jp: 3
  - P(Chinese|cn) = (5+1) / 8 = 6/8
  - P(Tokyo|cn) = (0+1) / 8 = 1/8
  - P(Japan|cn) = (0+1) / 8 = 1/8
  - P(Chinese|jp) = (1+1) / 3 = 2/3
  - P(Tokyo|jp) = (1+1) / 3 = 2/3
  - P(Japan|jp) = (1+1) / 3 = 2/3
- P(c|d) = P(d|c) P(c) / P(d)
  - P(cn|d5) = 6/8 * 6/8 * 6/8 * 1/8 * 1/8 * 3/4 = 0.00494
  - P(jp|d5) = 2/3 * 2/3 * 2/3 * 2/3 * 2/3 * 1/4 = 0.03

Notes:
- P's for Beijing, Shanghai, and Macao are not listed
- in order to prevent that $P(f_i|c)$ is 0, we always add 1
- this variant of NB is called **multinomial** because we add up the occurrences
- if we only note the occurrence with 0 or 1, we have Bernoulli NB

Underflow Prevention

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since log(xy) = log(x) + log(y), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

Naive Bayes Classifier

- Classify based on prior weight of class and conditional parameter for what each word says:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \left[\log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j) \right]$$

- Training is done by counting and dividing:

$$P(c_j) \leftarrow \frac{N_{c_j}}{N} \qquad P(x_k \mid c_j) \leftarrow \frac{T_{c_j x_k} + \alpha}{\sum_{x_i \in V} [T_{c_j x_i} + \alpha]}$$

Naive Bayes is not so Naive

- Very fast learning and testing
  - basically just count feature occurrences
- Low storage requirements
- Optimal if the independence assumptions hold
- Very good in domains with many equally important features
- More robust to irrelevant features than many other learning methods
  - Irrelevant features cancel each other without affecting results
- A good dependable baseline for text classification
  - but not the best classifier