Computational Sociology

Supervised Text Classification

Dr. Thomas Davidson

Rutgers University

April 4, 2024

Plan

1. Course updates
2. Introduction to supervised text classification
3. Supervised text classification in R
4. Data sampling and annotation
5. Biased predictions

Course updates

Homework

- Homework 4 released today
  - Due 4/12 at 5pm

Course updates

Project timeline

- Initial data collection
  - Deadline extended until 4/8 (Monday) at 5pm
  - Submit Github repository and write-up

Course updates

Initial data collection

- Write-up
  - Preliminary analysis of some or all of the data you will use in your project. Consider this as a draft of the Data section of your manuscript
    - Description of the data collection process
    - Preliminary data analysis
    - Include 1-2 summary tables and 1-2 visualizations

Course updates

Initial data collection

- Submit the following via email (include SOC577 in subject line):
  - A link to a Github repository (add me as a collaborator if it is private)
    - Code used to collect and analyze data (ideally in R/RMarkdown)
    - A document containing the analysis (can be rendered from RMarkdown, but any PDF is fine)

Introduction to supervised text classification

Procedure

- Supervised text classification uses machine-learning to automatically label documents
  - Phase 1: Select a corpus of documents
  - Phase 2: Annotate a small sample of documents with "ground truth" labels
  - Phase 3: Train a model to predict the labels of the annotated documents
  - Phase 4: Use the trained model to predict the labels for the entire corpus

Introduction to supervised text classification

Supervised versus unsupervised approaches

- Both supervised text classification and topic modeling can be used as a replacement for conventional content analysis
  - Topic modeling is an *inductive* approach, useful for summarizing an entire corpus and deriving categories
  - Supervised machine learning is a *deductive* approach, designed to classify documents into discrete (or probabilistic) classes based on pre-defined categoriess
    - Unlike topic modeling, the categories are determined in advance to the analyst

Introduction to supervised text classification

Approaches to text analysis

- Nelson et al. 2018 compare different four approaches to text analysis:
  1. Handcoding
  2. Dictionaries
  3. Unsupervised learning
  4. Supervised learning
- What computational approach can best replicate handcoding?

Introduction to supervised text classification

Approaches to text analysis

- Handcoding is the ideal approach but difficult to scale to large corpora
- Dictionaries
  - Small dictionaries can have high precision, but low recall
  - Note also how large dictionaries can have high recall, but low precision
- Unsupervised learning
  - Topic modeling and similar approaches more useful for exploratory analyses as no guarantee that clusters/topics map onto concept of interest
- Supervised learning
  - Automatically reproduce handcoding at scale without needing a predefined dictionary
  - Important features are learned from the data

Introduction to supervised text classification

Handcoding, dictionaries, and supervised learning

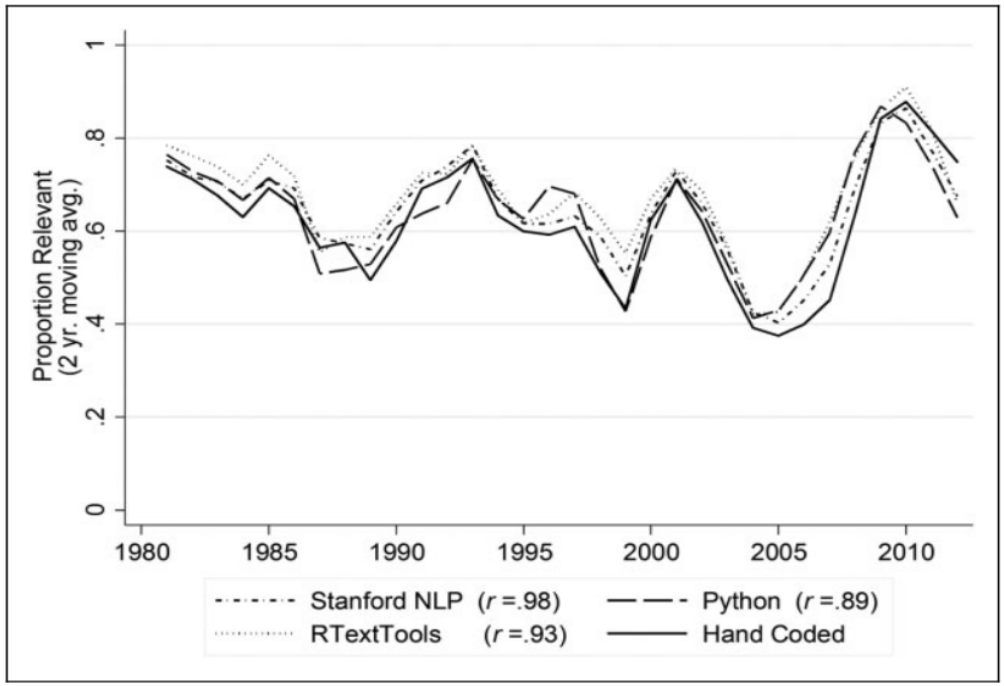
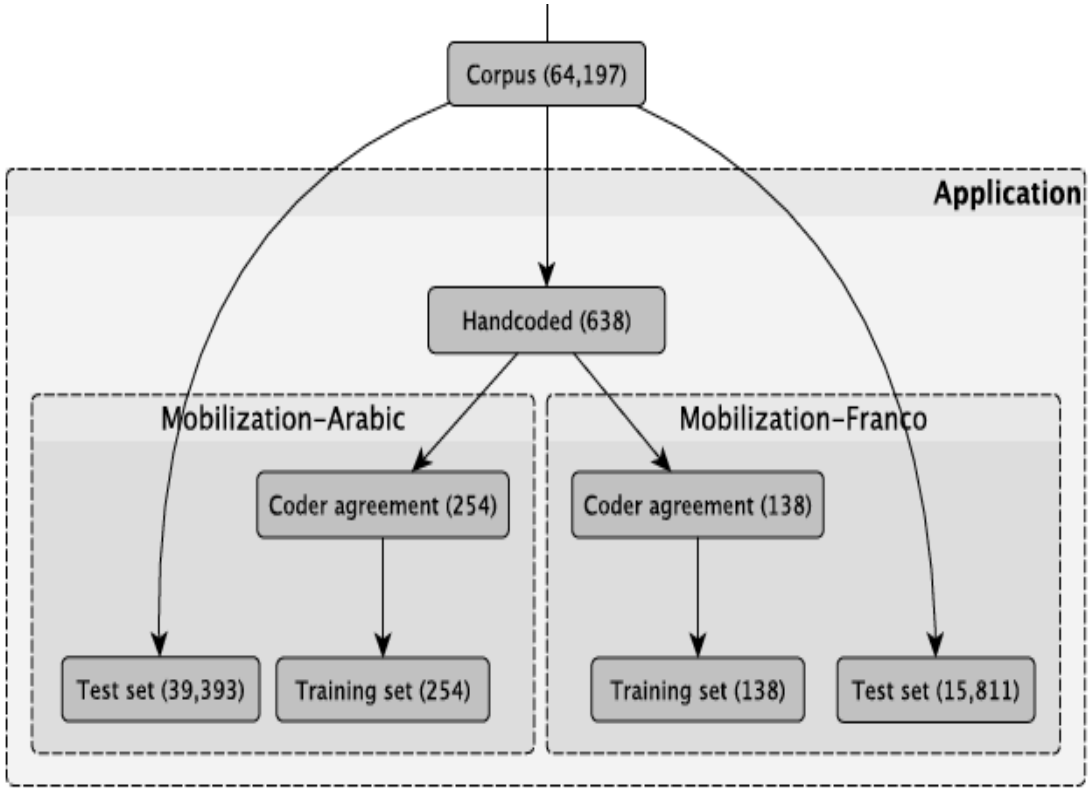


Figure 3. Trends in supervised machine learning analysis of hand-coded articles for relevant versus irrelevant binary scheme (combined relevant substantive categories versus irrelevant category; combined relevant substantive categories shown).

Introduction to supervised text classification

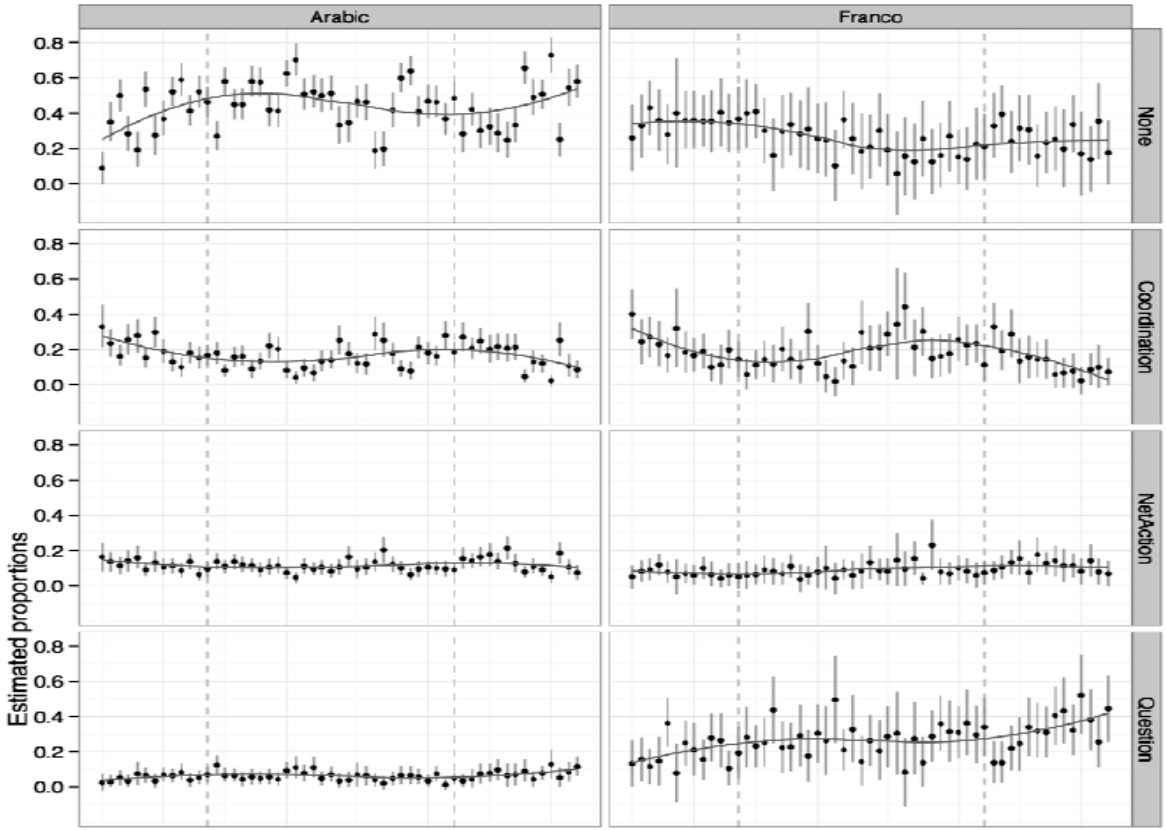
Sociological applications: Social movement framing



Hanna, Alex. 2013. "Computer-Aided Content Analysis of Digitally Enabled Movements." *Mobilization: An International Quarterly* 18 (4): 367–88.

Introduction to supervised text classification

Sociological applications: Social movement framing



Introduction to supervised text classification

Sociological applications: Media frames

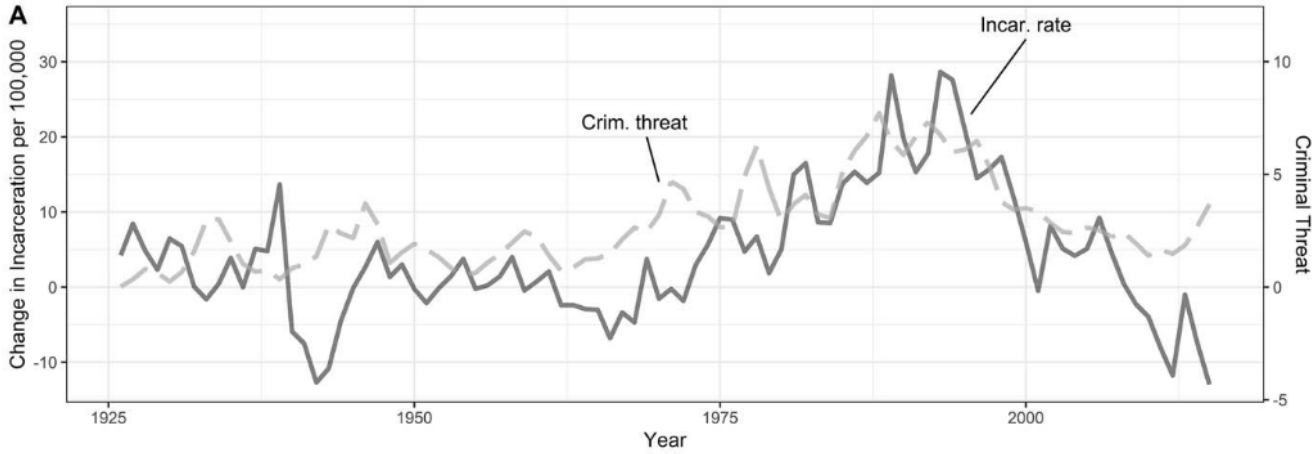
Table 3. Supervised Machine Learning Validation Results

Themes	Accuracy	Precision	Recall	F1 score
Economic threat	0.96	0.79	0.80	0.79
Political threat	0.95	0.88	0.89	0.87
Criminal threat	0.94	0.83	0.86	0.84

Duxbury, Scott W. 2023. "A Threatening Tone: Homicide, Racial Threat Narratives, and the Historical Growth of Incarceration in the United States, 1926–2016." *Social Forces*.

Introduction to supervised text classification

Sociological applications: Media frames



Supervised text classification in R

Case study

- Data
  - A subsample of the IMBD reviews dataset*
    - 5000 IMBD reviews
    - half positive ($\geq 7/10$), half negative ($\leq 4/10$), neutral excluded.
- Classification task
  - Predict which reviews are positive and which are negative (binary)
  - The assumption that the classifier learns the sentiment of the reviews

*Maas, Andrew L, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. "Learning Word Vectors for Sentiment Analysis." In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 142–50.

Supervised text classification in R

Loading data

In this case, we take a random sample of the training data to make the process more tractable.

```
library(tidyverse)
path <- '../data/imbd_reviews_sample.tsv'
d <- read_tsv(path)

d %>% group_by(sentiment) %>% count()
```

Code based on examples from Emil Hvitfeldt and Julia Silge, *Supervised Machine Learning for Text Analysis in R*, Chapter 7

Supervised text classification in R

Train-test split

The first step is to divide our dataset into training and testing data.

```
library(tidymodels)
set.seed(987654)

review_split <- initial_split(d, strata="sentiment", prop = 0.8)
train <- training(review_split)
test <- testing(review_split)
```

Supervised text classification in R

Creating a `recipe`

We begin by creating a `recipe` by specifying the equation and the dataset.

```
review_recipe <- recipe(sentiment ~ text, data  = train)
```

Supervised text classification in R

Adding preprocessing steps

Next, we can use the `textrecipes` library to add preprocessing steps to our recipe. Note that we could also do our preferred preprocessing first using `tidytext` or another package then pass the resulting data directly to our recipe.

```
library(textrecipes)
review_recipe <- review_recipe %>% step_tokenize(text) %>%
  step_tokenfilter(text, max_tokens = 1000) %>%
  step_tfidf(text)
```

Supervised text classification in R

Creating a `workflow`

Once we have a recipe, we're going to be using something called a `workflow` to chain together a sequence of modeling operations.

```
review_wf <- workflow() %>% add_recipe(review_recipe)
```

Supervised text classification in R

Adding a model

We can then add more operations to our workflow to train a model. In this case we will use a logistic regression with a LASSO penalty.

```
#install.packages("glmnet")
lasso <- logistic_reg(penalty = 0.01, mixture = 1) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

review_wf <- review_wf %>% add_model(lasso)
```

Supervised text classification in R

Reviewing the `workflow`

We can print the workflow to review the sequence of operations to be performed.

```
print(review_wf)
```

Supervised text classification in R

Cross-validation

We can then incorporate cross-validation into our model to get a better estimate of out-of-sample performance. In this case we use k-fold/v-fold cross-validation, where `v` is set to 5.

```
review_folds <- vfold_cv(train, v = 5)
```

Supervised text classification in R

Fitting a cross-validated model

We can then use the `fit_resamples` function from the `tune` package to fit our workflow to each of the 10 subsets of data. The `control` parameter specifies information we want to store for further analysis.

```
fitted <- fit_resamples(
  review_wf,
  review_folds,
  control = control_resamples(save_pred = TRUE),
  metrics = metric_set(precision, recall, f_meas, roc_auc)
)
```

Supervised text classification in R

Evaluating overall performance

The `collect_` functions from the `tune` package then allow us to evaluate each model.

```
lasso_pred_probs <- collect_predictions(fitted)


collect_metrics(fitted)
```

Supervised text classification in R

Evaluating performance by fold

By grouping on `id` we can view the estimate for each cross-validation sub-group.

```
lasso_pred_probs %>% group_by(id) %>%
  f_meas(sentiment, .pred_class) %>%
  select(id, .estimate)
```

Supervised text classification in R

Computing an ROC curve

We can view the performance of each classifier using the ROC curve. In general they all appear to perform quite well.

Supervised text classification in R

Selecting tuning parameters

Now we have code we can use to fit a model using cross-validation. The next step is to find the optimal set of parameters. In this case there are two things we might want to vary: the size of the feature matrix and the regularization strength. To do this we need to modify the recipe and model object to specify that we want to `tune` these parameters.

```
review_recipe_2 <- recipe(sentiment ~ text, data  = train) %>% step_tok
  step_tokenfilter(text, max_tokens = tune()) %>%
  step_tfidf(text)

review_wf <- review_wf %>% update_recipe(review_recipe_2)

lasso_2 <- logistic_reg(penalty = tune(), mixture = 1) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

review_wf <- review_wf %>% update_model(lasso_2)
```

Supervised text classification in R

Specifying a parameter grid

Next, we specify a parameter grid using `grid_regular` this defines the parameter space and how it should be broken down. We specify a range of values for each parameter and how many cut-points in this range we are interested in.

```
param_grid <- grid_regular(
  penalty(range = c(-5, 1)),
  max_tokens(range = c(1000, 3000)),
  levels = c(penalty = 5, max_tokens = 3)
)

print(param_grid)
```

Supervised text classification in R

Fitting the model to the parameter grid

Finally, we use `tune_grid` to fit the workflow to these different tuning parameters, using the same cross-validation splits as above. This will take a while since we have 5x3x5 model fits accounting for the combinations of tuning parameters and number of folds. This is similar in logic to the `fit_resamples` but it returns the model with the best-fitting parameters.

```
tune_params <- tune_grid(
  review_wf,
  review_folds,
  grid = param_grid,
  metrics = metric_set(precision, recall, f_meas, roc_auc),
  control = control_resamples(save_pred = TRUE)
)
```

Supervised text classification in R

Evaluating performance by parameter

```
tuned_metrics <- collect_metrics(tune_params)

tuned_metrics %>% group_by(penalty, .metric) %>%
  summarize(mean_score= mean(mean)) %>%
  filter(.metric == "f_meas")

tuned_metrics %>% group_by(max_tokens, .metric) %>%
  summarize(mean_score= mean(mean)) %>%
  filter(.metric == "f_meas")
```

Supervised text classification in R

Plotting the results

Even better, we can directly plot the relationship between the variables. In this case it seems like the regularization makes a much bigger difference than the size of the feature matrix.

```
autoplot(tune_params) +
  labs(title = "Lasso model performance across regularization penalties
       color = "Number of tokens") + scale_color_viridis_d()
```

Supervised text classification in R

Selecting the best model and updating the workflow

Let's take the model with the best F1 score and fit it to our data.

```
best_f1 <- tune_params %>% select_best(metric = "f_meas")
print(best_f1)

final_wf <- finalize_workflow(review_wf, best_f1)

print(final_wf)
```

Supervised text classification in R

Fitting the model to the training data

Now we can fit the model to our entire training dataset *and* assess its performance on the test set, which has not been used so far. Note how we are not using cross-validation now, we are re-training the best model using all of the training data.

```
final_fitted <- last_fit(final_wf, review_split)
```

Supervised text classification in R

Evaluating out-of-sample performance

```
collect_metrics(final_fitted)
```

Supervised text classification in R

Evaluating out-of-sample performance

```
final.precision <- collect_predictions(final_fitted) %>% precision(truth
final.recall <- collect_predictions(final_fitted) %>% recall(truth=sent
final.f1 <- collect_predictions(final_fitted) %>% f_meas(truth=sentimen
print(bind_rows(final.precision, final.recall, final.f1))
```

Supervised text classification in R

Evaluating out-of-sample performance

```
collect_predictions(final_fitted) %>%
  conf_mat(truth = sentiment, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```

Supervised text classification in R

Evaluating out-of-sample performance

We can similarly view the ROC curve for the held-out data. This shows that our model performs well out-of-sample.

Supervised text classification in R

Error analysis

The final step we can take is to conduct some analysis of the predictions. In particular, its often most insightful to look at the errors.

```
reviews_bind <- collect_predictions(final_fitted) %>%
  bind_cols(test %>% select(-sentiment))

pos_errors <- reviews_bind %>%
  filter(sentiment == "pos", .pred_pos < 0.3) %>%
  select(text) %>%
  slice_sample(n = 5) %>% print()

neg_errors <- reviews_bind %>%
  filter(sentiment == "neg", .pred_neg < 0.3) %>%
  select(text) %>%
  slice_sample(n = 5) %>% print()
```

Supervised text classification in R

Calculating feature importance

The vip package allows us to calculate feature importance scores and to see which are most strongly associated with each class.

```
library(vip)

reviews_imp <- pull_workflow_fit(final_fitted$.workflow[[1]]) %>%
  vi(lambda = best_f1$penalty)

imp <- reviews_imp %>%
  mutate(
    Sign = case_when(Sign == "POS" ~ "More positive",
                     Sign == "NEG" ~ "More negative"),
    Importance = abs(Importance),
    Variable = str_remove_all(Variable, "tfidf_text_")) %>% group_by(Si
  top_n(15, Importance) %>%
  ungroup
```

Supervised text classification in R

Visualizing feature importance

Supervised text classification in R

Next steps

- Test different algorithms and parameters
  - SVM and neural networks both work well for many text classification problems
- Alternative feature representations
  - N-grams
  - Document embeddings (LSA, Word2vec, GLoVE, BERT)
  - Topic distributions
  - Non-textual features
- Evaluate the impact of different pre-processing techniques

(Denny and Spirling 2018)

Data sampling and annotation

Overview

- We used a dataset with "ground truth" labels, derived from the numeric scores given to movies by IMDB users.
- In practice, we often have to produce this annotated dataset before we can train a model.
  - What data do we want to classify?
  - What is the coding scheme?
  - Which texts should we annotated?
  - How many annotated examples do we need?
  - Who should perform annotation?

Data sampling and annotation

What data and categories?

- The choice of data and categories is usually driven by a combination of substantive and theoretical concerns.
- Consider the following example based on my research on hate speech
  - Sampling frame: 27k tweets containing keywords from a crowdsourced hate speech database.
  - Categories:
    - Hate speech
    - Offensive language
    - Neither

Data sampling and annotation

What data and categories?

- Codebook development to identify a meaningful set of categories by reading documents
- Clear definitions and examples
- Intercoder reliability checks

Data sampling and annotation

What data and categories?

- Important to consider generalizability:
  - Context (e.g. publication, platform)
  - Period
  - Language / dialect
  - Medium (e.g. text, image, video)
- Heuristic: Classifier trained on one domain likely less accurate (or entirely wrong) on other tasks
  - Example: Sentiment $\neq$ Stance (more next week)

(Bestvater and Monroe 2022)

Data sampling and annotation

The unit of analysis

- Select an appropriate unit of analysis
  - A book, a chapter, a paragraph, a sentence?
- This is usually guided by theory
  - e.g. If we want to classify the political leaning of a comment then the unit is the comment but if we want to classify users then we might consider aggregating their comments.
- Empirical evaluations
  - Limited benefits to classifying sentences rather than chunks of newspaper articles, but this may not generalize to other cases

(Barberá et al. 2020)

Data sampling and annotation

Keywords and categories

- Keywords often used to sample documents for analysis
  - Social media posts
  - Newspaper articles
  - Academic articles
  - Archival documents

Data sampling and annotation

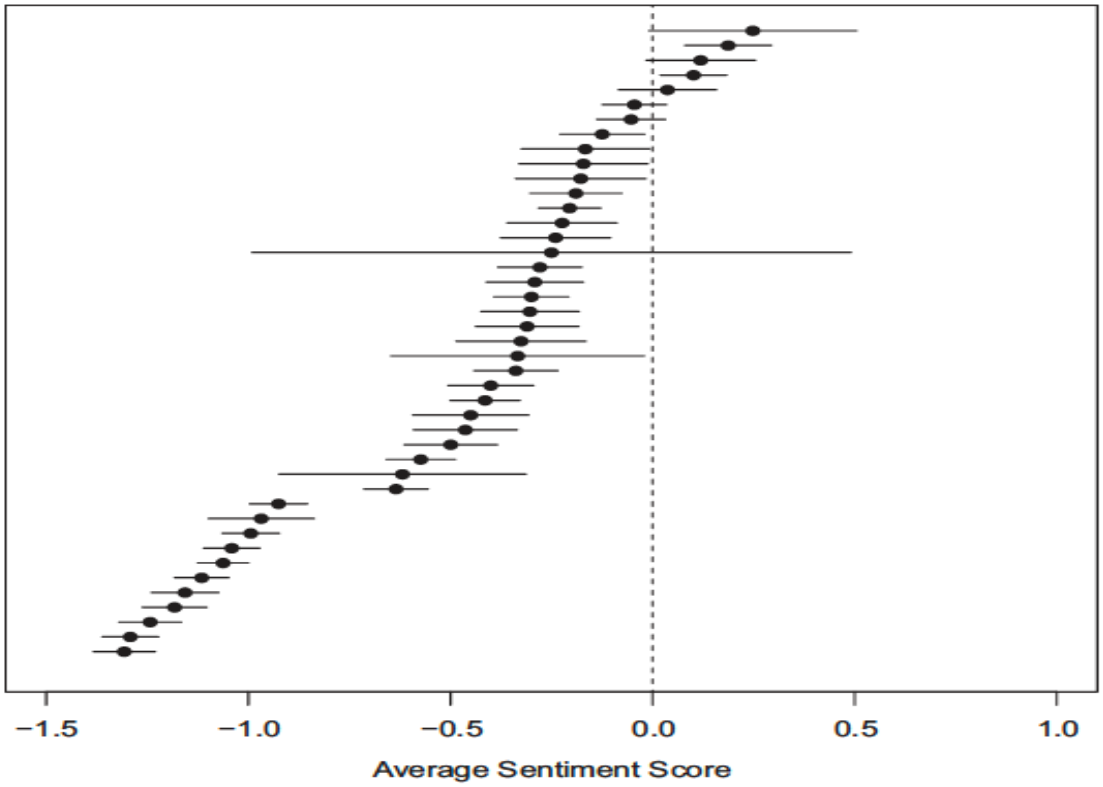
Keywords and categories

A Obamacare

King, Gary, Patrick Lam, and Margaret E. Roberts. 2017. "Computer-Assisted Keyword and Document Set Discovery from Unstructured Text." *American Journal of Political Science* 61 (4): 971–88.

Data sampling and annotation

Keywords and categories



Note: Each document set was selected by a different keyword list, with point estimates (as dots) and 95% confidence intervals (horizontal lines) shown.

King, Gary, Patrick Lam, and Margaret E. Roberts. 2017. "Computer-Assisted Keyword and Document Set Discovery from Unstructured Text." *American Journal of Political Science* 61 (4): 971–88.

Data sampling and annotation

Keywords and categories

- Recommendations
  - Start with a small set of keywords ~1-5
  - Use an automated discovery approach* and/or domain expertise to expand the keyword set
  - Increase the size of the keyword set as long as returned documents increase but relevant proportion does not decline (Barberá et al. 2020: 8).

* Python code to implement King, Lam, and Roberts' approach.

Data sampling and annotation

Which specific examples should be annotated?

- Once we have our sample we need a way to select examples for annotation
  - The goal is to annotate a representative sample of documents such that we can train a classifier that generalizes well beyond the training data
- Generally we use some form of random sampling or stratified random sampling

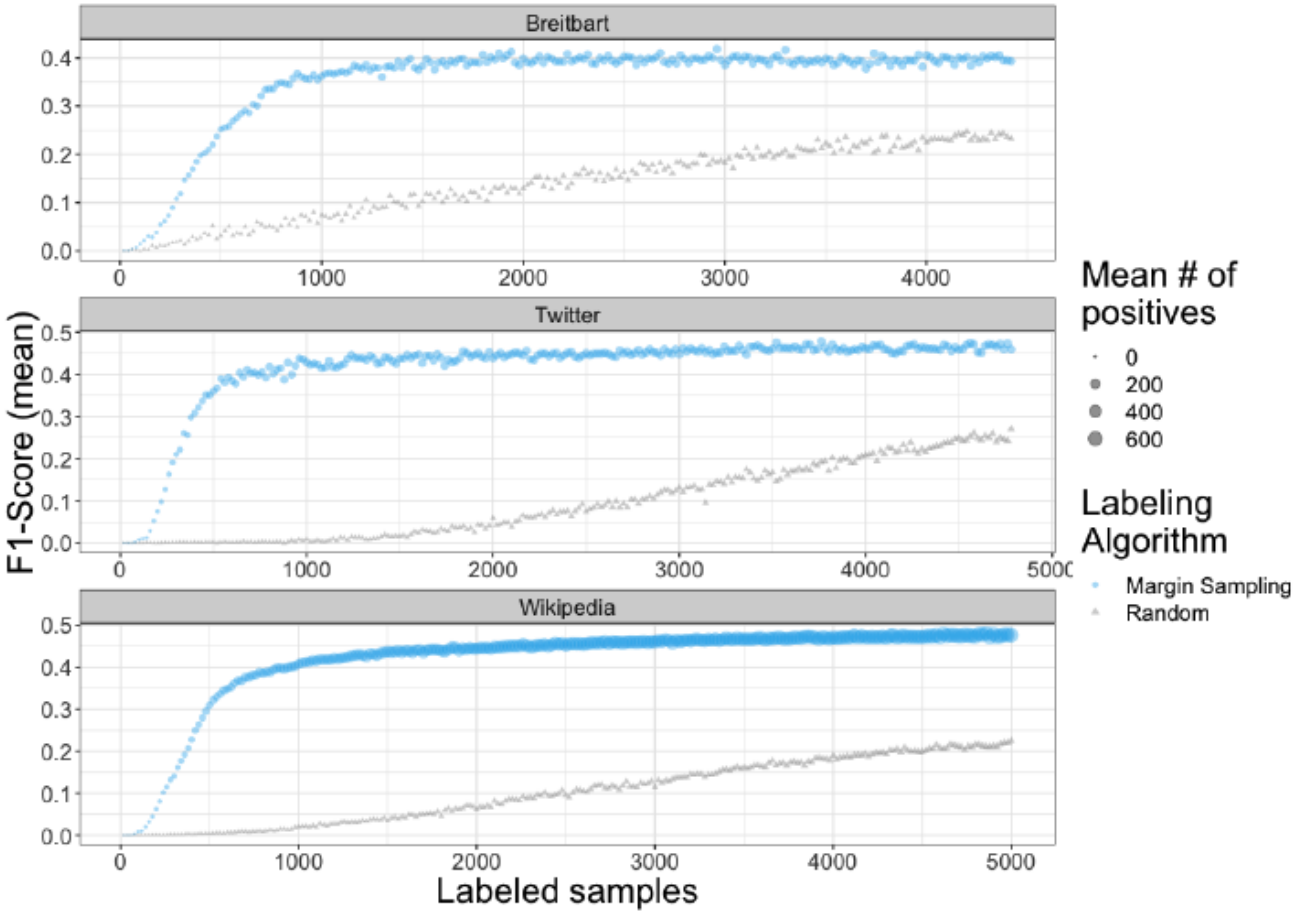
Data sampling and annotation

Which specific examples should be annotated?

- Random sampling can be inefficient
  - Duplicate or highly similar documents are redundant
  - Many documents do not provide any new information
- We can do better by using an approach called *active learning*

Data sampling and annotation

Active learning in practice



Miller, Blake, Fridolin Linder, and Walter R. Mebane, Jr. 2019. "Active Learning Approaches for Labeling Text: Review and Assessment of the Performance of Active Learning Approaches." *Political Analysis*.

Data sampling and annotation

How many cases do we need?

- Context specific
  - How balanced are the class distributions?
    - e.g. Evenly split like IMBD or are some classes rare?
  - How complex/difficult is the classification task?
    - Number of categories
    - Difficulty of classification
- Heuristic: Required sample size increases with imbalance, complexity, and task difficulty

Data sampling and annotation

How many cases do we need?

- Empirical results
  - Evidence of diminishing returns in a newspaper content classification task after ~1000 examples

    (Barberá et al. 2020)
  - Active learning can substantially reduce the necessary sample size compared to random sampling

    (Mebane et al. 2019)
  - Large language models can drastically reduce necessary sample size (more next week)

Data sampling and annotation

How should annotation be conducted?

- Who should do the coding?
  - Expert coders vs. research assistants vs. crowdworkers
    - Task and budget key determinants
- How many coders per data point?
  - Conventional to have multiple coders to ensure reliability
  - Preferable to have more examples coded by fewer people than fewer examples coded by more people

    (Barberá et al. 2020)
    - Although there should be some overlap for calculating intercoder-reliability

Data sampling and annotation

Biased predictions

- Sampling and annotation procedures can also result in downstream biases
  - Sampling biases
    - Over/undersampling
    - Selection bias
  - Annotation biases
    - Stereotypical judgments
    - Domain knowledge and representativeness

Biased predictions

Digital audit studies of text classifiers

How it Works

Using Machine Learning to Reduce Toxicity Online

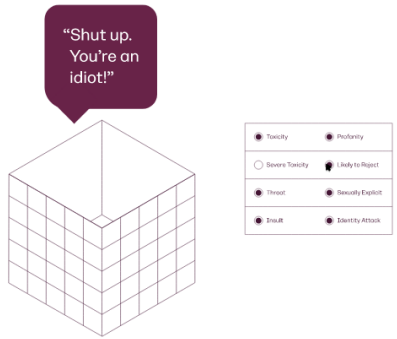
Perspective uses machine learning models to identify abusive comments. The models score a phrase based on the perceived impact the text may have in a conversation. Developers and publishers can use this score to give feedback to commenters, help moderators more easily review comments, or help readers filter out "toxic" language.

Perspective models provide scores for several different attributes. In addition to the flagship Toxicity attribute, here are some of the other attributes Perspective can provide scores for:

- Severe Toxicity
- Identity attack
- Insult
- Threat
- Profanity
- Sexually explicit

To learn more about our ongoing research and experimental models, visit our Developers site.

LEARN MORE



https://perspectiveapi.com/how-it-works/

Biased predictions

Digital audit studies of text classifiers

Term	Toxic	Overall
atheist	0.09%	0.10%
queer	0.30%	0.06%
gay	3%	0.50%
transgender	0.04%	0.02%
lesbian	0.10%	0.04%
homosexual	0.80%	0.20%
feminist	0.05%	0.05%
black	0.70%	0.60%
white	0.90%	0.70%
heterosexual	0.02%	0.03%
islam	0.10%	0.08%
muslim	0.20%	0.10%
bisexual	0.01%	0.03%

Table 1: Frequency of identity terms in toxic comments and overall.

Dixon et al. 2018.

Biased predictions

Digital audit studies of text classifiers

Template Examples	Label
I am <IDENTITY>	Non-Toxic
I am a <IDENTITY> person, ask me anything	Non-Toxic
<IDENTITY> people are just like everyone else	Non-Toxic
I hate all <IDENTITY>	Toxic
I am a <IDENTITY> person and I hate your guts and think you suck	Toxic
<IDENTITY> people are gross and universally terrible	Toxic

Table 2: Phrase template examples.

Biased predictions

Digital audit studies of text classifiers

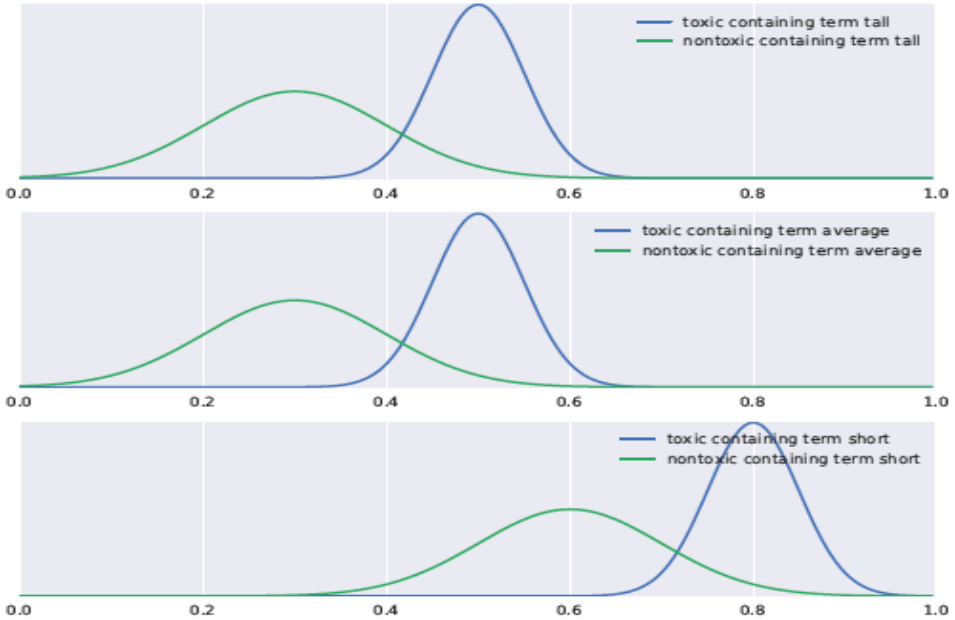
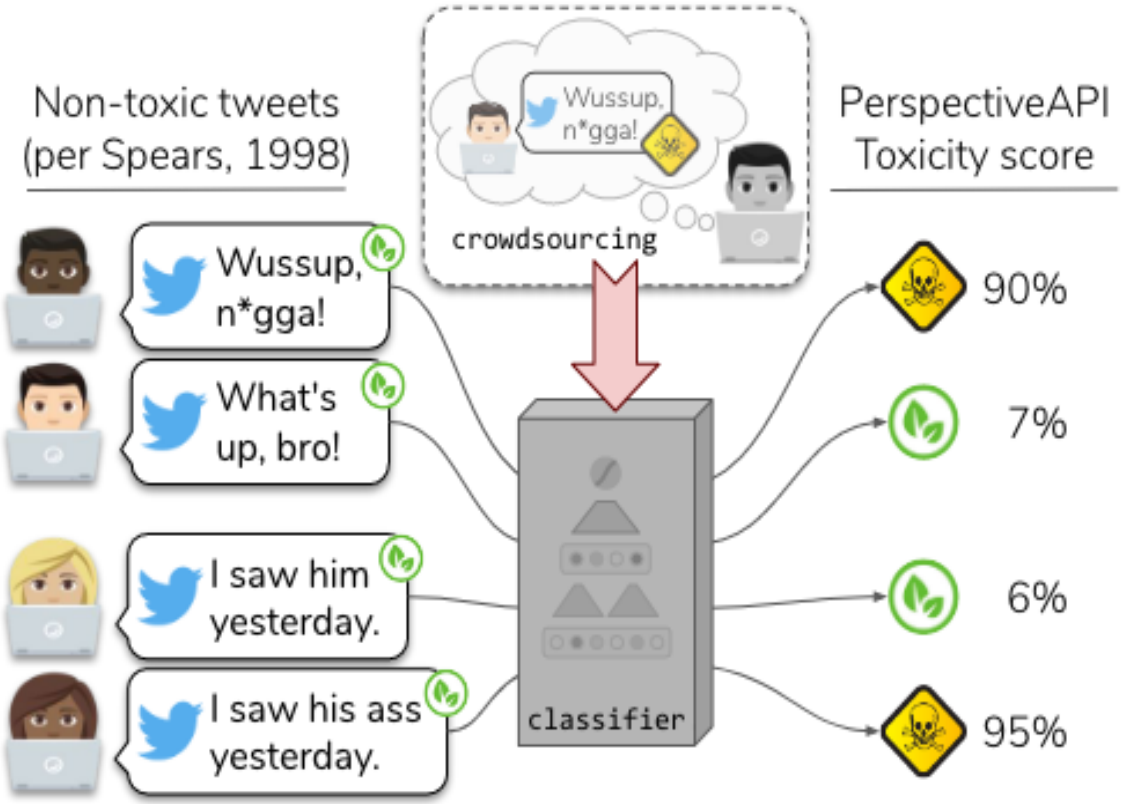


Figure 2: Distributions of toxicity scores for three groups of data, each containing comments with different identity terms, "tall", "average", or "short".

Biased predictions

Racial bias in hate speech detection classifiers



Sap, Maarten, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A Smith. 2019. *The Risk of Racial Bias in Hate Speech Detection.* In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 1668–78. ACL.

Biased predictions

Racial bias in hate speech detection classifiers

Dataset	Class	$\widehat{p_{i_{black}}}$	$\widehat{p_{i_{white}}}$	t	p	$\frac{\widehat{p_{i_{black}}}}{\widehat{p_{i_{white}}}}$
Waseem and Hovy	Racism	0.001	0.003	-20.818	***	0.505
	Sexism	0.083	0.048	101.636	***	1.724
Waseem	Racism	0.001	0.001	0.035		1.001
	Sexism	0.023	0.012	64.418	***	1.993
	Racism and sexism	0.002	0.001	4.047	***	1.120
Davidson et al.	Hate	0.049	0.019	120.986	***	2.573
	Offensive	0.173	0.065	243.285	***	2.653
Golbeck et al.	Harassment	0.032	0.023	39.483	***	1.396
Founta et al.	Hate	0.111	0.061	122.707	***	1.812
	Abusive	0.178	0.080	211.319	***	2.239
	Spam	0.028	0.015	63.131	***	1.854

Davidson, Bhattacharya, and Weber 2019.

Biased predictions

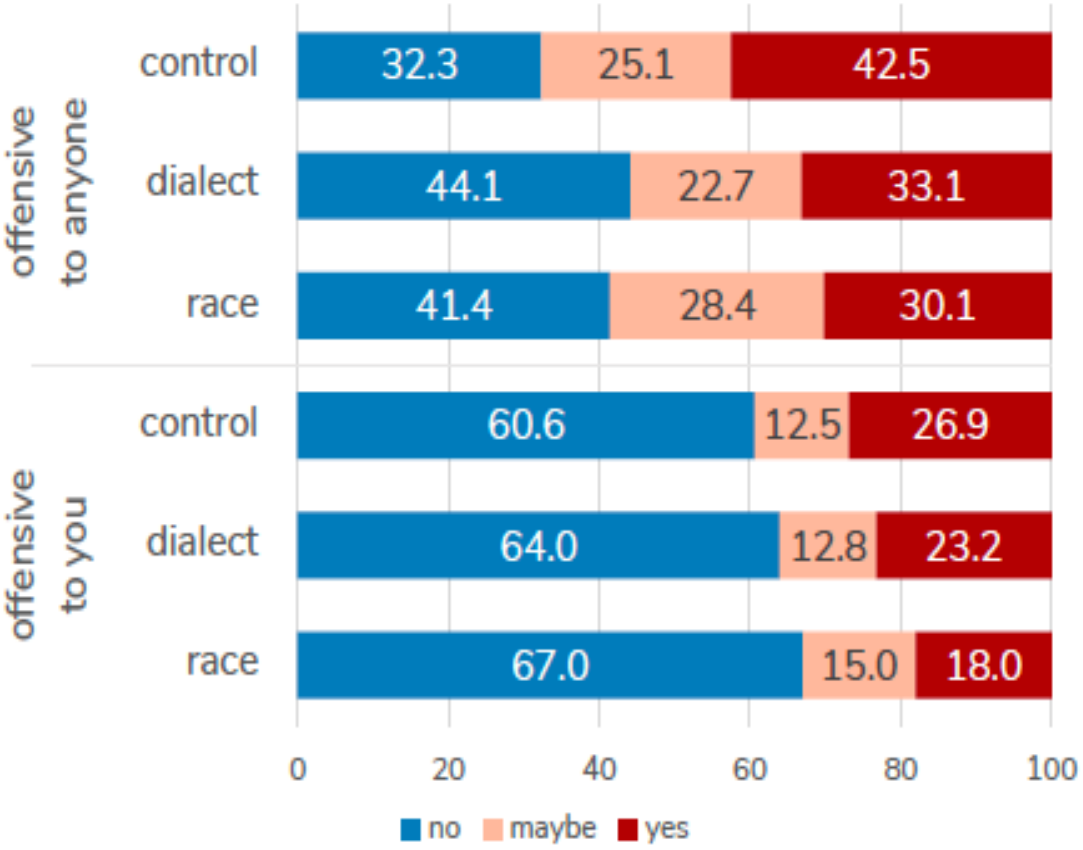
Racial bias in hate speech detection classifiers

Dataset	Class	$\widehat{p_{i_{black}}}$	$\widehat{p_{i_{white}}}$	t	p	$\frac{\widehat{p_{i_{black}}}}{\widehat{p_{i_{white}}}}$
Waseem and Hovy	Racism	0.010	0.010	-0.632		0.978
	Sexism	0.963	0.944	20.064	***	1.020
Waseem	Racism	0.011	0.011	-1.254		0.955
	Sexism	0.349	0.290	28.803	***	1.203
	Racism and sexism	0.012	0.012	-0.162		0.995
Davidson et al.	Hate	0.017	0.015	4.698	***	1.152
	Offensive	0.988	0.991	-6.289	***	0.997
Golbeck et al.	Harassment	0.099	0.091	6.273	***	1.091
Founta et al.	Hate	0.074	0.027	46.054	***	2.728
	Abusive	0.925	0.968	-41.396	***	0.956
	Spam	0.010	0.010	0.000		1.000

This table shows results for tweets containing the word "b***h". This word was used in ~1.7% of AAE and 0.5% of SAE tweets.

Biased predictions

Racial bias in hate speech detection classifiers



Sap et al. 2019

Biased predictions

Racial bias in hate speech detection classifiers

Threatening language	**F5**: Direct threat	"I am going to shoot the next [IDENTITY] I see."	hateful
	F6: Threat as normative statement	"We should get rid of [IDENTITY] once and for all."	hateful
Slur usage	**F7**: Hate expressed using slur	"You are just a [SLUR] to me."	hateful
	F8: Non-hateful homonyms of slurs	"I had this queer feeling we were being watched."	non-hate
	F9: Reclaimed slurs	"I'm the freshest n*gga in the room."	non-hate
Profanity usage	**F10**: Hate expressed using profanity	"These [IDENTITY] should shut the f*ck up once and for all."	hateful
	F11: Non-hateful use of profanity	"I wish someone could get me the f*ck out of here."	non-hate

Röttger, Paul, Bertie Vidgen, Dong Nguyen, Zeerak Waseem, Helen Margetts, and Janet Pierrehumbert. 2021. "HateCheck: Functional Tests for Hate Speech Detection Models." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 41–58. ACL.

Data sampling and annotation

Final thoughts

- Data sampling and annotation is an overlooked area of research but has a huge impact on downstream applications
  - *Garbage in, garbage out*
- There are many different decisions involved that require careful consideration
  - Unlike the machine learning phase, we typically don't have the budget to do a grid-search over the parameter space!

Summary

- Supervised text classification combines NLP and ML to classify documents into classes
- Training data must be carefully sampled and annotated
- Model features and parameters are selected to maximize predictive accuracy
- Error analysis and feature importance provide insight into perforance
- Models should be evaluated for biases
- Once a model performs well and has been validated out-of-sample, we use it to predict the remainder of the corpus

Next week

- Large language models