Classification in NL

Text categorization

- Spam: yes/no
- Language: Polish/Czech/Slovak/Hungarian
- Authorship: Shakespeare/Marlowe
- Persuasive argument: yes/no
- Inference: entailed/contradictory/neither
- Sentiment: positive/neutral/negative
  - Of word/sentence/paragraph/review/article/corpus
  - Toward a target, e.g., hotel/phone/restaurant/app
  - With respect to (*aspect*) cleanliness/screen/service

Bayes Basics

- $P(x \wedge y) = P(x|y)P(y) = P(y|x)P(x)$
- $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$, assuming $P(y) \neq 0$
- Given observation d and classes C
  - We want $\hat{c} = \operatorname{argmax} P(c|d)$, where $c \in C$ (sometimes omitted)
  - Estimate $P(c|d)$ via

$$\hat{c} = \underset{c}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)}$$

  - Get rid of normalization by $P(d)$, fixed for all c
  - $\hat{c} = \operatorname{argmax} P(d|c)P(c) =$ Likelihood $\times$ Prior

Representing Documents

Sometimes not even a complete sentence

- Document d maps to (values for) features $F = \{f_1 \ldots f_n\}$
- What features are apparent in a document?
  - Words, punctuation, paragraph breaks
  - Assume just the words
- How do the features in a document interact?
  - Word order, negation, adjectives, ...
  - Bag of Words (BoW): assume the word counts but nothing else matters
  - Includes bags of n-grams
- Remove stop words
  - From a preset list
  - The top K most frequent words with $K = 10$ or 100, for example

Naïve Bayes for Documents

Naïve: Words are conditionally independent of each other given the class

- $P(f_1 \ldots f_n | c) = P(f_1 | c) \ldots P(f_n | c)$
- Set of classes C, e.g., {pos, neg}
- Set of features F

$$\hat{c_{\mathrm{NB}}} = \operatorname*{argmax}_{c \in C} P(c) \prod_{f \in F} P(f|c)$$

  - Feature: word
  - Feature value: Boolean or frequency
- Use in logspace to avoid arithmetic underflow and improve complexity (addition instead of multiplication)

$$\hat{c_{\mathrm{NB}}} = \operatorname*{argmax}_{c \in C} \log P(c) \sum_{i \in \text{positions}} \log P(w_i|c)$$

- *Linear classifier*: linear function of input features

Training

- V: vocabulary, i.e., set of words
- N: number of documents
- N_c: number of documents in class c

$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(w_i|c) = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)}$$

- Suppose for some w_i

$$\frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)} = 0$$

  - Then, our estimate $\hat{P}(w_i|c) = 0$
  - Then, because of the $\prod$, the net probability is zero
- Smoothing to the rescue
  - Laplace (add 1) remains common for text categorization

Variations for Sentiment

- Remove duplicates within each document before counting
- Generate fake negated tokens
  - From negative word until next punctuation
  - *didn't like this movie, but*
    $\Rightarrow$
    didn't NOT_like NOT_this NOT_movie, but
- Use established sentiment lexicon
  - Fixed positive and negative meanings (all else are neutral)
  - Work well when there isn't enough training data
  - Ignore domain and context

Spam Detection

- Nontextual features
  - Ratio of text to images
  - HTML errors
- Suspicious phrases and tokens
  - Millions of dollars
  - Urgent
  - !!!
- Email message properties
  - Subject line
  - Existence of URLs in the message body

Language Identification

- Subword features
- Bigrams of letters
- Think about languages whose scripts are not letter based
- Think about connection with unknown words

Evaluation

- Ground truth also known as gold labels
- How obtained?
  - People: in what setting? how reliable (same person from time to time; agreement between different people)? how many people?
  - Implicit versus explicit
  - Some other process—as for word vectors (coming up)

Contingency Table and Metrics

Other metrics to come up later

	Gold positive	*Gold negative*
Classified positive	True Positive	False Positive
Classified negative	False Negative	True Negative

- (Top row) Precision $= \frac{\text{TP}}{\text{TP}+\text{FP}}$
- (Left column) Recall $= \frac{\text{TP}}{\text{TP}+\text{FN}}$
- (All) Accuracy $= \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}}$
- F-measure,

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Macroaveraging and Microaveraging

Suited for multinomial classification, e.g., for three classes

- Microaveraging: dominated by most frequent class
  - A simple method such as majority class would dominate
  - Imagine a single, 3×3 contingency table
  - Each row gives the precision for its class
  - Each column gives the recall for its class
- Macroaveraging: treats all classes equally
  - More cautious than microaveraging
  - Separate 2×2 true/false contingency table for each class
  - Precision, recall as before

Test Sets and Cross-Validation

- Ideal
  - Training set
  - Devset or Development test set to tune parameters
  - Test set (unseen until testing) to evaluate
- Training-dev-test split costs too much data
- Cross-validation: in each fold
  - Split training data randomly, e.g., for 10-folds
  - Use one part to train, e.g., 90%
  - Remainder to test, e.g., 10%
- Pollutes our understanding since we see the data
  - We may choose features that suit it well
  - Overfitting
  - Poor performance on real data
- Split off main test set and hold it aside
- Cross-validate within the training set
- Test on the test set to report results

Comparing Classifiers via the Bootstrap Test

Using accuracy as an example

- Methods being compared: A, B
- Test set x of n entries
- Performance gain of A over B $\delta(\cdot)$
- Draw *bootstrap samples* from the test set
  - Surrogates for having real new data
  - Draw b samples $x^{*(i)}$, each of a fixed number n of instances
  - The b samples can overlap
  - Compute $\delta(x^{*(i)})$, expected to be $\delta(x)$
- Compute statistics on the b samples
  - Percentile: count $x^{*(i)}$ where $\delta(x^{*(i)}) > 2\delta(x)$
- Empirical bootstrap: from observations
- Parametric bootstrap: from some parametrized distribution