Sentiment Classification

Andrea Esuli

Sentiment Classification

Given a document *d*, a sentiment classification problem can be expressed in Bing Liu's model, as the problem of extracting from text an opinion **quintuple** of the form:

<_, GENERAL, s, _, _>

where _ marks an information that is assumed as **known** or **not relevant**, s is a **sentiment label**, e.g., positive or negative.

The **whole document** is considered as **a basic unit of information**.

The (strong, yet very common) assumption is that *d* expresses opinions only on the entity of interest and from a single opinion holder.

Supervised/unsupervised

Supervised learning methods are the most commonly used one, yet also some **unsupervised** methods have been successfully.

Unsupervised methods rely on the shared and recurrent characteristics of the sentiment dimension across topics to perform classification by means of hand-made heuristics and simple language models.

Supervised methods rely on a **training set** of labeled examples that describe the correct classification label to be assigned to a number of documents.
A learning algorithm then exploits the examples to model a general classification function.

Unsupervised methods

Unsupervised Sentiment Classification

Unsupervised methods do not require labeled examples.

Knowledge about the task is usually added by using lexical resources and hard-coded heuristics, e.g.:

- Lexicons + patterns: VADER
- Patterns + Simple language model: SO-PMI

Neural language models have been found that they learn to recognize sentiment with no explicit knowledge about the task.

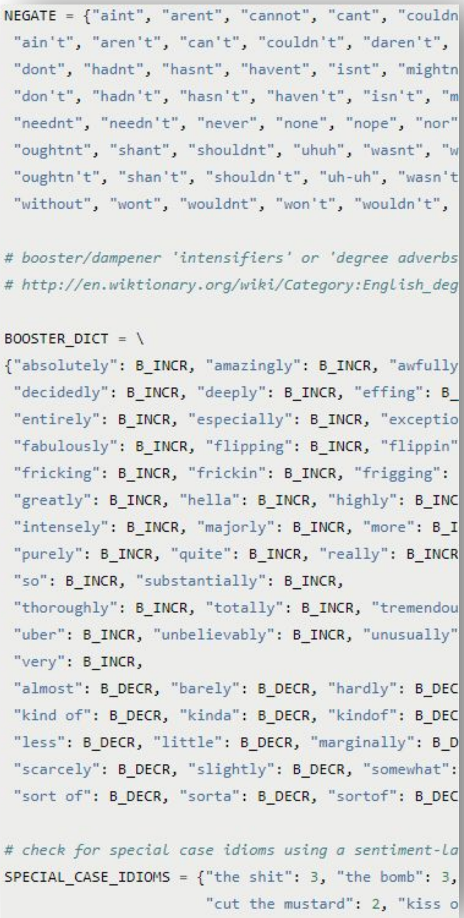
VADER

VADER (Valence Aware Dictionary for sEntiment Reasoning) uses a curated lexicon derived from well known sentiment lexicons that assigns a positivity/negativity score to 7k+ words/emoticons.

It also uses a number of hand-written pattern matching rules (e.g., negation, intensifiers) to modify the contribution of the original word scores to the overall sentiment of text.

[Hutto and Gilbert. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. ICWSM 2014.]()

VADER is integrated into [NLTK]()



VADER

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
vader = SentimentIntensityAnalyzer()

vader.polarity_scores('the best experience I had')
Out: {'neg': 0.0, 'neu': 0.417, 'pos': 0.583, 'compound': 0.6369}

vader.polarity_scores('not the best experience I had')
Out: {'neg': 0.457, 'neu': 0.543, 'pos': 0.0, 'compound': -0.5216}
```

VADER can be used to bootstrap a training set for *supervised learning.*

In this case we can talk of a *weakly-supervised* or *semi-supervised* approach, since training data are not all validated by a human, and can contain errors.

```
NEGATE = {"aint", "arent", "cannot", "cant", "couldn
 "ain't", "aren't", "can't", "couldn't", "daren't",
 "dont", "hadnt", "hasnt", "havent", "isnt", "mightn
 "don't", "hadn't", "hasn't", "haven't", "isn't", "m
 "neednt", "needn't", "never", "none", "nope", "nor"
 "oughtnt", "shant", "shouldnt", "uhuh", "wasnt", "w
 "oughtn't", "shan't", "shouldn't", "uh-uh", "wasn't
 "without", "wont", "wouldnt", "won't", "wouldn't",

# booster/dampener 'intensifiers' or 'degree adverbs
# http://en.wiktionary.org/wiki/Category:English_deg

BOOSTER_DICT = \
{"absolutely": B_INCR, "amazingly": B_INCR, "awfully
 "decidedly": B_INCR, "deeply": B_INCR, "effing": B_
 "entirely": B_INCR, "especially": B_INCR, "exceptio
 "fabulously": B_INCR, "flipping": B_INCR, "flippin"
 "fricking": B_INCR, "frickin": B_INCR, "frigging":
 "greatly": B_INCR, "hella": B_INCR, "highly": B_INC
 "intensely": B_INCR, "majorly": B_INCR, "more": B_I
 "purely": B_INCR, "quite": B_INCR, "really": B_INCR
 "so": B_INCR, "substantially": B_INCR,
 "thoroughly": B_INCR, "totally": B_INCR, "tremendou
 "uber": B_INCR, "unbelievably": B_INCR, "unusually"
 "very": B_INCR,
 "almost": B_DECR, "barely": B_DECR, "hardly": B_DEC
 "kind of": B_DECR, "kinda": B_DECR, "kindof": B_DEC
 "less": B_DECR, "little": B_DECR, "marginally": B_D
 "scarcely": B_DECR, "slightly": B_DECR, "somewhat":
 "sort of": B_DECR, "sorta": B_DECR, "sortof": B_DEC

# check for special case idioms using a sentiment-la
SPECIAL_CASE_IDIOMS = {"the shit": 3, "the bomb": 3,
                       "cut the mustard": 2, "kiss o
```

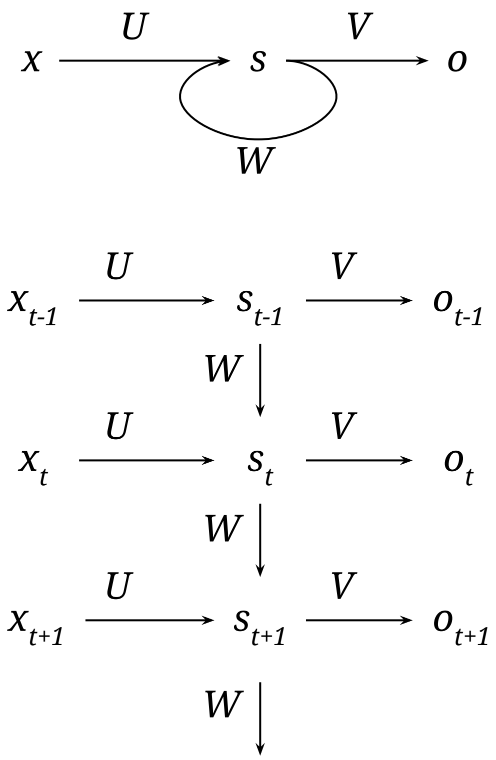
Thumbs Up or Thumbs Down?

Pointwise Mutual Information has been applied to determine the overall sentiment of text.

- Short phrases extracted from text using POS patterns, e.g.:
  `JJ+NN, RB+JJ, JJ+JJ, NN+JJ, RB+VB`
- SO-PMI score of each phrase is computed using a search engine and proximity queries, e.g.: `"very solid" NEAR good`
- SO-PMI scores for phrases are averaged to produce the document score.

[Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. ACL 2002]

Recurrent Neural Networks



A Recurrent Neural Network (RNN) is a neural network in which connections between units form a **directed cycle**.

Cycles allow the network to have a **memory** of previous inputs, combining it with current input.

RNNs are fit to process **sequences**, such as text.

Text can be seen as a sequence of values at many different levels: characters, words, phrases...

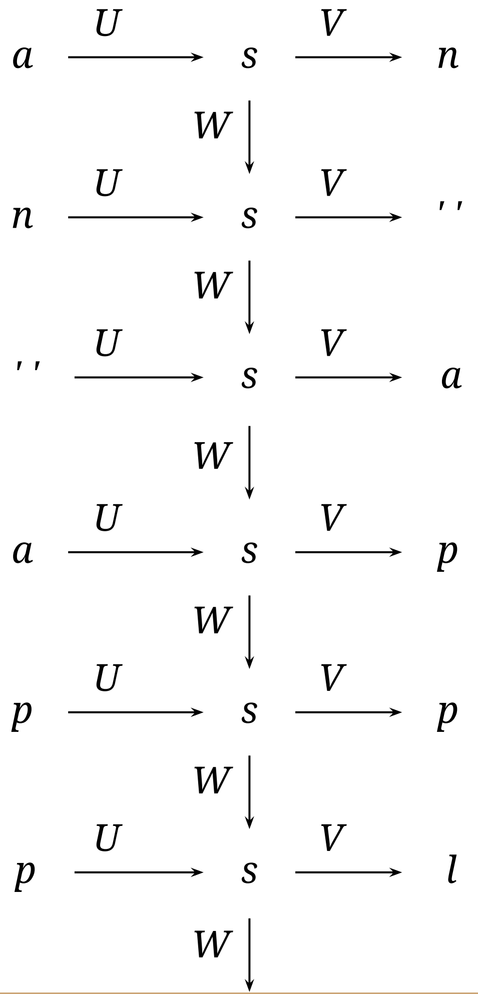
Suggested read

Char-level LM & text generation

RNNs are key tools in the implementation of many NLP applications, e.g., machine translation, summarization, or image captioning.

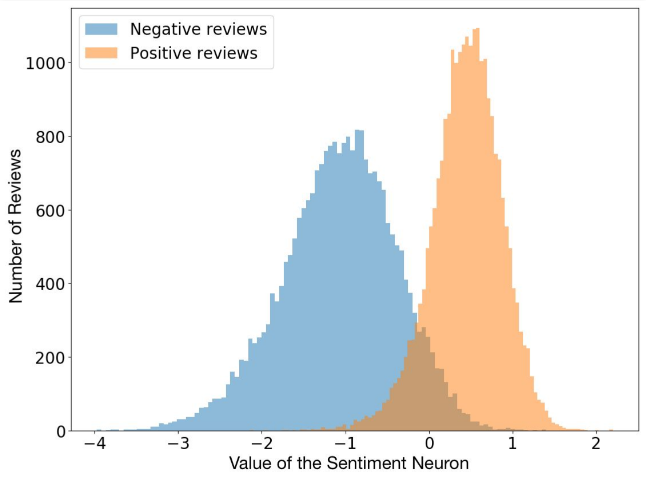
A RNN can be used to learn a language model that **predicts the next character from the sequence of previous ones**.

Let's build one! The RNN node we use is an Long Short Term Memory (LSTM), which is [robust to typical issues of RNNs].



Sentiment Classification from a single neuron

A char-level LSTM with 4096 units has been trained on **82 millions** of reviews from Amazon.



After training one of the units had a very high correlation with sentiment, resulting in state-of-the-art accuracy when used as a classifier.

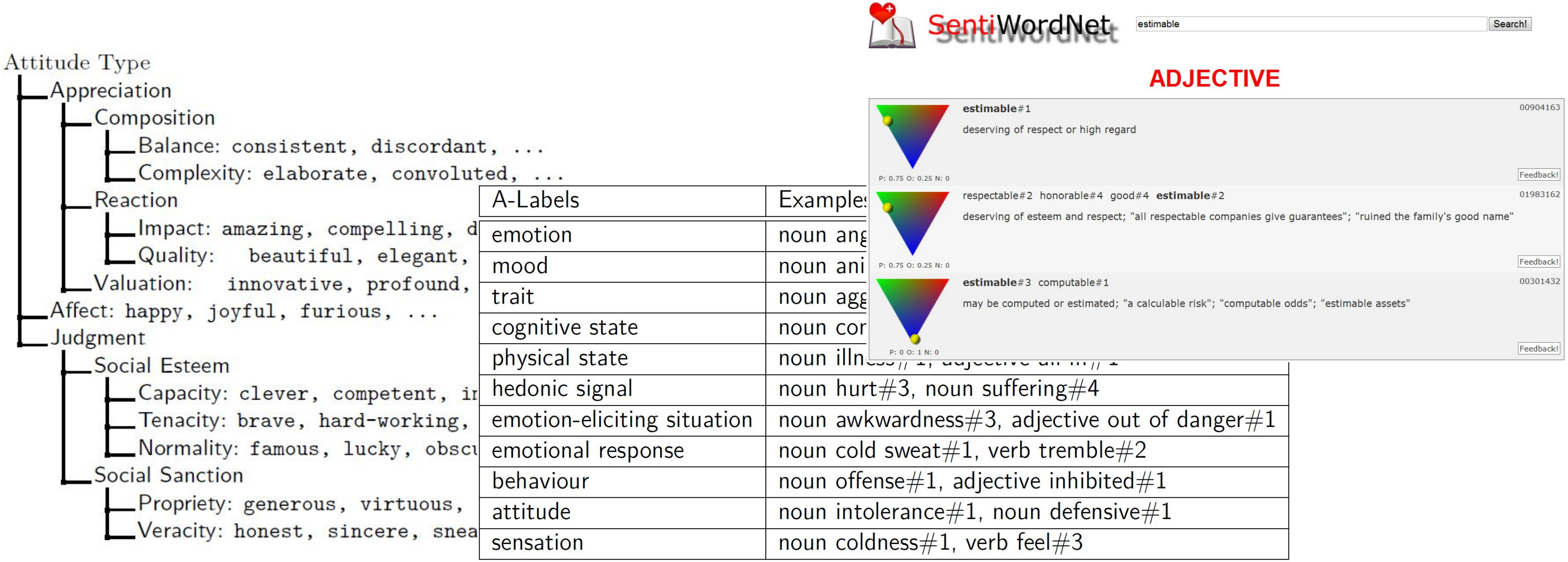
By fixing the sentiment unit to a given value, the generation process has been forced to produce reviews with a given sentiment polarity.

Blog post - Radford et al. Learning to Generate Reviews and Discovering Sentiment. Arxiv 1704.01444

Supervised methods

Supervised methods

Supervised methods use a traditional ML pipeline, typically exploiting the use of lexical resources to improve the number and quality of sentiment-related features extracted from text.



Supervised methods

Supervised classification methods follow a **learning by examples metaphor**, exploiting a **training set** of examples to learn a classification function.

- The correct classification of each document in the training is know.
- The training set is a representative sample of the domain on which classification takes place.

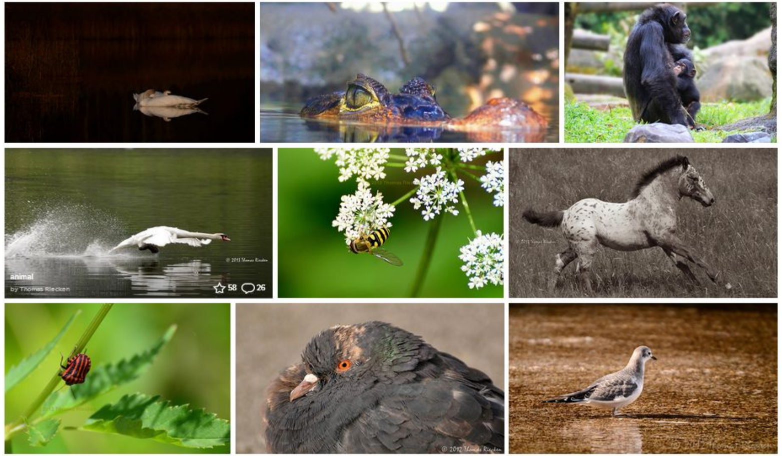
A learning algorithm observes the document-label pairs in the training set to determine a classification model $\Phi^*(d)$ that better approximates the true (unknown) classification function $\Phi(d)$ on the whole domain.

Supervised methods

A training set for a classifier of cat images.

$\Phi(i)$ = cat

$\Phi(i)$ = not cat



Supervised methods

Text is usually converted into a vectorial form through a **processing pipeline** that combines **NLP** (tokenization, POS tagging, lemmatization, parsing, lexical resources) and **IR** (feature selection, weighting).

- NNs can be built to directly work on sequences of word/characters.

The learned classification model depends on the specific learning algorithm:

- a probability distribution (Naïve Bayes),
- a hyperplane (SVM),
- a tree/a forest (decision trees),
- centroids (KNN),
- weights in matrices (neural networks), and so on...

The classification pipeline

The elements of a classification pipeline are:

1. Tokenization
2. Feature extraction
3. Feature selection
4. Weighting
5. Learning

Steps from 1 to 4 define the feature space and how text is converted into vectors.

Step 5 creates the classification model.

The classification pipeline

The scikit-learn library defines a rich number of data processing and machine learning algorithms.

Most modules in scikit implement a 'fit-transform' interface:

- fit method learns the parameter of the module from input data
- transform method apply the method implemented by the module to the data
- fit_transform does both actions in sequence, and is useful to connect modules in a pipeline.

Sentiment features

Sentiment lexicon can be exploited to add sentiment information in text representation.

In this way a general knowledge about language connects words that are observed in the training set with words that occur only in the test set (which would have been considered out-of-vocabulary words).

good → SWN_Pos

gentle → SWN_Pos

bad → SWN_Neg

hostile → SWN_Neg

Convolutional Neural Network

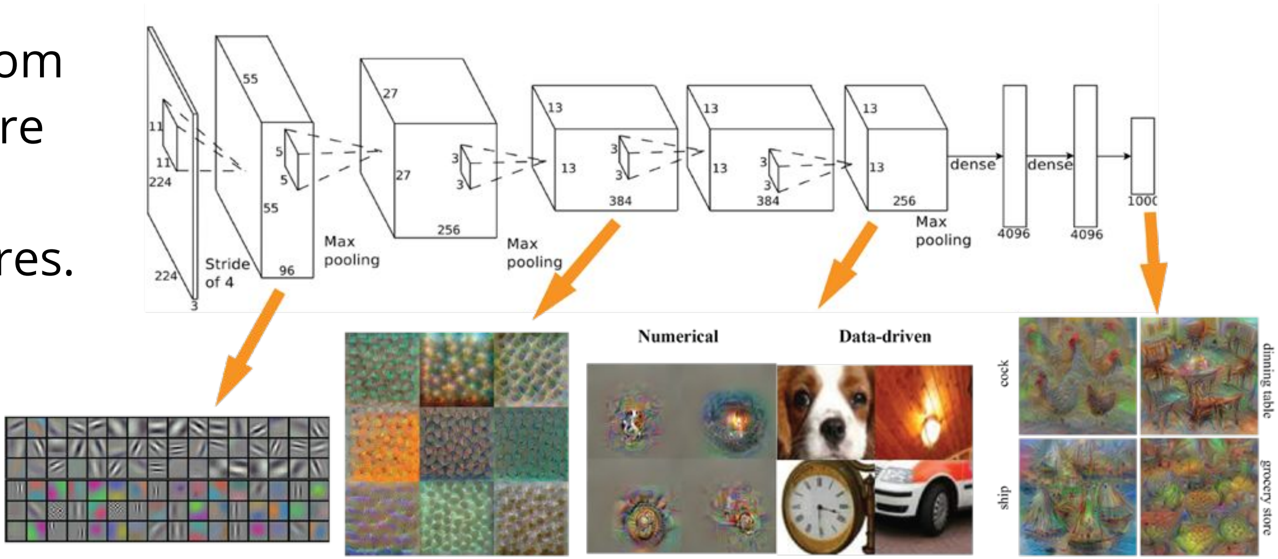
A convolutional layer in a NN is composed by a set of **filters**.

- A filter combines a "local" selection of input values into an output value.
- All filters are "sweeped" across all input.
  - A filter using a window length of 5 is applied to all the sequences of 5 words in a text.
  - 3 filters using a window of 5 applied to a text of 10 words produce 18 output values. Why?
  - Filters have additional parameters that define their behavior at the start/end of documents (padding), the size of the sweep step (stride), the eventual presence of holes in the filter window (dilation).
- During training each filter specializes into recognizing some kind of relevant combination of features.
- CNNs work well on stationary feats, i.e., those independent from position.

Convolutional Neural Network

CNNs have been successfully applied on images.

- First level of a stack of CNNs capture local pixel features (angles, lines)
- Successive layers combine features from lower levels into more complex, less local, more abstract features.



Conv 1: Edge+Blob Conv 3: Texture Conv 5: Object Parts Fc8: Object Classes

[image source]

Dropout, Pooling

A **dropout** layer hides output of random units from a layer to the next.

- It is a regularization technique that contrasts overfitting (i.e., being too accurate on training data and not learning to generalize).

A **pooling** layer aggregates (max, average) output of groups of units into a single value for the next layer.

- It reduces the number of parameters of the model (downsampling)
- It contrasts overfitting.
- It add robustness to local variations (translation)
- It can be used to reduce variable length inputs to the same length.

Recurrent Neural Network

A RNN learns to process a sequence and it incrementally builds an abstract representation of it.

By setting the last output of the network to fit on a classification label, the RNN learns a classifier.

Output can have many different forms, e.g.:

- another sequence (seq2seq): translation, summarization, text2speech, speech2text
- an image
- ...or you can have an image as input and an RNN generates a caption.

RNNs model a more natural way to process text over CNNs.

Distant supervision

Producing training data for supervised learning may have a relevant cost.

Distant supervision exploits "cheap" methods that "weakly" label examples to bootstrap a training set, e.g.:

- labeling tweets with 😊 as positive and those with 😒 as negative.
- using VADER to perform a first labeling (skipping low confidence labels).

The rationale behind distant supervision is that:

- noisy information in training data will cancel out in the learning phase.
- discriminant features that have a decent correlation with the weak labeling emerge among the other.

Distant supervision likes sentiment

Distant supervision fits better with sentiment analysis than with topic-related analysis because in the former it is easier to define negative examples.

A negative sentiment is a concept on its own, opposite to a positive one.

The "negation" of a topic is just the absence of the topic. It is harder to define a heuristic to label negative docs.

- How to automatically mark a negative example for a "soccer" classifier?
- Just use random sampling when nothing else works.