An Introduction To Natural Language Processing

Chapter 4: Applications of Text Classification

Jacob Eisenstein

Roadmap for this chapter

- Classical applications of text classification
  - Sentiment and opinion analysis
  - Word sense disambiguation
- Design decisions in text classification
- Evaluation

Sentiment analysis

The **sentiment** expressed in a text refers to the author's subjective or emotional attitude towards the central topic of the text:

- In consumer reviews, the sentiment is targeted at a product or service, and may align with a 1-5 star rating.[1]
- In political statements, the sentiment may reflect a favorable or unfavorable view toward a proposed policy.[2]

Sentiment analysis is a classical application of text classification, and is typically approached with a bag-of-words classifier.

[1] Pang, Lee, and Vaithyanathan 2002.
[2] Thomas, Pang, and Lee 2006.

Beyond the bag-of-words

Some linguistic phenomena require going beyond the bag-of-words:

(1)
a. That's not bad for the first day.
b. This is not the worst thing that can happen.
c. It would be nice if you acted like you understood.
d. This film should be brilliant. The actors are first grade. Stallone plays a happy, wonderful man. His sweet wife is beautiful and adores him. He has a fascinating gift for living life fully. It sounds like a great plot, **however**, the film is a failure.[3]

How would you handle these cases?

[3] Pang, Lee, and Vaithyanathan 2002.

Related classification problems

Subjectivity Does the text convey factual or subjective content?

Stance classification Given a set of possible positions, or **stances**, which is being taken by the author?

Targeted sentiment analysis What is the author's attitude towards several different entities?

(2) The vodka was good, but the meat was rotten.

Emotion classification Given a set of possible emotional states, which are expressed by the text?

These problems have many applications, including both commercial products as well as research in the digital humanities and computational social sciences.[4]

[4] e.g., Jockers 2015; Miller et al. 2011.

Word sense disambiguation

Consider the the following headlines:

(3) a. Iraqi **head** seeks arms
    b. Prostitutes **appeal** to Pope
    c. Drunk gets nine years in violin **case**[5]

[5]These examples, and many more, can be found at
`http://www.ling.upenn.edu/~beatrice/humor/headlines.html`

Word senses

Many words have multiple **senses**, or meanings. For example, the verb *appeal* has the following senses:

$appeal^1$	take a court case to a higher court for review
$appeal^2$, $invoke$	request earnestly (something from somebody)
$attract$, $appeal^3$	be attractive to

- **Word senses disambiguation** is the problem of identifying the intended word sense in a given context.
- More formally, senses are properties of **lemmas** (uninflected word forms), and are grouped into **synsets** (synonym sets). These synsets are collected in WORDNET.[6]

[6] e.g., `http://wordnetweb.princeton.edu/perl/webwn?s=appeal`

Word sense disambiguation as classification

How can we tell living $plants$ from manufacturing $plants$? Context.

(4) a. Town officials are hoping to attract new manufacturing plants through weakened environmental regulations.

b. The endangered plants play an important role in the local ecosystem.

Word sense disambiguation as classification

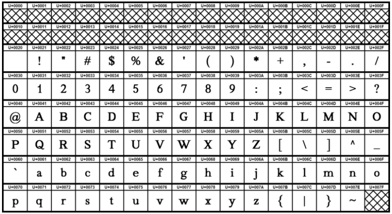
How can we tell living *plants* from manufacturing *plants*? Context.

(4) a. Town officials are hoping to attract new manufacturing plants through weakened environmental regulations.
b. The endangered plants play an important role in the local ecosystem.

$$f((plant, The\ endangered\ plants\ play\ an\ \ldots), y) =$$
$$\{(the, y) : 1, (endangered, y) : 1, (play, y) : 1, (an, y) : 1, \ldots\}$$

Applying text classification

- The "raw" form of text is usually a sequence of characters, or more generally, unicode code points.
- Converting this into a meaningful feature vector $\boldsymbol{x}$ requires a series of design decisions, such as tokenization, normalization, and filtering.



[6]https://commons.wikimedia.org/wiki/File:UCB_Basic_Latin.png
https://commons.wikimedia.org/wiki/File:UCB_Kannada.png

Tokenization

- **Tokenization** is the task of splitting the input into discrete tokens.
- This may seem easy for Latin script languages like English, but there are some tricky parts. How many tokens do you see in this example?

  (5) O'Neill's prize-winning pit bull isn't really a "bull".

  How would you separate these tokens?

Four English tokenizers

Input: *Isn't Ahab, Ahab? ;)*

Whitespace	Isn't	Ahab,	Ahab?	;)					
Treebank	Is	n't	Ahab	,	Ahab	?	;	)	
Tweet	Isn't	Ahab	,	Ahab	?	;)			
TokTok	Isn	'	t	Ahab	,	Ahab	?	;	)

Tokenization in other scripts

- Some languages are written in scripts that do not include whitespace. Chinese is a prominent example.
- Tokenization can usually be solved by matching character sequences against a dictionary, but some sequences have multiple possible segmentations:[7]

(1) 日文 章魚 怎麼 說?
Japanese octopus how say
How to say octopus in Japanese?

(2) 日 文章 魚 怎麼 說?
Japan essay fish how say

[7] Sproat et al. 1996.

Normalization

Distinctions with a difference?

- *apple* vs *apples*
- *apple* vs *Apple*
- *1,000* vs *1000* vs *one thousand*
- *sooooooooo* vs *so*
- *Aug 11* vs *August 11* vs *8/11* vs *11 August* ...

Normalization

Distinctions with a difference?

- *apple* vs *apples*
- *apple* vs *Apple*
- *1,000* vs *1000* vs *one thousand*
- *sooooooooo* vs *so*
- *Aug 11* vs *August 11* vs *8/11* vs *11 August* ...

More aggressive ways to group words:

- **Stemming**: removing inflectional affixes, $whales \rightarrow whale$.
- **Lemmatization**: converting to a base form, $geese \rightarrow goose$.

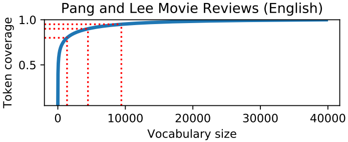
Three English stemmers

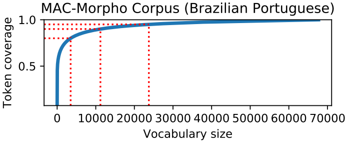
Original	The	Williams	sisters	are	leaving	this	tennis	centre
Porter	the	william	sister	are	leav	thi	tenni	centr
Lancaster	the	william	sist	ar	leav	thi	ten	cent
WordNet	The	Williams	sister	are	leaving	this	tennis	centre

- The WordNet system is a lemmatizer, and incorporates word-specific rules.
- Stemming and lemmatization rarely help supervised classification, but can be useful for string matching and unsupervised learning (chapter 5).

Vocabulary size filtering

A small number of word **types** accounts for the overwhelming majority of word **tokens**:





- The number of parameters in a classifier usually grows linearly with the size of the vocabulary.
- It can be useful to limit the vocabulary, e.g., to word types appearing at least x times, or in at least y% of documents.

Evaluating your classifier

Goal is to predict **future** performance, on unseen data.

- It is hard to predict the future.
- Do not evaluate on data that was already used ...
  - for training;
  - for hyperparameter selection;
  - for selecting the classification model or model structure;
  - for making preprocessing decisions, such as vocabulary selection.

Evaluating your classifier

Goal is to predict **future** performance, on unseen data.

- It is hard to predict the future.
- Do not evaluate on data that was already used ...
  - for training;
  - for hyperparameter selection;
  - for selecting the classification model or model structure;
  - for making preprocessing decisions, such as vocabulary selection.
- Even if you follow all these rules, you will still probably overestimate your classifier's performance, because real future data will differ from your test set in ways that you cannot anticipate.

Accuracy

Most basic metric is **accuracy**: how often is the classifier right?

$$\mathrm{acc}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N} \sum_{i=1}^{N} \delta(y^{(i)} = \hat{y}).$$

[8] Bergsma et al. 2012.

Accuracy

Most basic metric is **accuracy**: how often is the classifier right?

$$\mathrm{acc}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{i=1}^{N} \delta(y^{(i)} = \hat{y}).$$

The problem with accuracy is **rare labels**.

- Consider a system for detecting tweets written in Telugu.
- 0.3% of Tweets are written in Telugu.[8]
- A system that says $\hat{y} = \textsc{NotTelugu}$ is 99.7% accurate.

[8] Bergsma et al. 2012.

Beyond right and wrong

For any label, there are two ways to be wrong:

- **False positive**: the system incorrectly predicts the label.
- **False negative**: the system incorrectly fails to predict the label.

Similarly, there are two ways to be right:

- **True positive**: the system correctly predicts the label.
- **True negative**: the system correctly predicts that the label does not apply to this instance.

Recall and precision are defined in terms of these counts, and distinguish between the two types of errors.

Recall and precision

Recall is $r = \frac{TP}{TP+FN}$.

- Recall is the fraction of positive instances which were correctly classified.
- The "never Telugu" classifier has zero recall.
- An "always Telugu" classifier would have perfect recall.

Recall and precision

Recall is $r = \frac{TP}{TP+FN}$.

- Recall is the fraction of positive instances which were correctly classified.
- The "never Telugu" classifier has zero recall.
- An "always Telugu" classifier would have perfect recall.

Precision is $p = \frac{TP}{TP+FP}$.

- Precision is the fraction of positive *predictions* that were correct.
- The "never Telugu" classifier has precision $\frac{0}{0}$.
- An "always Telugu" classifier would have precision $p = 0.003$, which is the rate of Telugu tweets in the dataset.

Combining recall and precision

- In binary classification, there is an inherent tradeoff between recall and precision.
- The correct navigation of this tradeoff is problem-specific!
  - For a preliminary medical diagnosis, we might prefer high recall. False positives can be screened out later.
  - The "beyond a reasonable doubt" standard of U.S. criminal law implies a preference for high precision.
- If recall and precision are weighted equally, they can be combined into a single number called the F-measure:

$$F = \frac{2 \times r \times p}{r + p}. \tag{1}$$

Evaluating multi-class classification

- Recall and precision imply binary classification: each instance is either positive or negative.
- In multi-class classification, each instance is positive for one class, and negative for all other classes.
- There are two ways to combine performance across classes:
  - **Macro F-measure**: compute the F-measure per class, and average across all classes. This treats all classes equally, regardless of their frequency.
  - **Micro F-measure**: compute the total number of true positives, false positives, and false negatives across all classes, and compute a single F-measure. This emphasizes performance on high-frequency classes.

Comparing classifiers

Suppose two teams build classifiers to solve a problem:

- C_1 gets 82% accuracy
- C_2 gets 73% accuracy

Remember that we are interested in **future** performance.
Will C_1 be more accurate in the future?

Comparing classifiers

Suppose two teams build classifiers to solve a problem:

- C_1 gets 82% accuracy
- C_2 gets 73% accuracy

Remember that we are interested in **future** performance.
Will C_1 be more accurate in the future?

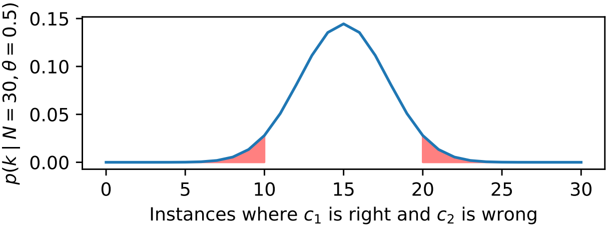
- What if the test set had 1000 examples?
- What if the test set had 11 examples?

Hypothesis testing

- Consider two hypotheses that explain the observed data:
  - H_1: C_1 is more accurate than C_2, and therefore can be expected to be more accurate in the future (in the limit of an infinite number of independent evaluations).
  - H_0: C_1 is not more accurate than C_2, and its superior performance on the test set was due only to luck. This is the **null hypothesis**.
- If the test set is small, H_0 might be true.
- If the test set is large, the probability of observing a 9% difference in accuracy becomes vanishingly small unless C_1 really is more accurate.
- These probabilities are quantified by hypothesis testing.

The binomial test

- If the two classifiers are equally accurate, then each time they disagree, they are equally likely to be correct.
- Over 30 such disagreements, each classifier will "win" roughly half the time:



- The total probability mass in the pink region is less than 5%. If the data is in this region, we reject the null hypothesis (of equal accuracy) with $p < .05$.

Other hypothesis tests

- The binomial test compares two classifiers in terms of accuracy. It can be computed in closed form using a numerical computing package such as SCIPY or R.
- Hypotheses about other metrics, such as F-measure, cannot be tested in this way.
- For these hypotheses, the best approach is randomization: randomly sample many test sets, and count how often each hypothesis holds.

Getting labels

Text classification relies on large datasets of labeled examples. There are two main ways to get labels:

- **Metadata** sometimes tell us exactly what we want to know: Did the Senator vote for the bill? How many stars did the reviewer give? Was the request for free pizza accepted?[9]
- Other times, the labels must be **annotated**, either by experts or by novice "crowdworkers."

[9] **althoff2014ask**.

Validating annotations

Annotations are validated by computing **inter-annotator agreement**.

- How likely are two annotators to choose the same label for an instance?
- How likely would this be if their labels were randomly shuffled?

Computing inter-annotater agreement

Example: Asha and Boris rate jokes as "funny" or "not funny":

	A = funny	A = not funny
B = funny	70	20
B = not funny	5	5

- Observed agreement: 75%
- Chance agreement:

Computing inter-annotater agreement

Example: Asha and Boris rate jokes as “funny” or “not funny”:

	A = funny	A = not funny
B = funny	70	20
B = not funny	5	5

- Observed agreement: 75%
- Chance agreement:

$$\Pr(\text{agree}) = \frac{70+20}{100} \times \frac{70+5}{100} + \frac{5+5}{100} \times \frac{20+5}{100}$$

Computing inter-annotater agreement

Example: Asha and Boris rate jokes as "funny" or "not funny":

	A = funny	A = not funny
B = funny	70	20
B = not funny	5	5

- Observed agreement: 75%
- Chance agreement:

$$\Pr(\text{agree}) = \frac{70+20}{100} \times \frac{70+5}{100} + \frac{5+5}{100} \times \frac{20+5}{100}$$

Computing inter-annotater agreement

Example: Asha and Boris rate jokes as “funny” or “not funny”:

	A = funny	A = not funny
B = funny	70	20
B = not funny	5	5

- Observed agreement: 75%
- Chance agreement:

$$\Pr(\text{agree}) = \frac{70+20}{100} \times \frac{70+5}{100} + \frac{5+5}{100} \times \frac{20+5}{100}$$

Computing inter-annotater agreement

Example: Asha and Boris rate jokes as "funny" or "not funny":

	A = funny	A = not funny
B = funny	70	20
B = not funny	5	5

- Observed agreement: 75%
- Chance agreement:

$$\Pr(\text{agree}) = \frac{70+20}{100} \times \frac{70+5}{100} + \frac{5+5}{100} \times \frac{20+5}{100}$$

Computing inter-annotater agreement

Example: Asha and Boris rate jokes as "funny" or "not funny":

	A = funny	A = not funny
B = funny	70	20
B = not funny	5	5

- Observed agreement: 75%
- Chance agreement:

$$\Pr(\text{agree}) = \frac{70+20}{100} \times \frac{70+5}{100} + \frac{5+5}{100} \times \frac{20+5}{100}$$

Computing inter-annotater agreement

Example: Asha and Boris rate jokes as "funny" or "not funny":

	A = funny	A = not funny
B = funny	70	20
B = not funny	5	5

- Observed agreement: 75%
- Chance agreement:

$$\begin{aligned}\Pr(\text{agree}) =& \frac{70+20}{100} \times \frac{70+5}{100} + \frac{5+5}{100} \times \frac{20+5}{100}\\ =& 27/40 + 1/40 = 70\%\end{aligned}$$

Roadmap for this chapter

- Classical applications of text classification
  - Sentiment and opinion analysis
  - Word sense disambiguation
- Design decisions in text classification
- Evaluation

References I

- Bergsma, Shane et al. (June 2012). "Language Identification for Creating Language-Specific Twitter Collections". In: *Proceedings of the Second Workshop on Language in Social Media*. Montréal, Canada: Association for Computational Linguistics, pp. 65–74.
- Jockers, Matthew L. (2015). *Revealing Sentiment and Plot Arcs with the Syuzhet Package*. `http://www.matthewjockers.net/2015/02/02/syuzhet/`.
- Miller, Mahalia et al. (2011). "Sentiment Flow Through Hyperlink Networks". In: *Proceedings of the International Conference on Web and Social Media (ICWSM)*.
- Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan (2002). "Thumbs up?: sentiment classification using machine learning techniques". In: *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pp. 79–86.

References II

- Sproat, Richard et al. (1996). "A stochastic finite-state word-segmentation algorithm for Chinese". In: *Computational linguistics* 22.3, pp. 377–404.
- Thomas, Matt, Bo Pang, and Lillian Lee (2006). "Get Out The Vote: Determining Support Or Opposition From Congressional Floor-Debate Transcripts". In: *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pp. 327–335.