Linguistic classification (INLP ch. 4)

CS 685, Spring 2021
Advanced Topics in Natural Language Processing
http://brenocon.com/cs685
https://people.cs.umass.edu/~brenocon/cs685_s21/

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

Today

- Examples of cool language tasks
  - Sentiment
  - WSD
- Preprocessing and linguistic design decisions
- Agreement rates and annotation

Sentiment

- Often conceived of as *polarity*:
  negative, neutral, positive
  - Dislike/like, love/hate …
- Do you believe sentiment analysis?
  - Overall sentiment of a tweet
  - Stars in a review
- Targeted sentiment analysis:
  - author's attitude
  - toward a particular concept (often, *word* in the text)
- Many, many variants: affective analysis, opinion analysis, etc.

Word senses

(4.3) a. Iraqi head seeks arms

b. Prostitutes appeal to Pope

c. Drunk gets nine years in violin case[2]

- Supervised WSD
  - Use features/embeddings from neighboring contextual words
- Is supervised WSD a realistic task?

Ling. preproc. decisions

- To define the symbolic units for either features or to have neural embeddings, we must preprocess (e.g. tokenize) the text somehow
- Preprocessing decisions encode linguistic assumptions!
- e.g. **What is a word?**
- Example
  - Tokenize tweets by splitting text on regex ***[^a-zA-Z0-9]+***
  - => Among top-100 most common words
    - p
    - d
- Why? *[Owoputi et al. 2013, section 4]*

Tokenizers

Whitespace	Isn't	Ahab,	Ahab?	;)				
Treebank	Is	n't	Ahab	,	Ahab	?	;	)
Tweet	Isn't	Ahab	,	Ahab	?	;)		
TokTok (Dehdari, 2014)	Isn	'	t	Ahab	,	Ahab	?	;	)

Figure 4.1: The output of four NLTK tokenizers, applied to the string *Isn't Ahab, Ahab? ;)*

Word normalization

- Case normalization (even that can be lossy)
- Stemmers and lemmatizers: delete inflectional affixes
  - Language specific!
  - “Stemmers”: crude affix analyzers.
  - “Lemmatizers”: trying to be smarter (more linguistically motivated).
    - High quality lemmatization requires part-of-speech category — requires contextual disambiguation!
    - More generally: morphological analysis

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

Figure 4.2: Sample outputs of the Porter (1980) and Lancaster (Paice, 1990) stemmers, and the WORDNET lemmatizer

N-grams

- Word n-grams: all (often overlapping) subsequences of length n
  - Vary n: trade off coarse/generalizable vs. specific/sparse
  - How big can you make n?
- For features, typically use progressively larger n-grams at once
  - E.g. “up to 3-grams”: all 1-grams, and 2-grams, and 3-grams
  - Option: Filter to grammatical phrases (e.g. POS patterns)? Depends on data volume
- *Character* n-grams often work really well
  - As word-internal features
  - As alternative to word n-grams when word segmentation is hard/wasteful (e.g. CJK, social media hashtag compounds, …)
    - If you make ’n’ as high as the average word length in the language, is this better or worse than having using word unigrams?

General preproc tradeoff

- For many preproc or feature decisions, a general tradeoff:
  1. Overproduce fine-grained terms/features with minimal normalization or filtering. Possibly highly redundant.
  2. Only produce a highly selective set of very normalized terms/features.
- Supervised learning with lots of labeled data: (1) tends to be better
- Low amounts of data and/or unsupervised learning: (2) tends to be better

Where to get labels?

- Natural annotations
  - Metadata - information associated with text document, but not in text itself
  - Clever patterns from text itself
- New human annotations
  - Yourself
  - "Friends & family"
  - Hire people locally
  - Hire people online
    - Mechanical Turk — most commonly used crowdsourcing site
    - (For larger/more expensive tasks: Upwork/ODesk)

Proceedings of the Ninth International AAAI Conference on Web and Social Media

Contextualized Sarcasm Detection on Twitter

David Bamman and Noah A. Smith
School of Computer Science
Carnegie Mellon University
{dbamman,nasmith}@cs.cmu.edu

Abstract

Sarcasm requires some shared knowledge between speaker and audience; it is a profoundly *contextual* phenomenon. Most computational approaches to sarcasm detection, however, treat it as a purely linguistic matter, using information such as lexical cues and their corresponding sentiment as predictive features. We show that by including extra-linguistic information from the context of an utterance on Twitter – such as properties of the author, the audience and the immediate communicative environment – we are able to achieve gains in accuracy compared to purely linguistic features in the detection of this complex phenomenon, while also shedding light on features of interpersonal interaction that enable sarcasm in conversation.

people who know each other well than do not.

In all of these cases, the relationsl and audience is central for understandi nomenon. While the notion of an "auc well defined for face-to-face conversa people, it becomes more complex wh are present (Bell 1984), and especially when a user's "audience" is often unkn or "collapsed" (boyd 2008; Marwick an ing it difficult to fully establish the sha for sarcasm to be detected, and underst (or imagined) audience.

We present here a series of experime fect of extra-linguistic information on

A Large Self-Annotated Corpus for Sarcasm

Mikhail Khodak and **Nikunj Saunshi** and **Kiran Vodrahalli**
Computer Science Department, Princeton University
35 Olden St., Princeton, New Jersey 08540
{mkhodak,nsaunshi,knv}@cs.princeton.edu

Abstract

We introduce the Self-Annotated Reddit Corpus (**SARC**)[1], a large corpus for sarcasm research and for training and evaluating systems for sarcasm detection. The corpus has 1.3 million sarcastic statements — 10 times more than any previous dataset — and many times more instances of non-sarcastic statements, allowing for learning in regimes of both balanced and unbalanced labels. Each statement is furthermore *self-annotated* — sarcasm is labeled by the author and not an independent annotator — and provided with user, topic, and conversation context. We evaluate the corpus for accuracy, compare it to previous related corpora, and provide baselines for the task of sarcasm detection.

1 Introduction

Sarcasm detection is an important component of many natural language processing (NLP) systems, with direct relevance to natural language understanding, dialogue systems, and text mining. However, detecting sarcasm is difficult because it occurs infrequently and is difficult for even human annotators to discern (Wallace et al., 2014). Despite these properties, existing datasets self-annotated labels and does not consist of low-quality text snippets from Twitter[2]. With more than a million examples of sarcastic statements, each provided with author, topic, and contex information, the dataset also exceeds all previous sarcasm corpora by an order of magnitude. This dataset is possible due to the comment structure of the social media site Reddit[3] as well its frequently-used and standardized annotation for sarcasm.

Following a discussion of corpus construction and relevant statistics, in Section 4 we present results of a manual evaluation on a subsample of the data as well as a direct comparison with alternative sources. Then in Section 5 we examine simple methods of detecting sarcasm on both a balanced and unbalanced version of our dataset.

2 Related Work

Since our main contribution is a corpus and not a method for sarcasm detection, we point the reader to a recent survey by Joshi et al. (2016) that discusses many interesting efforts in this area. Note that many of the works the authors mention will be discussed by us in this section, with many papers using their own datasets; this illustrates the need for common baselines for evaluation.

Sarcasm datasets can largely be distinguished by the sources used to get sarcastic and non-sarcastic statements, the amount of human anno-

Where to get labels?

- Natural annotations
  - Metadata - information associated with text document, but not in text itself
  - Clever patterns from text itself
- New human annotations
  - Yourself
  - Your friends
  - Hire people locally
  - Hire people online
    - Mechanical Turk — most commonly used crowdsourcing site
    - (For larger/more expensive tasks: Upwork/ODesk)

Secure https://www.mturk.com/mturk/welcome

amazon mechanical turk beta
Artificial Artificial Intelligence

Your Account | HITs | Qualifications

Already have an account?
Sign in as a **Worker** | **Requester**

Introduction | **Dashboard** | **Status** | **Account Settings**

Mechanical Turk is a marketplace for work.

We give businesses and developers access to an on-demand, scalable workforce.
Workers select from thousands of tasks and work whenever it's convenient.

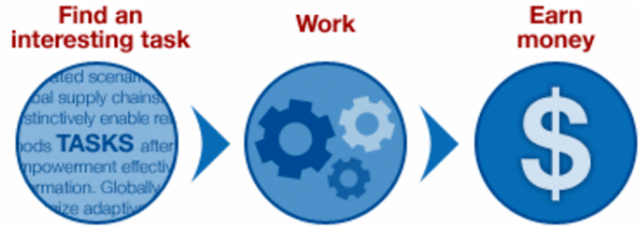
247,056 HITs available. View them now.

Make Money
by working on HITs

HITs - *Human Intelligence Tasks* - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:

- Can work from home
- Choose your own work hours
- Get paid for doing good work



Get Results
from Mechanical Turk Workers

Ask workers to complete HITs - *Human Intelligence Tasks* - and get results using Mechanical Turk. Get Started.

As a Mechanical Turk Requester you:

- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you're satisfied with the results

Annotation process

- To pilot a new task, requires an iterative process
  - Look at data to see what's possible
  - Conceptualize the task, try it yourself
  - Write annotation guidelines
  - Have annotators try to do it. Where do they disagree? What feedback do they have?
  - Revise guidelines and repeat
- If you don't do this, your labeled data will have lots of unclear, arbitrary, and implicit decisions inside of it

- stopped here on 2/15

Annotation is paramount

- Supervised learning is the most reliably successful approach to NLP and artificial intelligence more generally.
- Alternative view: it's *human* intelligence through the human-supplied training labels, that's at the heart of it. Supervised NLP merely extends a noisier, less-accurate version to more data.
- If we still want it: we need a plan to get good annotations!

Annotation process

- To pilot a new task, requires an iterative process
  - Look at data to see what's possible
  - Conceptualize the task, try it yourself
  - Write annotation guidelines
  - Have annotators try to do it. Where do they disagree? What feedback do they have?
  - Revise guidelines, repeat, resolve disagreements
- If you don't do this, your labeled data will have lots of unclear, arbitrary, and implicit decisions inside of it
- (Also alternative processes, e.g. active learning: simultaneously do annotation and model training. But you still need smart human design & intervention!)

Interannotator agreement

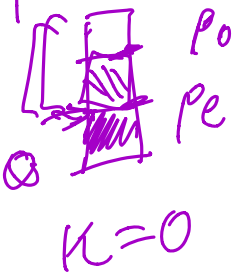
- How "real" is a task? Replicable? Reliability of annotations?
- How much do two humans *agree* on labels?
  - Difficulty of task. Human training? Human motivation/effort?
  - Goal: get the human performance upper bound
- If some classes predominate, raw agreement rate may be misleading
  - Chance-adjusted agreement: Cohen kappa for a pair of human annotators (see also Fleiss kappa, Krippendorff alpha…)

Cohen's kappa

p_o: observed agreement rate

p_e: agreement rate by chance

$$\frac{p_o - p_e}{1 - p_e}$$



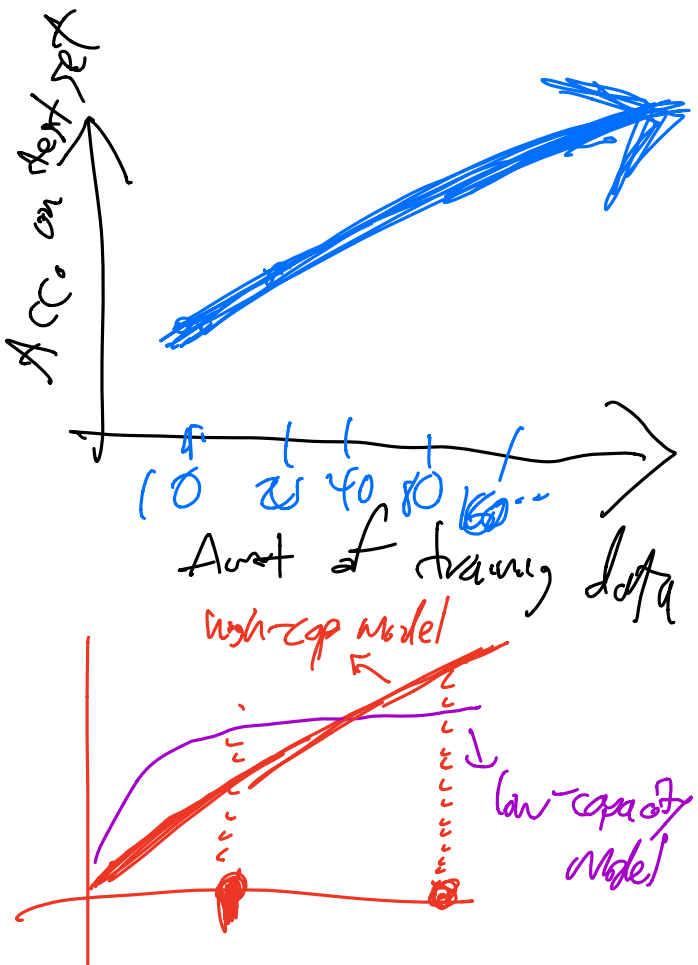
- Reliability analysis: from the social sciences, especially psychology, content analysis, communications, etc.

Exercise

- Let's collect annotations and check agreement rates!
- See links

Do I have enough labels?

- For training, typically thousands of annotations are necessary for reasonable performance
- For evaluation, can get away with fewer (amenable to traditional power analysis)
- Exact amounts are difficult to know in advance. Can do a **learning curve** to estimate if more annotations will be useful.
- (Open research question: how to usefully make NLP models with ~10 or ~100 training examples. "Few-shot learning")



Evaluation of NLP model

- Confusion matrix of counts of each pair (gold standard label, predicted label)
- Several evaluation metrics
  - Accuracy (misleading with class skew)
  - For a single class:
    - Precision, Recall and F1
  - For multiclass: Macro-averaged F1
- Many different metrics out there
  - Ranking metrics (MAP, ROC…): no need to specify threshold for hard classif.
  - Probabilistic calibration
  - Kappa = chance-adjusted accuracy. Typically used for inter-annotator agreement instead of F1; there is no principled reason why this is done.