Text Processing - Basics

Advanced Social Computing

Department of Computer Science
University of Massachusetts, Lowell
Fall 2020

Hadi Amiri
hadi@cs.uml.edu

UMASS LOWELL

Lecture Topics

- Text Data

- Learning word vectors
  - Word2vec
  - Glove
- Evaluating word vectors
- Retrofitting word vectors

NLP – High Level

- Tokenization, OCR
- Normalization
  - urls, hashtags, punctuations, numbers, dates, cases, stop words, etc.
  - Spell correction
- Morphological analysis
  - Stemming, lemmatization, etc.
- Syntactic analysis
  - Structure of sentences
- Semantic analysis
  - Meaning
- Discourse analysis
  - Pragmatics and context

NLP – High Level

- Tokenization
- Normalization
- Morphology
- Syntax
- Semantic
- Discourse

Text cleaning is a very important first step! But there is no general rule.

- Is it safe to remove punctuations or stop words from text?
  - “switching from Verizon” vs. “switching to Verizon.”
- Or convert all characters to lowercase?
  - “Bush” vs. “bush.”
- Or remove all numbers?
  - “7 yrs old” vs. “70 yrs old.”

- **General rule**: Use the exact same cleaning technique for all competing models.

Many Interesting Applications

- Search
- Information Extraction
- Question Answering
- Machine Translation
- Summarization
- Dialogue Systems
- Text Classification
  - Emails: spam, not-spam.
  - News articles: business, health, sports, tech, etc.
  - Reviews: positive, negative, neutral.
  - Word pairs: synonyms or not.
  - Essays as: A, B, C, D, or F
  - Etc.
- Chatbots
- Dialog agents
- Translators
- Advertisements
- Sentiment analysis
  - Stock market
  - Products

Text Classification

- Let's say we have:
  - A set of documents
    - $\mathbf{X}=\{x_1, \ldots, x_n\}$
  - A set of labels or predicted classes
    - $\mathbf{Y}=\{Class\text{-}1, \ldots, Class\text{-}k\}$
  - We know the label for each document
    - $(x_1,y_1),\ldots,(x_n,y_n)$
  - We aim to learn a function *f* (classifier) that can map inputs to their corresponding outputs
    - $f{:}\mathbf{X} \rightarrow \mathbf{Y}$

Input

Output

Text Classification

- Let's say we have:
  - A set of documents
    - $\mathbf{X}=\{x_1,\ldots,x_n\}$
  - A set of labels or predicted classes
    - $\mathbf{Y}=\{Class\text{-}1, \ldots, Class\text{-}k\}$
  - We know the label for each document
    - $(x_1,y_1),\ldots,(x_n,y_n)$
  - We aim to learn a function *f* (classifier) that can map inputs to their corresponding outputs
    - $f: \mathbf{X} \rightarrow \mathbf{Y}$

Input

Output

Text Classification- Cnt.

- **X**={ i love verizon's coverage ,
  actually t-mobile has great deals,
  i hate t-mobile! One more bill!!,
  i cant take it anymore! hate verizon}

- **Y**={+1, -1}

negative

positive

Text Classification- Cnt.

- **X**={ i love verizon's coverage ,
  actually t-mobile has great deals,
  i hate t-mobile! One more bill!!,
  i cant take it anymore! hate verizon}
- **Y**={+1, -1}

- $\{(x_1, y_1), (x_2, y_2), \ldots, (x_4, y_4)\}=$
  {(i love verizon's coverage, +1),
  (actually t-mobile has great deals, +1),
  (i hate t-mobile! One more bill!!, -1),
  (i cant take it anymore! hate verizon, -1) }

Text Classification- Cnt.

- **X**={ i love verizon's coverage ,
  actually t-mobile has great deals,
  i hate t-mobile! One more bill!!,
  i cant take it anymore! hate verizon}
- **Y**={+1, -1}

- f(i love verizon's coverage) = +1
- f(actually t-mobile has great deals) = +1
- f(i hate t-mobile! One more bill!!) = -1
- f(i cant take it anymore! hate verizon) = -1

Classification: the output variable takes class labels, i.e. Y={-1,+1}
Regression: the output variable takes continuous values, i.e. Y=[-1,+1].

Why do we need to learn f?

Text Classification- Cnt.

$f\Big($

cant wait to leave verizon for t-mobile! one more bill!!

i cant take it anymore, the unlimited data isnt even worth it. my days with verizon are numbered.

verizon: i will change carriers as soon as contract is up.

verizon your customer service is horrible. this loyal customer will be gone.

$\Big) = \mathbf{Y}$

?

How can we determine *f*?

How can we determine f

- Given $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, we aim to find $f(.)$!
- An ideal $f(.)$ is a function such that
  - $f(x_i) = y_i$ *for all* i
  - Hard to find, why?
  - We just expect $f(x_i)$ to be very close to y_i.

y	f(x)	$error = (y - f(x))^2$
+1	+1	0
-1	-1	0
+1	-1	4
-1	+1	4

zero error when prediction and actual label are the same,
Non-zero, otherwise!

Loss function

$$l(y_i, f(x_i)) = (y_i - f(x_i))^2$$

$$\sum_i l(y_i, f(x_i)) = \sum_i (y_i - f(x_i))^2$$

How can we determine f- Cnt.

- Given $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ we aim to find a function $f(.)$ that minimizes the error

$$L(x, y) = \sum_{i}^{n} l(y_i, f(x_i))$$

- Three popular loss functions
  - Squared loss (linear classifier)
  - Hinge loss (the SVMs),
  - Logistic loss (logistic classifier)

How can we determine *f*- Cnt.

- Three popular loss functions $$L(x, y) = \sum_{i}^{n} l(y_i, f(x_i))$$
  - Squared loss (linear classifier)

$$l(y, f(x)) = (y - f(x))^2$$

  - Hinge loss (the SVMs)

$$l(y, f(x)) = max(0, 1 - y.f(x))$$

  - Logistic loss (logistic classifier)

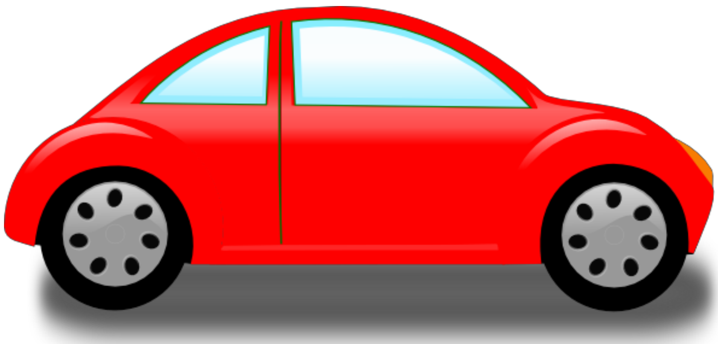
$$l(y, f(x)) = log(1 + exp(-y.f(x)))$$

Text Representation

- What is a good way to represent the input text?

Text Representation- Cnt.

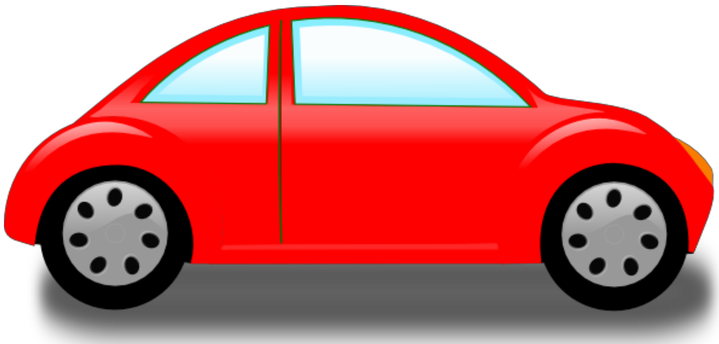
- Features
  - How to classify objects such as **People** and **Cars**?



  - We use *features / characteristics* of those objects!

Text Representation- Cnt.

- Knowledge about features that make good predictors of class membership!
  - having wheels or not distinguishes people from cars, but doesn't distinguish cars from planes.



Text Representation- Cnt.

- **X**={ i would love verizon coverage,
  i hate verizon one more bill,
  i hate verizon}
- **Y**={+1, -1}

Bag of Word representation

- `Features=[i, would, love, verizon, coverage, hate, one, more, bill]`

Text Representation- Cnt.

- **X**={ i would love verizon coverage,
  i hate verizon one more bill,
  i hate verizon}
- **Y**={+1, -1}

Bag of Word representation

- `Features=[i, would, love, verizon, coverage, hate, one, more, bill]`

Feature Weights

	i	would	love	verizon	coverage	hate	one	more	bill
x_1	1	1	1	1	1	0	0	0	0
x_2	1	0	0	1	0	1	1	1	1
x_3	1	0	0	1	0	1	0	0	0

Text Representation- Cnt.

- Bag of Word representation
- **X**={ i would love verizon coverage,
  i hate verizon one more bill,
  i hate verizon}
- **Y**={+1, -1}

Sentiment Words

- Features={would, love, hate}

	would	love	hate
x_1	1	1	0
x_2	0	0	1
x_3	0	0	1

Text Representation- Cnt.

- Other ways of representation?

- Other ways to set weights?

- How to encode semantics?
  - Suggest & recommend
  - Pretty & beautiful
  - Etc.

Vowpal Wabbit (VW)

- Vowpal Wabbit:
  - Fast learning
  - Simplicity
  - Namespace definition
    - Easy Ablation Analysis

http://hunch.net/~vw/

Label [Importance] [Base] ['Tag] |Namespace Feature ... |Namespace Feature ...

Namespace = A letter like ‘a’, ‘b’, ‘c’, ...
Feature = String[:Float]

Vowpal Wabbit (VW)

- Vowpal Wabbit:
  - Fast learning
  - Simplicity
  - Namespace definition
    - Easy Ablation Analysis

```
+1 |a i would love verizon coverage   |b would love
-1 |a i hate verizon one more bill    |b hate
-1 |a i hate verizon                  |b hate
```

Vowpal Wabbit (VW)

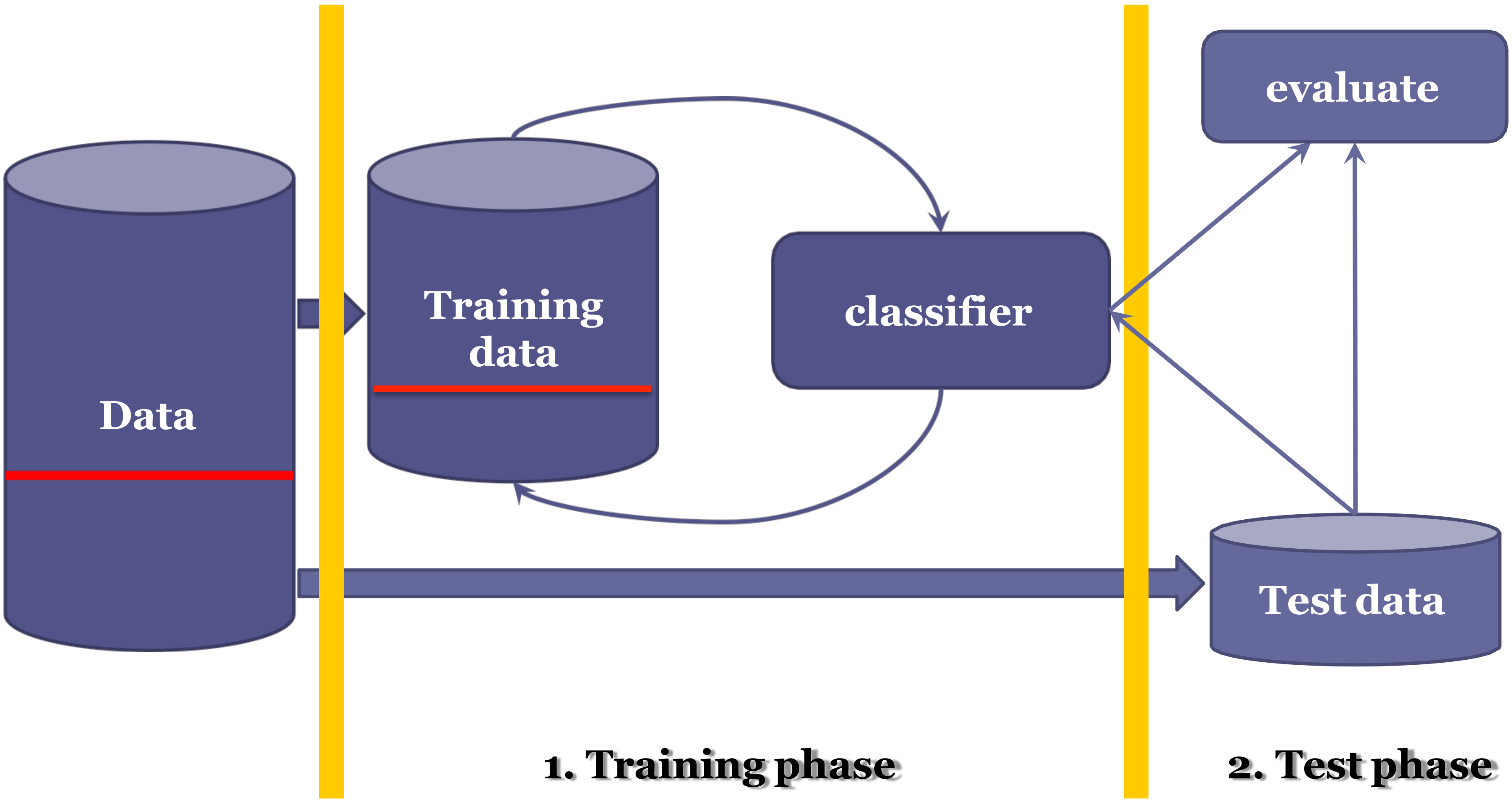
- Vowpal Wabbit:
  - Fast learning
  - Simplicity
  - Namespace definition
    - Easy Ablation Analysis

Label　　Features of a　　Features of b

```
+1 |a i would love verizon coverage |b would love
-1 |a i hate verizon one more bill  |b hate
-1 |a i hate verizon                |b hate
```

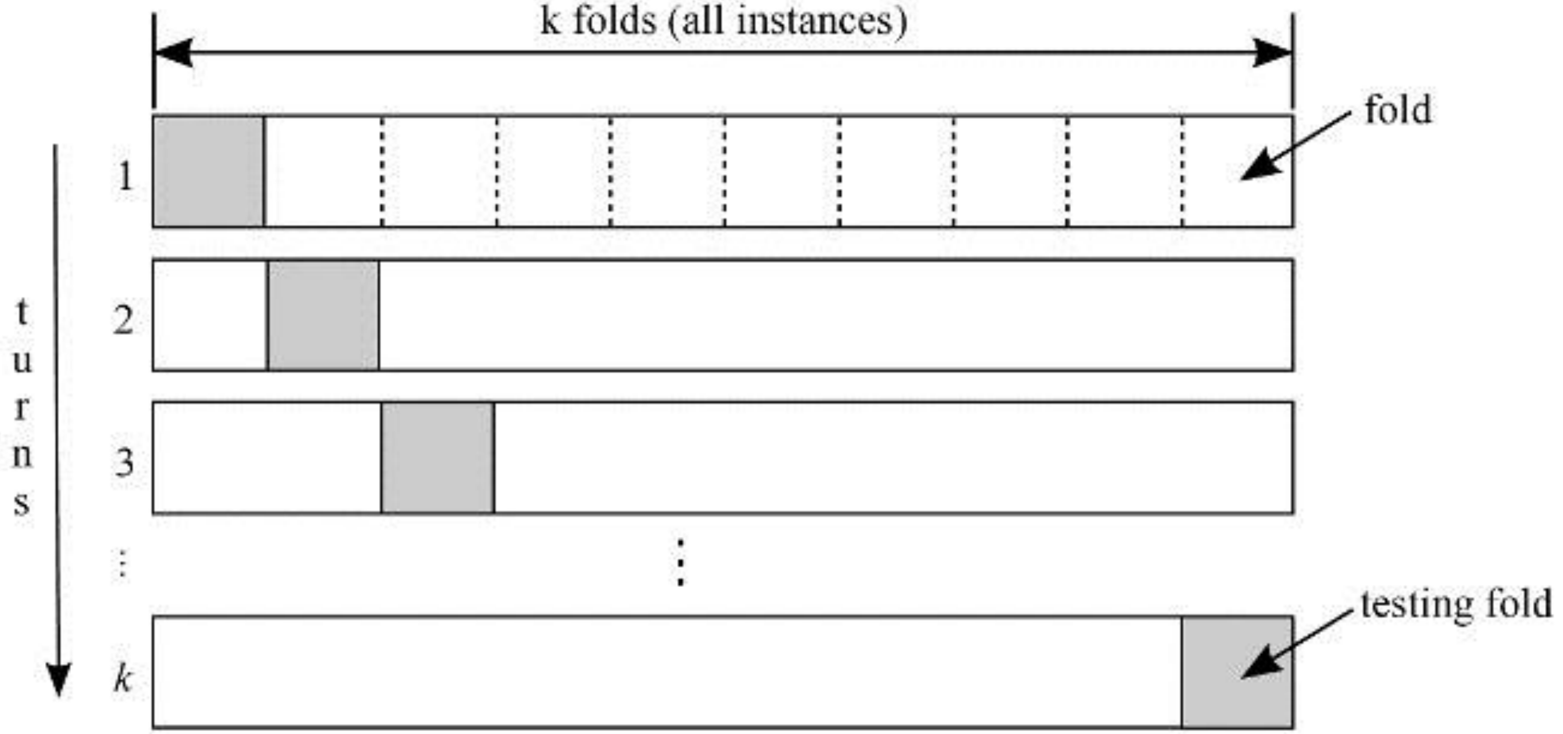
Namespace a　　Namespace b

Test and Training Data



Test and Training Data- Cnt.

- How to create test and training data?
  - Use k-fold cross validation, k=3 or 5



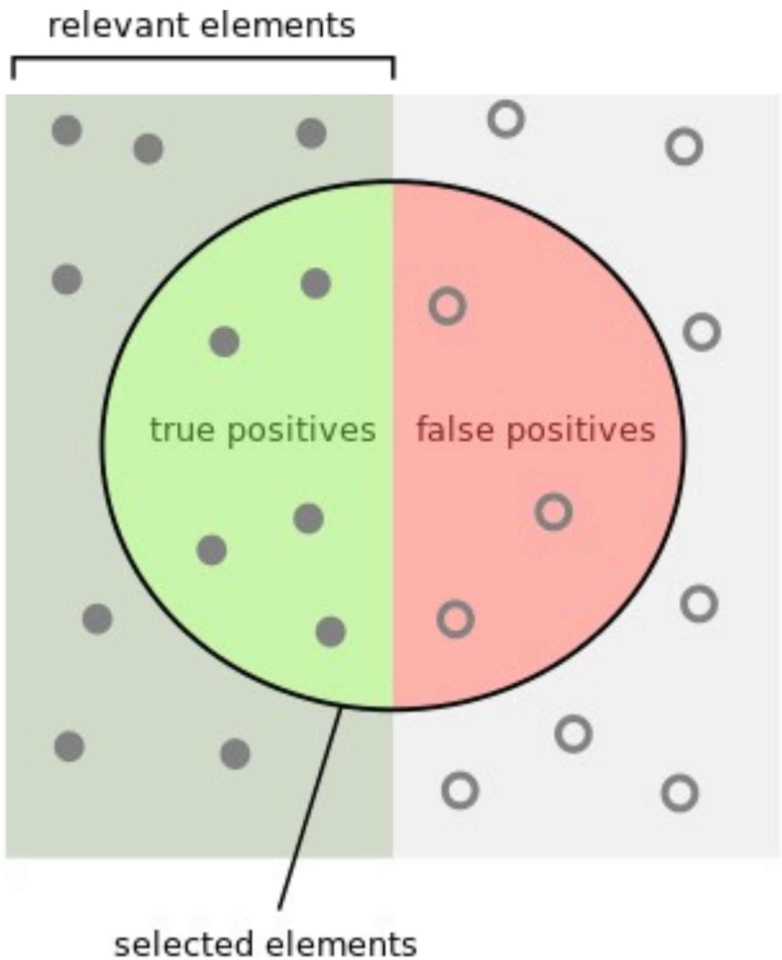
Evaluation

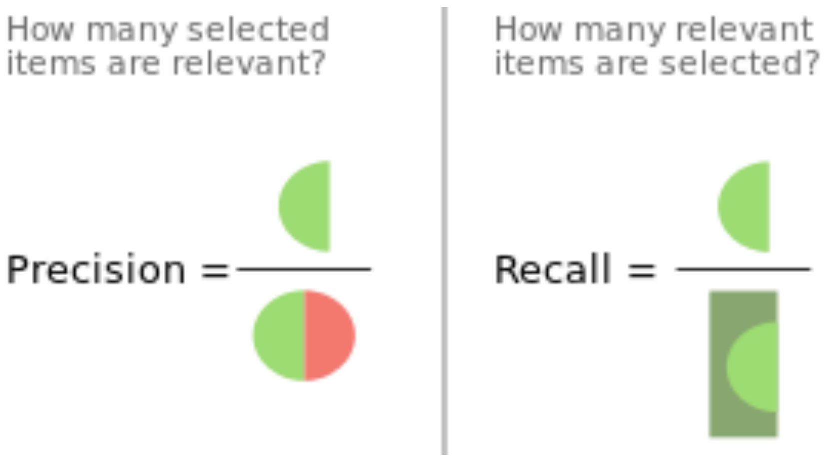
- Commonly-used evaluation metrics

Scoring	Function	Comment
Classification		
'accuracy'	`metrics.accuracy_score`	
'balanced_accuracy'	`metrics.balanced_accuracy_score`	
'average_precision'	`metrics.average_precision_score`	
'neg_brier_score'	`metrics.brier_score_loss`	
'f1'	`metrics.f1_score`	for binary targets
'f1_micro'	`metrics.f1_score`	micro-averaged
'f1_macro'	`metrics.f1_score`	macro-averaged
'f1_weighted'	`metrics.f1_score`	weighted average
'f1_samples'	`metrics.f1_score`	by multilabel sample
'neg_log_loss'	`metrics.log_loss`	requires `predict_proba` support
'precision' etc.	`metrics.precision_score`	suffixes apply as with 'f1'
'recall' etc.	`metrics.recall_score`	suffixes apply as with 'f1'
'jaccard' etc.	`metrics.jaccard_score`	suffixes apply as with 'f1'
'roc_auc'	`metrics.roc_auc_score`	
'roc_auc_ovr'	`metrics.roc_auc_score`	
'roc_auc_ovo'	`metrics.roc_auc_score`	
'roc_auc_ovr_weighted'	`metrics.roc_auc_score`	
'roc_auc_ovo_weighted'	`metrics.roc_auc_score`	
Clustering		
'adjusted_mutual_info_score'	`metrics.adjusted_mutual_info_score`	
'adjusted_rand_score'	`metrics.adjusted_rand_score`	
'completeness_score'	`metrics.completeness_score`	
'fowlkes_mallows_score'	`metrics.fowlkes_mallows_score`	
'homogeneity_score'	`metrics.homogeneity_score`	
'mutual_info_score'	`metrics.mutual_info_score`	
'normalized_mutual_info_score'	`metrics.normalized_mutual_info_score`	
'v_measure_score'	`metrics.v_measure_score`	
Regression		
'explained_variance'	`metrics.explained_variance_score`	
'max_error'	`metrics.max_error`	
'neg_mean_absolute_error'	`metrics.mean_absolute_error`	
'neg_mean_squared_error'	`metrics.mean_squared_error`	
'neg_root_mean_squared_error'	`metrics.mean_squared_error`	
'neg_mean_squared_log_error'	`metrics.mean_squared_log_error`	
'neg_median_absolute_error'	`metrics.median_absolute_error`	

Evaluation – Cnt.

- Precision, Recall, F1-Score

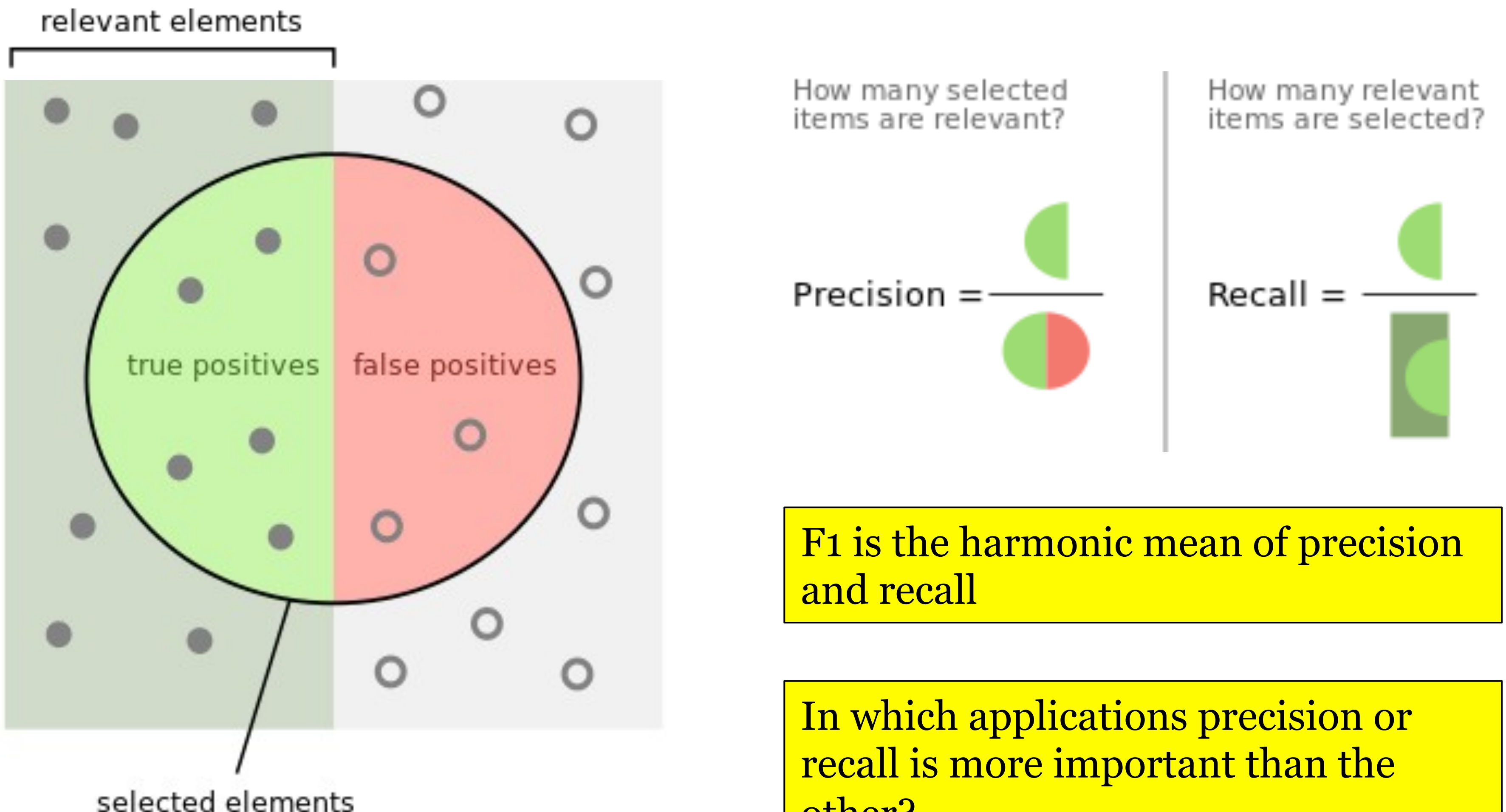




F1 is the harmonic mean of precision and recall

Evaluation – Cnt.

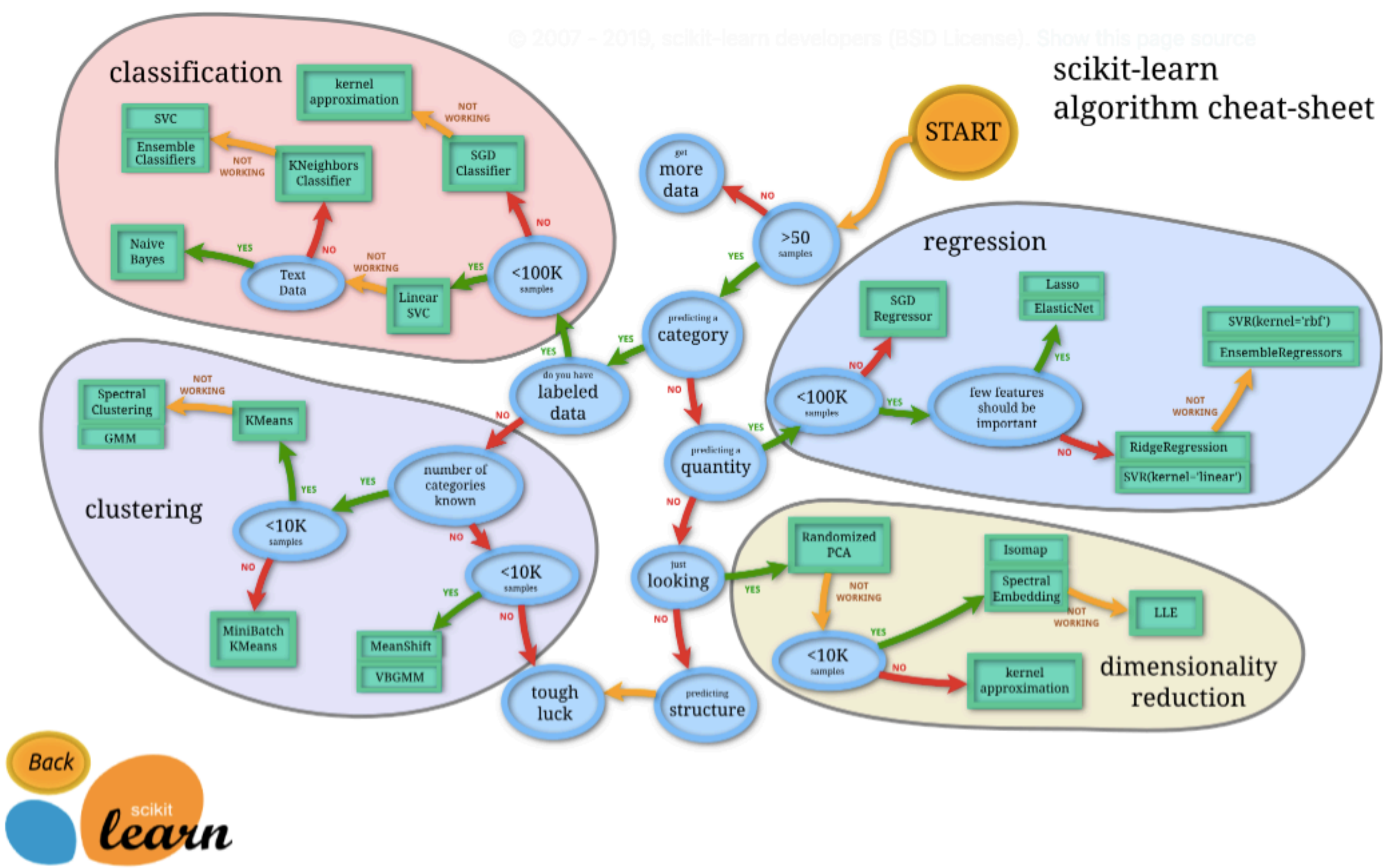
- Precision, Recall, F1-Score



F1 is the harmonic mean of precision and recall

In which applications precision or recall is more important than the other?

Quick Reference



Questions?