Supervised Learning for NLP II

HSS 510: NLP for HSS

Taegyoon Kim

Apr 3, 2024

Agenda

Things to be covered

- Algorithms for supervised learning (carried over from last week)
  - Logistic regression: generating predictions & estimating parameters
- Active learning for labeling text

Reminder

The next two weeks

- No classes on **Apr 10 (election day)** and **Apr 17 (mid-term break)**
- One-page proposal by April 17 23:59pm (okay to exceed one page)
  - This can include research question(s), selection/collection of corpus/data (including justification of selection*, potential bias, etc.), basic descriptive statistics of the corpus/data, plans for analysis, results from preliminary analysis, etc.
- Submitted to the instructor email in PDF
  - File name: firstname_proposal.pdf (e.g., taegyoon_proposal.pdf)

Algorithms for Supervised Learning

Various approaches

- Logistic regression
- Naive Bayes
- Support vector machine
- Tree-based models (decision tree, random forest, XGBoost, etc.)
- Neural networks: DNN, RNN, LSTM, CNN, etc.
- **Fine-tuning of large language models**
  - **Excellent performance with less data**

Algorithms for Supervised Learning

Logistic regression

- Used to classify a document into binary categories
  - Multinomial logistic regression for more than two
- One of the most useful analytics tools in science (not just NLP/ML)
- The baseline supervised learning algorithm for classification
- Forms the basis of neural networks

Algorithms for Supervised Learning

Components of classification with logistic regression

- Features
  - A document is represented as a vector of features $\vec{x}$ (= $[x_1, \ldots, x_n]$)
- A classification function (F)
  - $p(y = 1 \mid x)$ is computed for each document given the feature vector ($\vec{x}$) and the parameters (β: $\vec{w}$ and b)
  - The sigmoid function transforms $p(y = 1 \mid x)$ into a value between 0 & 1
- Loss function (measures how model prediction $\hat{y}$ is different from y during training), and an algorithm for optimizing it (gradient descent)

Algorithms for Supervised Learning

How does logistic regression compute predicted probabilities?

- $p(y = 1 \; x)$
  - The probability of y = 1 given a feature vector $\vec{x}$(= $[x_1, \ldots, x_n]$)
  - E.g., for a simple count vector, it would be # of times each token appears in the document
- Logistic regression learns β, a vector of coefficients
  - A bias term b: a single number (a.k.a. intercept)
  - Weights $\vec{w}$(= [$w_1, \ldots,$ w_n])
    - E.g., tokens signaling hateful intention would get high weights
  - With b, $\vec{w}$, and $\vec{x}$, we compute $z = (\sum_{i=1}^{n} w_i x_i) + b$

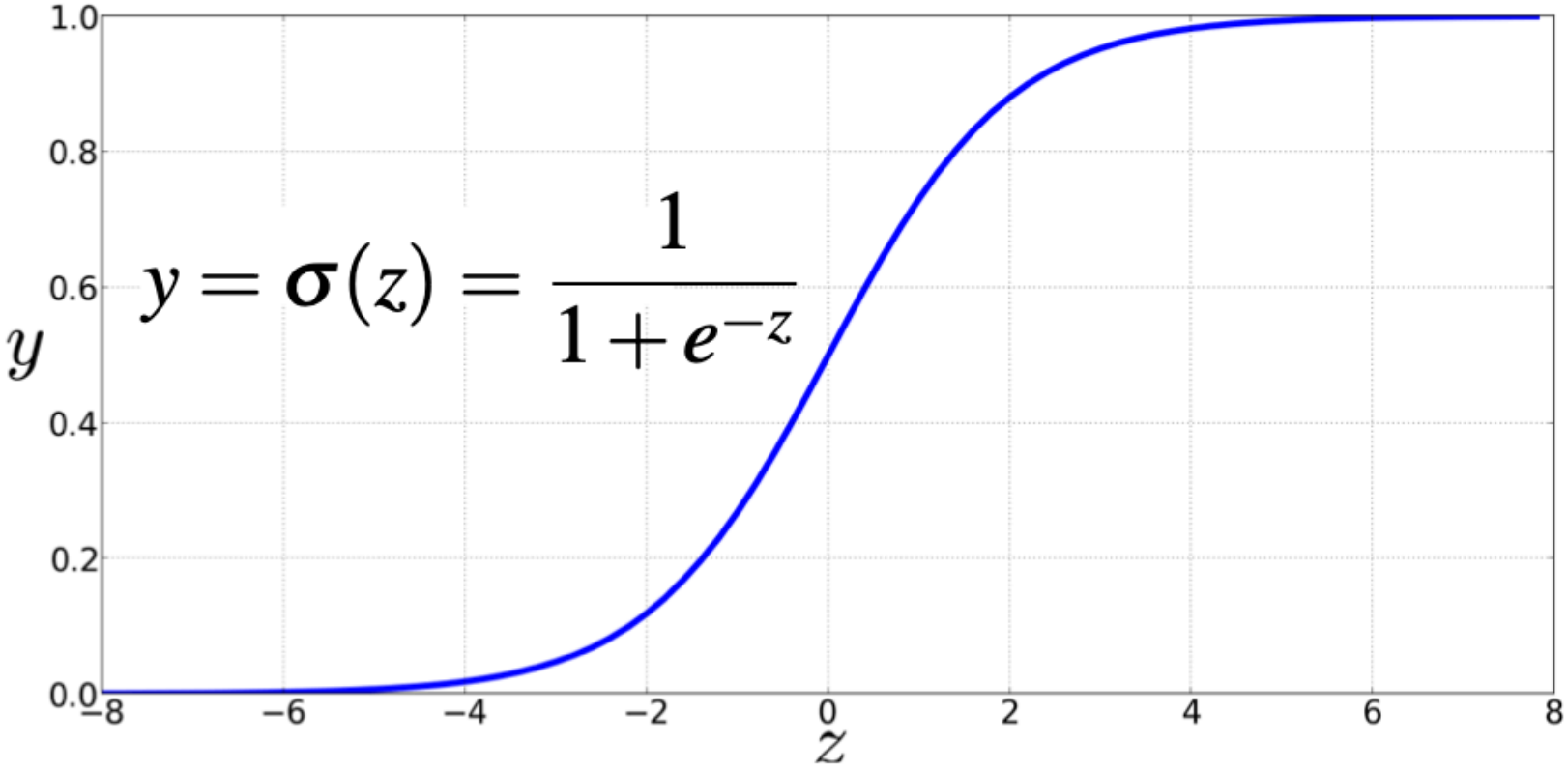
Algorithms for Supervised Learning

How does logistic regression compute predicted probabilities?

- $z = (\sum_{i=1}^{n} w_i x_i) + b = \vec{w} \cdot \vec{x} +$ b

Algorithms for Supervised Learning

Sigmoid function



Algorithms for Supervised Learning

How does logistic regression compute predicted probabilities?

- $z = (\sum_{i=1}^{n} w_i x_i) + b = \vec{w} \cdot \vec{x} + b$

- $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+exp(-z)}$

Algorithms for Supervised Learning

How does logistic regression compute predicted probabilities?

- $p(y = 1 \mid x) = \sigma(\vec{w} \cdot \vec{x} + b)$

- $p(y = 0 \mid x) = 1 - \sigma(\vec{w} \cdot \vec{x} + b)$

Algorithms for Supervised Learning

How do predicted probabilities turn into binary labels ($\hat{y}$)?

$$\begin{cases} 1 & \text{if } P(y = 1 \mid x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

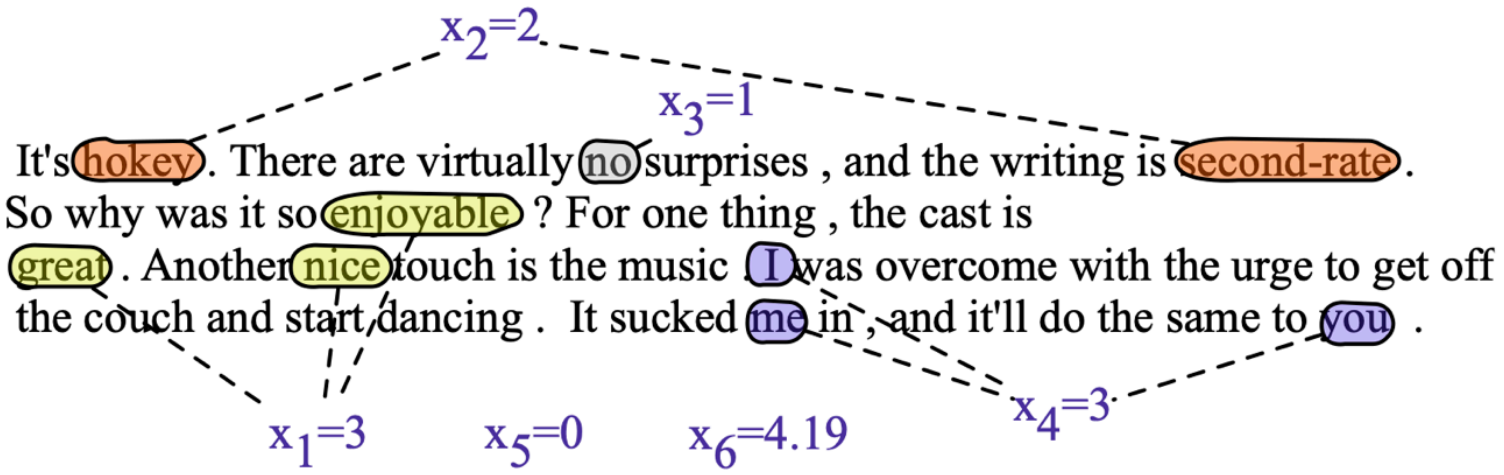
Algorithms for Supervised Learning

E.g., sentiment classification from movie reviews

“It’s hokey. There are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great, Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing? It sucked me in, and it’ll do the same to you.”

Algorithms for Supervised Learning

Var	Definition
x_1	count(positive lexicon words $\in$ doc)
x_2	count(negative lexicon words $\in$ doc)
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	count(1st and 2nd pronouns $\in$ doc)
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	ln(word count of doc)



Algorithms for Supervised Learning

How are the parameters, weights $\vec{w}$ and bias b, learned?

- Goal: learn $\vec{w}$ and b that make $\hat{y}^i_{train}$ for each training observation as "close" as possible to y^i_{train} (the true label)
- Two components for estimation
  - Metric for "closeness": loss/cost function (e.g., cross entropy loss)
  - Optimization algorithm: (stochastic) gradient descent

Algorithms for Supervised Learning

Learn parameters that maximize the chance of getting the correct label (Conditional Maximum Likelihood Estimation)

Algorithms for Supervised Learning

From p($y|x$), we drive loss (cost) function

- p($y|x$) = $\hat{y}^{y} * (1 - \hat{y})^{1-y}$

  $\rightarrow$ Log(p($y|x$)) = Log($\hat{y}^{y} * (1 - \hat{y})^{1-y}$)

  $\rightarrow$ Log(p($y|x$)) = $y$log$\hat{y}$ + (1-y)log(1-$\hat{y}$)

Turn it into the cross entropy loss function (that should be minimized)

- L_{CE} = -[$y$log$\hat{y}$ + (1-y)log(1-$\hat{y}$)]

Algorithms for Supervised Learning

Gradient descent

- With the cross entropy loss function, we have a metric for discrepancies
- Gradient descent is an algorithm used to find the optimal set of weights (and bias) that minimizes the discrepancies, averaged over all observations

$$\rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(f(x^{(i)}; \theta), y^{(i)})$$

Algorithms for Supervised Learning

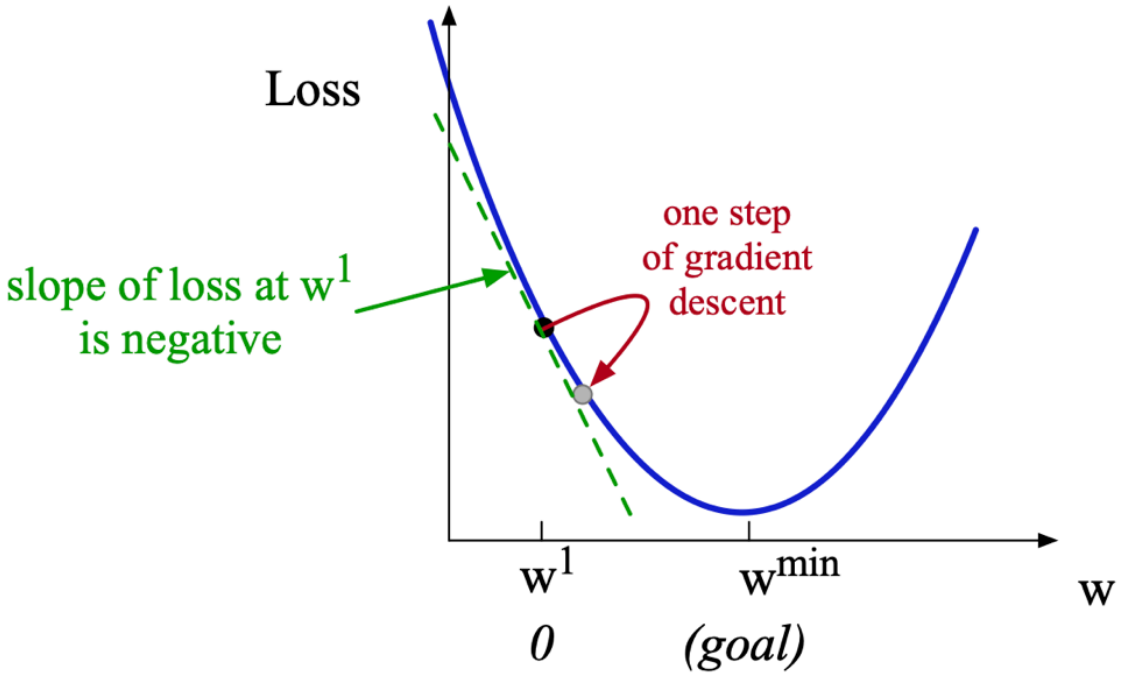
Gradient descent finds a minimum of a function

- Figures out in which direction the function’s slop rises most steeply
- Then move in the opposite direction
- Stops when it reaches the minimum

Algorithms for Supervised Learning

Gradient descent

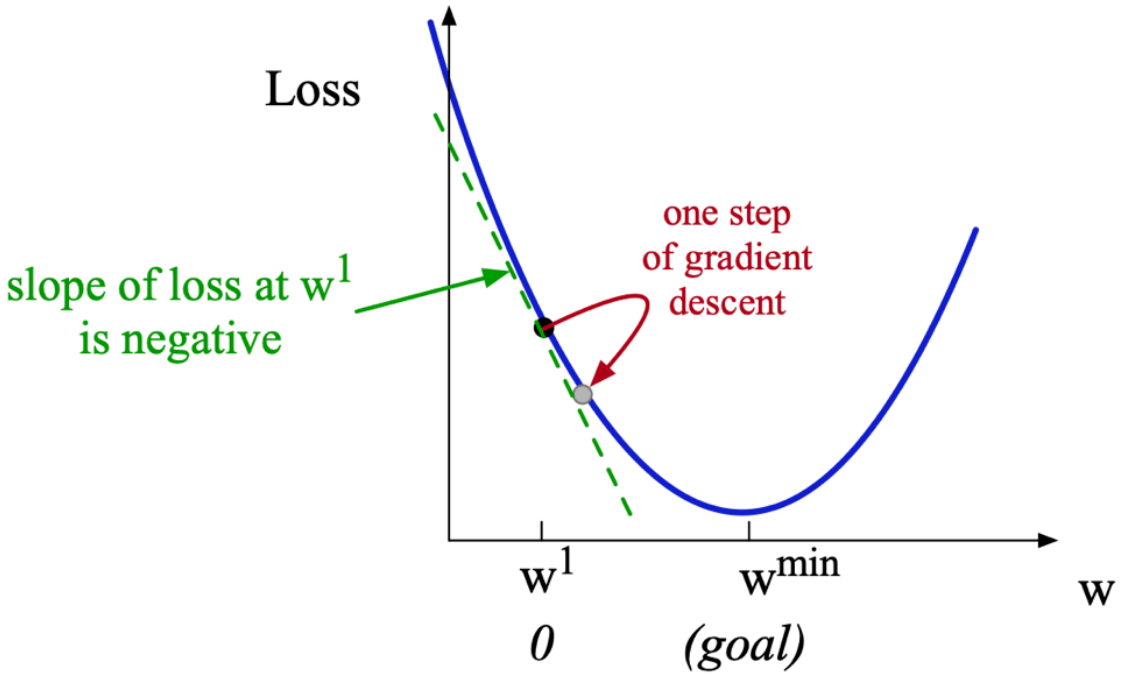
- 1-dimensional illustration



Algorithms for Supervised Learning

Gradient descent

- Updating w with learning rate (η): $w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x; w), y)$



Activce Learning for Labeling Text

Cost of Manual labeling

- Expensive because labeled data is required to train a model
- Particularly expensive when classes are imbalanced
- A random sampling will yield a very small number of relevant documents
  - E.g., identifying news articles that refer to a terrorist attack
  - E.g., identifying social media posts containing hate speech
    - Siegel et al. (2019): less than 1% for most of the time

Activce Learning for Labeling Text

Cost of Manual labeling

- Particularly relevant for data-hungry models (neural networks)
- Still relevant with transformer-based large language models
  - Model fine-tuning helps a lot, but does not eliminate the necessity for labeled data

Activce Learning for Labeling Text

Active learning

- Choose data to be labeled in a way that reduces the labeling of documents
- Documents that are informative to the model are selected
- Increase in the efficiency of human labeling
- Meaningful with imbalanced classification
  - For balanced tasks, random sampling is likely to be effective

Activce Learning for Labeling Text

Class imbalance

- One label occurs much more frequently than the other
- It takes an enormous amount of labeled data to get enough information about the minority class
- E.g., the relevant class appears 1% of the time
  - labeling of 5,000 documents → 50 relevant documents in expectation
  - The model trained on this data would likely be terrible

Activce Learning for Labeling Text

For each iteration

- Label documents
- Train a a model
- Use the model to identify documents to label

New documents to label are selected based on the model's uncertainty ("margin sampling")

- Documents with predicted probabilities near 0.5 are ones that the model is least certain about

Activce Learning for Labeling Text

Iterative process

1. Start with an initial set of labeled documents: $y_{labeled}$, $D_{labeled}$
2. Train a model F using $y_{labeled}$, $D_{labeled}$
3. Produce predicted probabilities for each unlabeled document in $D_{unlabeled}$
4. Select and label a new set of documents from $D_{unlabeled}$ based on model uncertainty ($|\hat{Pr}(y=1) - 0.5|$): y_z, D_z
5. Add y_z, D_z to $y_{labeled}$, $D_{labeled}$ (remove them from $D_{unlabeled}$)
6. Repeat Steps 1–5

Activce Learning for Labeling Text

Experiments (Miller et al. 2020)

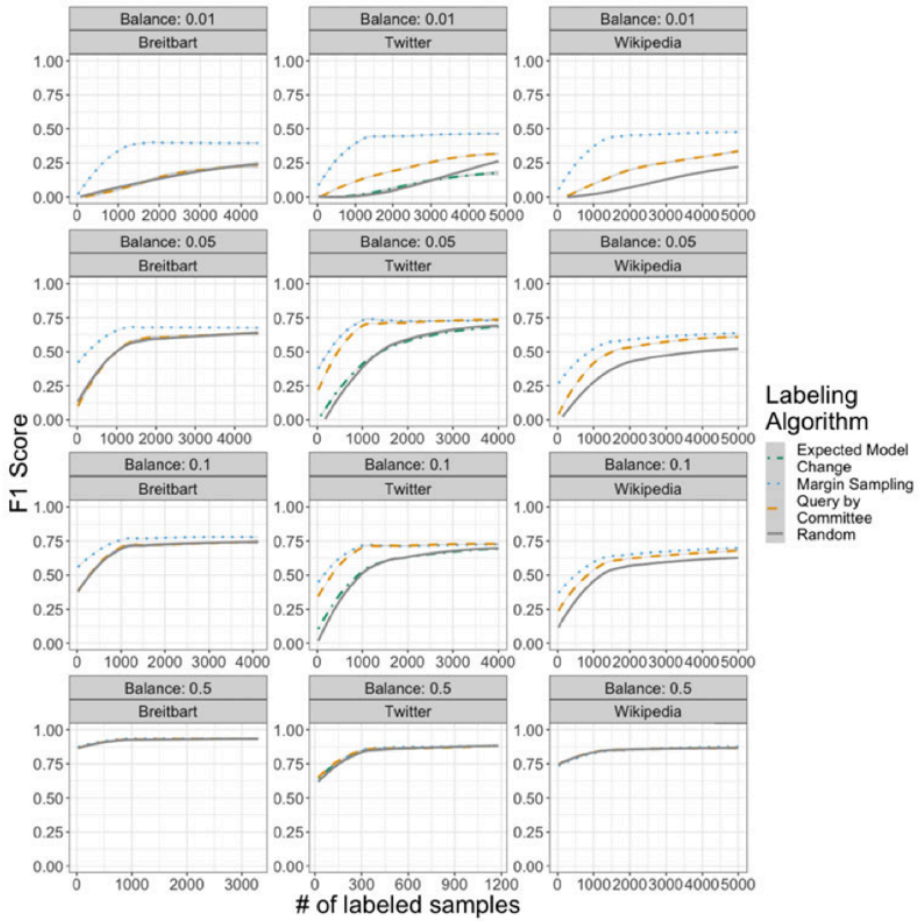


Figure 3. F1-score for experiments. The panel columns correspond to the datasets and the rows to the different levels of class imbalance. Dots represent single replications of the experiment and smoothed lines are fits (and standard errors) of a generalized additive model.

Activce Learning for Labeling Text

Experiments (Miller et al. 2020)

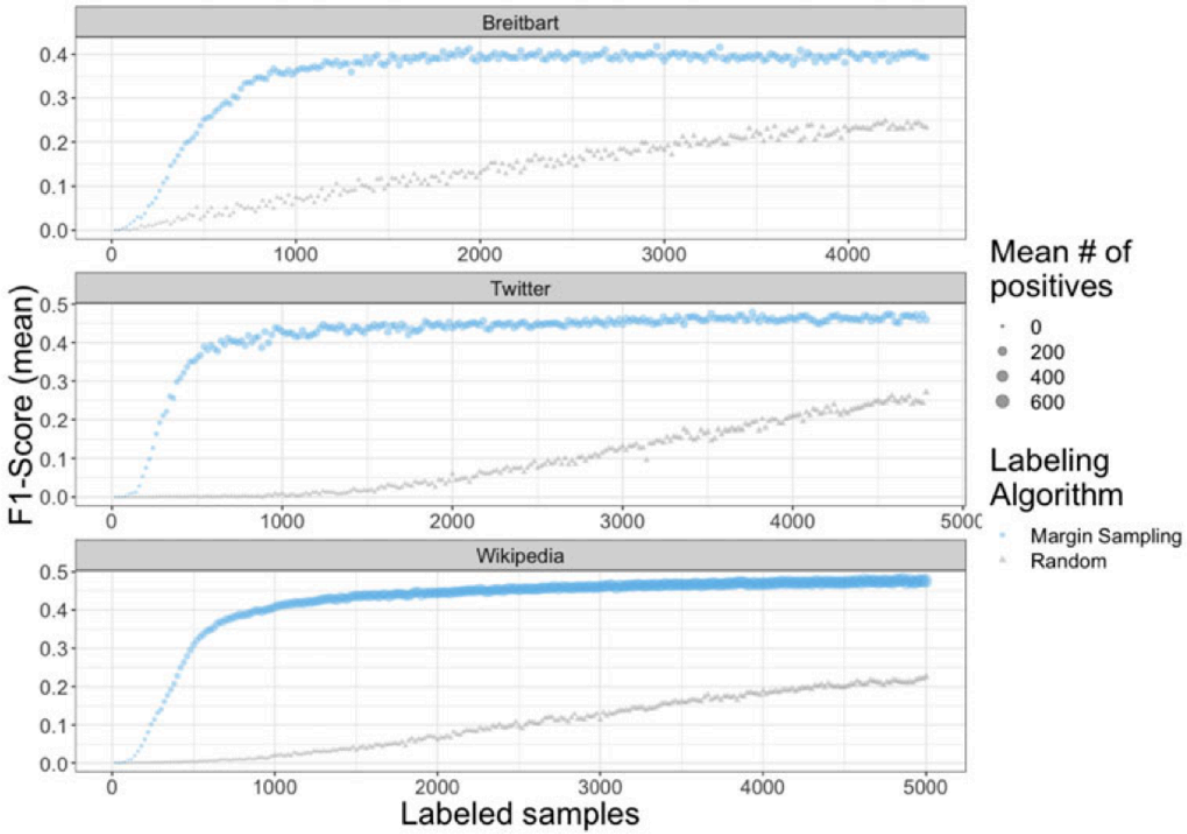


Figure 4. Performance by number of labeled examples for classifiers trained with active and passive learning (with class balance 0.01). Dots represent the average classifier performance across replications. Dot size is proportional to the average number of positively labeled observations in the training sample across replications.

Activce Learning for Labeling Text

Experiments (Miller et al. 2020)

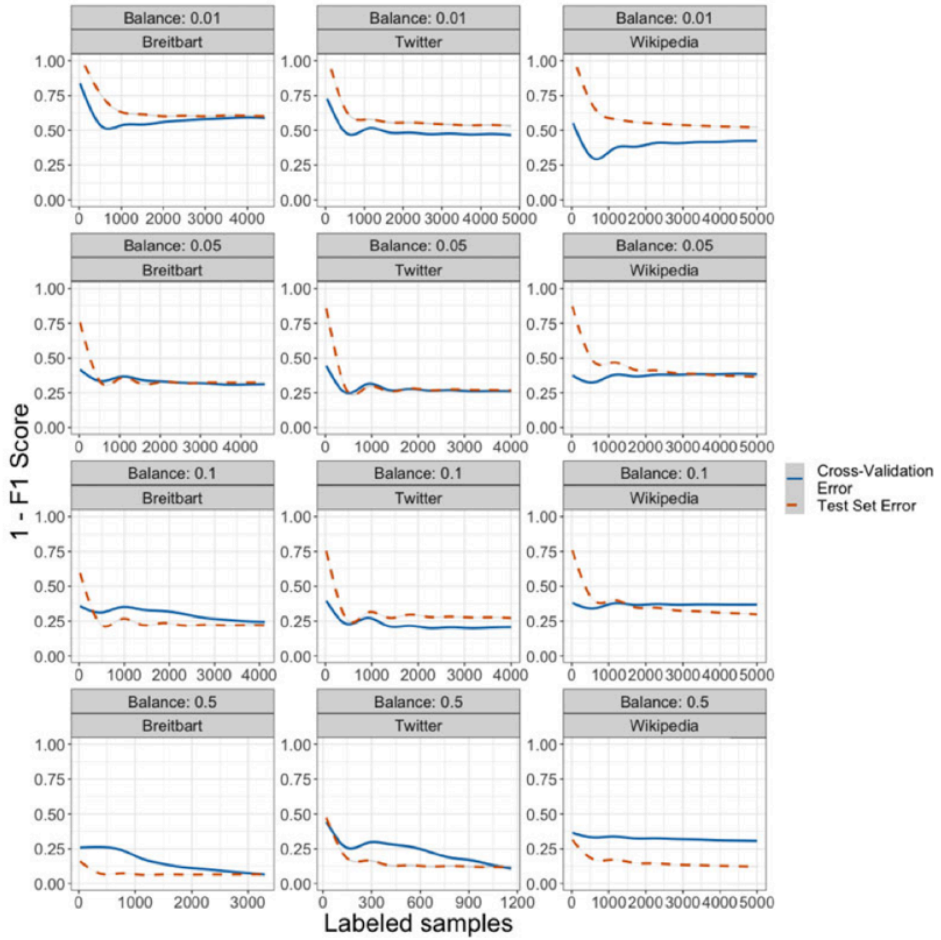


Figure 6. Comparison of generalization error (in terms of F1-score) in training and test set for active learning with margin querying strategy. Results are from the Twitter dataset.

Summary

Text classification with supervised learning

- Importance of carefully generated labeled data
  - Clear coding guidelines, evaluation of ICR, etc.
- Considerations in terms of resources put into generating annotations
  - Active learning can alleviate this concern
- Once we have a good labeled data, it can be very robust