Sentiment Analysis

Definition

General goal:

- Extract opinions, sentiments and emotions express by hummans in texts and use this information for business, intelligence, etc. purposes.
- Can't be done manually because of huge volume optionated text.

Applications:

- Product review mining
- Review classification
- Tracking sentiments towards topics over time
- Prediction (elction outcomes, market trends)

Motivatons

- Objective Sentence
- Positive and negativte opinions about what?
- Targets of opinions
- Holders of opinions

Definitions:

- Current text processing methods (e.g., web search, information extraction) work with factual information.
- Current search ranking strategy not appropriate for opinion retrieval.
- Sentiment analysis focuses on **subjective statements - opinions, sentiments, emotions**: hard to express with a few keywords.

Step of sentiment analysis :

- **Subjectivity classification**
- **Sentiment classification**

Target objects: Product, person, event, organization, or topic. It is represented as

- A hierarchy of components, sub-components
- Each node represent a component and has a set of attributes.

Features: components/sub-components/attributes

Bing Liu's model:

An opinion is a quintuple (o_j , f_{jk} ,so_{ijkl} ,h_i ,t_l), where

- o_j is a target object.

- f_{jk} is a feature of the object o_j .
- so_{ijkl} is the sentiment value of the opinion of the
- opinion holder h_i (usually the author of the post)
- on feature f_{jk} of object o_j at time t_l .

  so_{ijkl} is positive, negative, neutral, or a more granular rating, such as 1-5 stars as in movie reviews.

The task of opinion mining : With that, one structure the unstructured.

- Discover all quintuples ($o_j, f_{jk}, so_{ijkl}, h_i, t_l$), or
- Discover some of these components

Granularity level: 粒度

Document level: classify a document based on the overall sentiment expressed by opinion holder into, e.g. positive or negative or neutral.

- Assumption: Each document focuses on a single object and contains opinions from a single opinion holder($o_j, f_{jk}, so_{ijkl}, h_i, t_l$), where $o_j = f_{jk}$

Sentence level: idem, but for (subjective) sentences, so these need to be identified first.

Feature level: documents and sentences may contain mixed opinions and analysis at this level does not identify specifically what people like/dislike. Steps:

- indentify and extract objective features
- determine whether the opinions on the features are positive, negative or neutral
- group synonym features
- Optional: produce a feature-based opinion sumary of multiple reviews(thus we can do comparesion between different targets)

Challenges

- o_j is a target object. : **Named Entities Recognition** (well-known tools based on gazetteers and simple context rules) / **Bootstrap from seed gazetteers**
- f_jk is a feature of the object o_j . : **Information Extraction**
- so_ijkl is the sentiment about f_jk : **Sentiment Determination**
- h_i : opinion holder : **Information (or metadata) Extraction**
- t_l . : **Information (or metadata) Extraction**

Addtion challenges

- **Co-reference resolution** : Is important to resolve objects and features 共指
- **Relation extraction** 关系提取
- **Synonym match** (“voice” = “sound quality”) 同义词

Sentiment analysis Approaches

Lexicon-based

Use a lexicon of opinion/emtion words, like: good, bad, horrible, great, etc.

Binary

Rule-based sentiment classifier (Sentence/ document level)

- Rule-based **subjectivity** classifier: a sentence/document is subjective if it has at least n(say 2) words from the emotion words lexicon. **主观句**：句子包含至少n(2)个来自emotion lexicon的词汇。
- Rule-based **sentiment** classifier: for sujective sentences / documents, count positive and negative words/phrases in the sentence/document. If more negative than positive words/phrases, then negative; otherwise, positive; if equal , neutral. **情感**：对于主观句，查数，判断积极消极中立。

Rule-based sentiment classifier (Feature-level)

- Assume features can be identified in previous step by information extracation techniques.
- For each feature,count positive. and negative words/phrases from the lexicon.
- than positive words/phrases, then negative; otherwise, positive; if equal, neutral.
- simple approach
  - Input: a pair (f, s), where f is a feature, and s is a sentence contains f.
  - Output: f is positive? Negative? Neutral?
  - Step1: work on the sentence s containing f
  - Step2: select emotion words in $s : w_1, w_2, ..., w_n$
  - Step3: assign orientations for these emotion words: 1=positive, -1 = negative, 0=neutral
  - Step4: sum up the orientation and assign the orientation to (f, s) accordingly.
- More advanced approaches split the sentence in parts, e.g. based on BUT words.

Caveats: 注意

Need more fined-grained sentiment information

- Context-independent
- Context-dependent
- Negation
- Intensifiers

Gradable

Use ranges of sentiment (to deal with intensifiers)

- like absolutely, utterly, totally, nearly, virtually, mainly, almost)

And grading adverbs

- (Very, little, dreadful, extremely, fairly, hugely, immensely, intensely, rather, reasonably, slightly, unusually)

Rule-based gradable sentiment classifiers

Classifiers general valence(value) of a text(document-,sentence- or feature-level) based on the level of emotional content

Level of emotional contents given by

- The **lexicon**: word-lists wirh pre-assigned emotional weights
- Additional general **rules** to the original weight, AS FOLLOWS:
    - **Negation rule** 否定词 : positive: -1, *-1; negative: +1, *-1
    - **Captalization rule** 大写: positive: +1; negative: -1 ;
    - **Intensifiers rule** 强调成分: intensifiers list; Each initensifiers has a weight; the weight is added to positive words/ sustracted from negative words
    - **Diminisher rule:** 减少成分: diminishers list; Weight; the weight is substracted to positive words/ added from negative words
    - **Exclamation rule** !!! 感叹号: Fuctions like intensifiers
    - **Emoticon rule** 小表情: Each has its own emotional weight, like an emtional word.

Final decision based on Allemotion words:

- if $|C_pos| > |C_neg|$ then $positive$
- if $|C_pos| < |C_neg|$ then $neagative$
- if $|C_pos| == |C_neg|$ then $neutral$

Advantages and Disadvantages of lexicon-based approaches

Advantages

- Works effectively with different texts: forums, blogs, etc.
- Language independent - as long as an up-to-date lexicon of emotion words is available
- Doesn't require data for training
- Can be extended with additional lexica, e.g. for new emotion words/symbols as they become popular, esp. in social media

Disadvantages:

- Requires a lexicon of emotion words, which should be fairly comprehensive, covering new words, abbreviations (LOL, m8, etc.), misspelled words, etc.

Get Lexica of emotion words/phrases

（*Adjectives, Verbs, Nouns, Phrases* ）

Task: Collect relavant words/ phrases that can be used to express sentiment.Determine the emotion.

- **Manully**: word lists with pre-assigned emotional weights
    - Created resources: SentiWordNet, Linguistic Inquiry and Word Count (LIWC) Lexicon, General Inquirer.
    - such as:
        - **SentiWordNet**: Wordnet is a database with words grouped into sets of synonyms (synsets), and organised by semantic relations between them: synonyms, antonyms, hypernyms, etc. SentiWordNet is a version of it with one of three

        sentiment scores for each synset: positivity, negativity, objectivity.
        - **Linguistic Inquiry and Word Count (LIWC) lexicon**: made by psychologists with lists of words with various emotional and other dimensions.
        - **General Inquirer**: terms with various types of positive or negative semantic orientation.
- **Semi-automaticaly**:
    - **Dictionary-based**: find synonyms/antonyms of seed emotion words in dictionaries like WordNet
    - **Corpus-based**: in corpora

Semi-automatically created from **seed words**:

- start with seed positive and negative words:
- **Search** for synonyms/ antonyms in
    - **dictionaries** like WordNet (dictionary-based)
    - or **Build patterns** from **seed** words/phrases tp search on large **corpora** (corpora-based)

Corpus-based (Supervised Machine Learning)

Idea: Mostly "**supervised learning**": **corpora** of examples **annotated with sentiment** are used with machine learning algorithms to learn a classifier for each sentence/document.

Get Corpora

- **Manually**: reliable, can be used as gold-standards
- From **crowd-annotated resources**, like Amazon Product Reviews (1-5 stars); Rotten Tomatoes, complaints.com, bitterlemons.com

Corpus

A collection of text segments + humanly-annotated emotional indicators (e.g. positive, negative, etc).

Examples of corpora:

- Subjectivity corpus
    - 10,000 sentences: subjective/objective
    - Objective: IMDB plot summaries
    - Subjective: Rotten Tomatoes website.
- "Movie Review" corpus (Pang, Lee and Vaithyanathan, 2002):
    - 2, 000 movie reviews (equal number of positive/negative)
    - Source: IMDB

Features

Mostly words, but also other linguistic traits describing positive/negative examples:

- Words (unigrams)
- n-grams (sequences of n words)

- Emotions from words/phrases extracted from dictionaries
- Part-of-speech (POS) tags
- Syntactic patterns (e.g. sequences of POS tags)
- Language model scores: similarities to positive/negative corpora
- Negations

All automatically extracted from the corpus

Steps

1. **Subjectivity** classifier: first run binary classifier to identify and then eliminate objective segments.
2. **Sentiment** classifier with remaining segments: learn how to combine and weight different attributes to make decisions. E.g. Naive Bayes

Pre-processing of corpus similar to IR

- Remove HTML or other tags
- Remove stopwords
- Perform word stemming/lemmatisation
- etc.

Naïve Bayes classifier

A supervised probabilistic model of the observed data. Can be used to predict the class label of new/unseen data

Models:

- Multi-variate Bernoulli Model: a document is a binary vector over the space of words.
- Multinomial Model: captures word frequency information in documents.

Supervised Classification: Rely on syntactic or co-occurrence patterns in large text corpora.

Naïve Bayes Classifier: estimate the probability of each class given a text:

- Compute the posterior probability (Bayes rule) of each class c i for text segment T $p(c_i|T) = \frac{P(T|c_i)P(c_i)}{P(T)}$
- Assumption of independence between features ("naive" assumption) $P(T|c_i) = P(t_1, t_2, \ldots t_j|c_i) \approx \prod_{j=1}^{n} P(t_j|c_i)$,

  where T is described by a number of attributes or features $t_1, \ldots, t$

  I.e. joint probability of the features given the class is approximated by the product of the probabilities of each feature given the class.
- **Likelihood**: product of probabilities of each feature value of segment occurring with class c $\prod_{j=1}^{n} P(t_j|c_i)$

  Add **smoothing** to feature counts (add 1 to every count). where |V | is the number of distinct attributes in training (all classes). $P(t_j|c_i) = \frac{count(t_j,c_i)+1}{count(c_i)+|V|}$
- **Prior**: probability of segment having class c_i: $P(c_i)$

- **Evidence**: product of probabilities of features of segment – constant term for all classes, so can be disregarded $\prod_{j=1}^{n} P(t_j)$

- **Final decision:**

  $argmax_{c_i} \prod_{j=1}^{n} P(t_j|c_i)P(c_i) = argmax_{c_i} P(c_i) \prod_{j=1}^{n} p(t_j|c_i)$

Given a trained classifier that classifies arbitrary segments of text we can use it to:

- Classify entire documents, e.g an entire review.
- Classify sentences in a document (perhaps just those identified as subjective) and then compute a classification of the document by aggregating the sentiments of individual sentences, according to some function.
- Classify sentences or phrases identified as discussing an aspect/feature of a target object (e.g. a sentence discussing battery life of a phone) and interpret the sentiment as the sentiment of opinion holder towards the specific aspect under discussion

灵魂拷问 Questions:

- Is this a good solution? Is it robust? It's simple and will work well if data is not sparse(数据够多就管用)

- What is the role of the prior? Prior is very important esp. on biased cases

- How can we improve this solution?

  - Other features? Are we missing out critical information?

    - Using all words (in Naive Bayes) works well in some tasks
    - Finding subsets of words may help in other tasks
    - Using only adjectives can be limiting. Verbs like hate, dislike; nouns like love; words for inversion like not; intensifiers like very
    - Pre-built polarity lexicons can be helpful
    - Negation is important
  - Other algorithms? MaxEnt & SVM tend to do better than Naive Bayes

- What about non-binary classification (e.g. 5-grades of sentiment)? 5-class ordinal classification or regression algorithms can be used

Comparative SA

Can contrast direct opinions versus more complex comparative opinions:

- Direct sentiment expressions on target objects

  e.g., "the picture quality of this camera is great."

- Comparisons expressing similarities or differences between objects, e.g., "car x is cheaper than car y."

Bing Liu distinguishes 4 types of **comparative relations**:

- **Gradable** Non-equal gradable: Relations of the type greater or less than. E.g.: "lenses of camera A are better than those of camera B"
- **Equative**: Relations of the type equal to. E.g.: "camera A and camera B both come in 7MP"
- **Superlative**: Relations of the type greater or less than all others. E.g.:"camera A is the

cheapest camera available in market"

- **Non-gradable comparisons**: Relations that compare aspects of two or more entities, but do not grade them. e.g., "Coke tastes differently from Pepsi."

Evaluation

- Create experimental datasets (aka test corpora): i.e., text segments that have been classified by humans, e.g. positive vs negative
- Compare (positive vs negative) system to human classifications
- Compute metrics like

  $Accuracy = \frac{\#\ correctly\ classified\ texts}{\#\ texts}$

  $Precision\ Pos = \frac{\#\ texts\ correctly\ classified\ as\ possible}{\#\ texts\ classified\ as\ possible}$

  $Recall\ Pos = \frac{\#\ texts\ correctly\ classified\ as\ possible}{\#\ positive\ texts}$

  $F\ measure = \frac{2 * Precision\ Pos * Recall\ Pos}{Precision\ Pos + Recall\ Pos}$

  Same for negative class.

  Baseline: most frequent class in the training set.

Conclusions

Naïve Bayes classifier:

- Really easy to implement and often works well (good solution, robust, if data is not sparse)
- Often a good first thing to try
- Actually, the IID is almost never true
- Still, NB often performs surprisingly well even its assumption does not hold.

SA is an exciting topic, many applications, huge market for systems, particularly in focused domains.

Promising results with simple techniques, but many interesting research challenges to be addressed for high accuracy.

Example exercise

Doc	Words	Class
1	**Great** movie, **excellent** plot, **renowned** actors	Positive
2	I had not seen a **fantastic** plot like this in **good** 5 years. **amazing !!!**	Positive
3	**Lovely** plot, **amazing** cast, somehow I am in love with the **bad** guy	Positive
4	**Bad** movie with **great** cast, but very **poor** plot and **unimaginative** ending	Negative
5	I hate this film, it has nothing **original**. Really **bad**	Negative
6	**Great** movie, but not...	Negative
7	Very **bad** movie, I have no words to express how I dislike it	Negative

Relative frequency in corpus is the simplest approach to estimating probabilities:

Priors: (where N = total training examples)

P(positive) = count(positive)/N = 3/7 = 0.43 P(negative) = count(negative)/N = 4/7 = 0.57

Assume standard pre-processing: tokenisation, lowercasing, punctuation removal (except special punctuation like !!!)

Likelihoods:

$P(t_j|c_i) = \frac{count(t_j,c_i)}{count(c_i)}$

Count word t_j in class c_i / total words in that class

$$
\begin{array}{ll|ll}
P(amazing|positive) & = 2/10 & P(amazing|negative) & = 0/8 \\
P(bad|positive) & = 1/10 & P(bad|negative) & = 3/8 \\
P(excellent|positive) & = 1/10 & P(excellent|negative) & = 0/8 \\
P(fantastic|positive) & = 1/10 & P(fantastic|negative) & = 0/8 \\
P(good|positive) & = 1/10 & P(good|negative) & = 0/8 \\
P(great|positive) & = 1/10 & P(great|negative) & = 2/8 \\
P(lovely|positive) & = 1/10 & P(lovely|negative) & = 0/8 \\
P(original|positive) & = 0/10 & P(original|negative) & = 1/8 \\
P(poor|positive) & = 0/10 & P(poor|negative) & = 1/8 \\
P(renowned|positive) & = 1/10 & P(renowned|negative) & = 0/8 \\
P(unimaginative|positive) & = 0/10 & P(unimaginative|negative) & = 1/8 \\
P(!!!|positive) & = 1/10 & P(!!!|negative) & = 0/8
\end{array}
$$

Relative frequencies for prior ($P(c_i)$) and likelihood ($P(t_j|c_i)$) make the model in a Naive Bayes classifier.

At decision (test) time, given a new segment to classify, this model is applied to find the most likely class for the segment:

Given a new segment to classify (**test time**):

Doc	Words	Class
8	This was a fantastic story, good, lovely	???

Final decision

$$\underset{c_i}{\operatorname{argmax}} P(c_i) \prod_{j=1}^{n} P(t_j|c_i)$$

$$P(positive) * P(fantastic|positive) * P(good|positive) * P(lovely|positive)$$

$$3/7 * 1/10 * 1/10 * 1/10 = 0.00043$$

$$P(negative) * P(fantastic|negative) * P(good|negative) * P(lovely|negative)$$

$$4/7 * 0/8 * 0/8 * 0/8 = 0$$

So: $sentiment = positive$

What if the new segment to classify (**test time**) is:

Doc	Words	Class
10	Lovely plot, excellent cast, amazing everything	???

Final decision

$$P(positive) * P(lovely|positive) * P(excellent|positive) * P(amazing|positive)$$

$$3/7 * 1/10 * 1/10 * 1/10 = 0.00043$$

$$P(negative) * P(lovely|negative) * P(excellent|negative) * P(amazing|negative)$$

$$4/7 * 0/8 * 0/8 * 0/8 = 0$$

So: $sentiment = positive$

Given a new segment to classify (**test time**):

Doc	Words	Class
9	Great plot, great cast, great everything	???

Final decision

$$P(positive) * P(great|positive) * P(great|positive) * P(great|positive)$$

$$3/7 * 1/10 * 1/10 * 1/10 = 0.00043$$

$$P(negative) * P(great|negative) * P(great|negative) * P(great|negative)$$

$$4/7 * 2/8 * 2/8 * 2/8 = 0.00893$$

So: $sentiment = negative$

But if the new segment to classify (**test time**) is:

Doc	Words	Class
11	Boring movie, annoying plot, unimaginative ending	???

Final decision

$$P(positive) * P(boring|positive) * P(annoying|positive) * P(unimaginative|positive)$$

$$3/7 * 0/10 * 0/10 * 0/10 = 0$$

$$P(negative) * P(boring|negative) * P(annoying|negative) * P(unimaginative|negative)$$

$$4/7 * 0/8 * 0/8 * 1/8 = 0$$

So: $sentiment = ???$

Add smoothing to feature counts (add 1 to every count). **Likelihoods** =

$$P(t_j|c_i) = \frac{count(t_j, c_i) + 1}{count(c_i) + |V|}$$

where $|V|$ is the number of distinct attributes in training (all classes) = **12**

Doc	Words	Class
12	Boring movie, annoying plot, unimaginative ending	???

Final decision

$$P(positive) * P(boring|positive) * P(annoying|positive) * P(unimaginative|positive)$$

$$3/7 * ((0+1)/(10+12)) * ((0+1)/(10+12)) * ((0+1)/(10+12)) = 0.000040$$

$$P(negative) * P(boring|negative) * P(annoying|negative) * P(unimaginative|negative)$$

$$4/7 * ((0+1)/(8+12)) * ((0+1)/(8+12)) * ((1+1)/(8+12)) = 0.000143$$

So: $sentiment = negative$