Text Classification

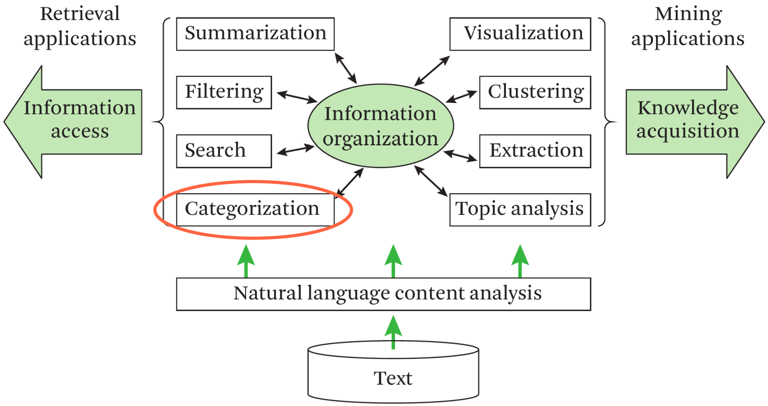
Marcel Bollmann

Department of Computer and Information Science (IDA)

LINKÖPING UNIVERSITY

Partly adapted from slides by Marco Kuhlmann.

Reminder: Conceptual framework



Zhai and Massung (2016)

Outline

1. Introduction
    - Examples
    - Statistical Pipeline
    - Neural Pipeline
2. Machine Learning
    - Training & Testing
    - Statistical Classifiers
    - Validation
3. Fine-Tuning Language Models
    - Language Modelling
    - BERT
    - Fine-Tuning BERT
4. Evaluation
    - The Importance of Baselines
    - Confusion Matrix
    - Precision/Recall/F1
    - Reporting Averages

Introduction to Text Classification

1

What is text classification?

Definition

Text classification is the task of categorizing text documents into predefined classes.

- We use the term *documents* to refer to text of any granularity.
  - social media posts
  - newspaper articles
  - entire books
  - single sentences
  - …

Example: Sentiment analysis

I love it so much! The mic works great!!!! I use it for online live classes, cosplay, and to look cute!! The lightup feature really works great! The app also works great too! The sound sounds amazing too! I just wish it had a case for when I travel.

positive

Not durable. The cord came apart from the audio adjuster. The saddest part is that happens only two months after it was purchased, and no force was applied. Definitely, I will not purchase and I do not recommend the item.

negative

Adapted from Amazon

Example: Topic classification

It took them an hour of huffing and puffing, but Arsenal did something at Stamford Bridge they hadn't managed since September – they scored an away goal in the Premier League.

- ✗ Business
- ✗ Politics
- ✗ Technology
- ✓ Sports
- ✗ Entertainment

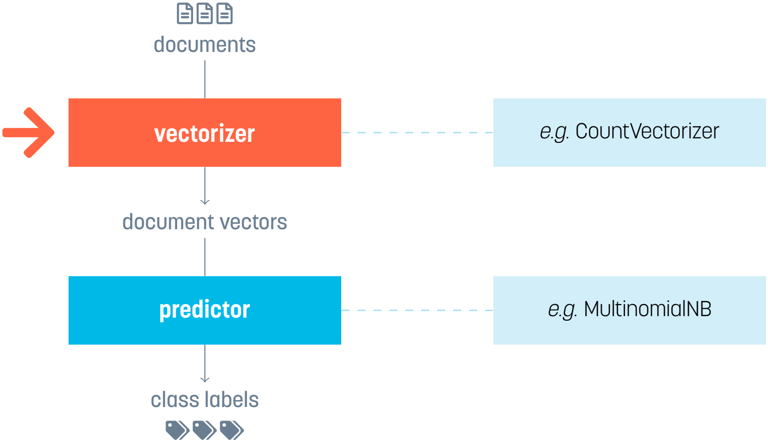
Quote source: The Guardian

Example: Forensic linguistics

> *I realized the faxed copy I just received was an outline of the manifesto, using much of the same wording, definitely the same topics and themes. … I invented [the language analysis] for this case and really, forensic linguistics took off after that.*
>
> — James Fitzgerald, profiler

Sources: Wikipedia & Newsweek

The statistical text classification pipeline



Reminder: Documents as tf-idf vectors

	Scandal in Bohemia	Final problem	Empty house	Norwood builder	Dancing men	Retired colourman
Adair	0.0000	0.0000	0.0692	0.0000	0.0000	0.0000
Adler	0.0531	0.0000	0.0000	0.0000	0.0000	0.0000
Lestrade	0.0000	0.0000	0.0291	0.1424	0.0000	0.0000
Moriarty	0.0000	0.0845	0.0528	0.0034	0.0000	0.0000

Documents as count vectors: Bag of words

It is a truth universally acknowledged, that a single man in possession of a good fortune must be in want of a wife.

Sentiment analysis with bag-of-words: vectors

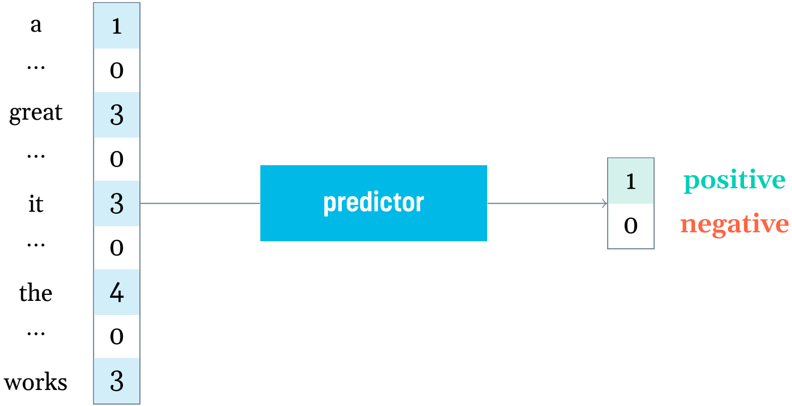
word	count
I	4
the	4
great	3
it	3
works	3
for	2
too	2
...	

positive

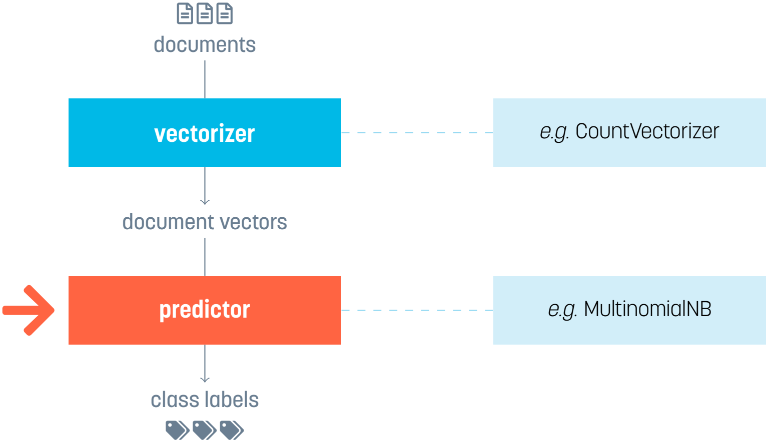
word	count
the	4
not	3
and	2
I	2
was	2
adjuster	1
after	1
...	

negative

Sentiment analysis with bag-of-words: pipeline



The statistical text classification pipeline



Install User Guide API Examples Community More

Go

Prev Up Next

scikit-learn 1.3.1
Other versions

Please **cite us** if you use the software.

User Guide
1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection
5. Visualizations
6. Dataset transformations
7. Dataset loading utilities
8. Computing with scikit-learn
9. Model persistence
10. Common pitfalls and recommended practices
11. Dispatching

Toggle Menu

User Guide

1. Supervised learning

- 1.1. Linear Models
- 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- 1.4. Support Vector Machines
- 1.5. Stochastic Gradient Descent
- 1.6. Nearest Neighbors
- 1.7. Gaussian Processes
- 1.8. Cross decomposition
- 1.9. Naive Bayes
- 1.10. Decision Trees
- 1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking
- 1.12. Multiclass and multioutput algorithms
- 1.13. Feature selection
- 1.14. Semi-supervised learning
- 1.15. Isotonic regression
- 1.16. Probability calibration
- 1.17. Neural network models (supervised)

Drawbacks of count-based document representations

1. Count-based document representations can easily yield **tens of thousands of features**.
   - computational challenge
   - data sparsity — *most of the entries are zero*

2. Count-based representations are very **limited in capturing dependencies** between words in a sentence.
   - *"I did **not** have a **good** experience with this product"*

n-gram models

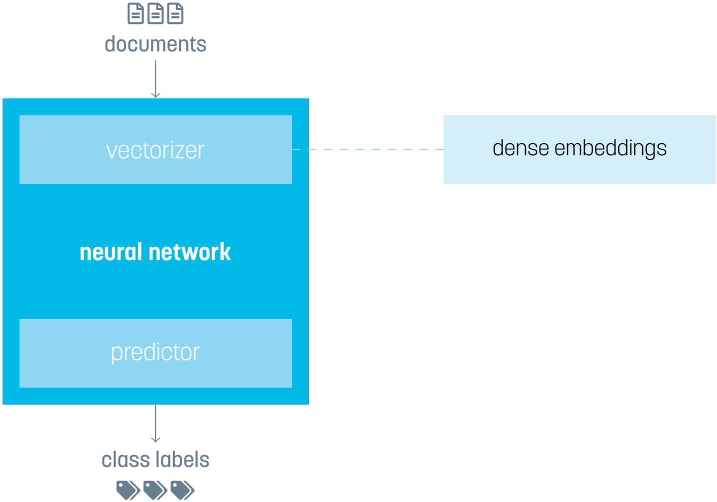
- One solution to problem 2 is to represent ***n*-grams** instead.
  - *n*-gram: a sequence of *n* consecutive tokens in a text

"I do not recommend the item"

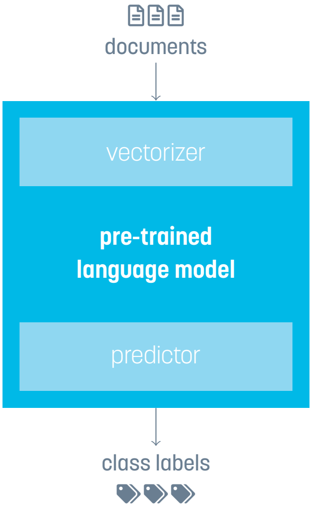
Unigrams	$n = 1$	*I, do, not, recommend, the, item*
Bigrams	$n = 2$	*I do, do not, not recommend, recommend the, the item*
Trigrams	$n = 3$	*I do not, do not recommend, not recommend the, recommend the item*

- Drawback: Very low frequency counts the higher *n* gets.

The neural text classification pipeline



Training a neural text classifier



- One option is to train a neural network "from scratch."
- It's usually more efficient to **fine-tune a pre-trained language model** instead.

Important concepts

- documents
- sentiment analysis
- bag of words, *n*-grams
- vectorizer + predictor pipeline

Text Classification as Machine Learning

2

Classification as machine learning

training set
learning

Hudson River (A)	Paul Klee (B)	Niche Retail (C)
Badger Mountain (A)	Linda Chapman (B)	Polk Brothers (C)

test set
evaluation

Shelf Glacier (A)	Charles Keating (B)	Aarti Industries (C)

Inspired by DBpedia14: A ∼ Natural Place; B ∼ Artist; C ∼ Company

Training and testing

training set
learning

Hudson River (A)	Paul Klee (B)	Niche Retail (C)
Badger Mountain (A)	Linda Chapman (B)	Polk Brothers (C)

When we train a classifier, we present it with a **document** x and its **gold-standard class** y and apply some **learning algorithm**.

Training and testing

When we evaluate a classifier, we present it with a **document** x and compare the **predicted class** with the **gold-standard class** y.

test set
evaluation

Document	Gold-standard class	Predicted class
Shelf Glacier	A	A
Charles Keating	B	B
Aarti Industries	C	A

Choosing a machine learning classifier

- No single type of classifier works best across *all* possible scenarios.
- Libraries like scikit-learn make it very easy to “switch out” one classifier for another:

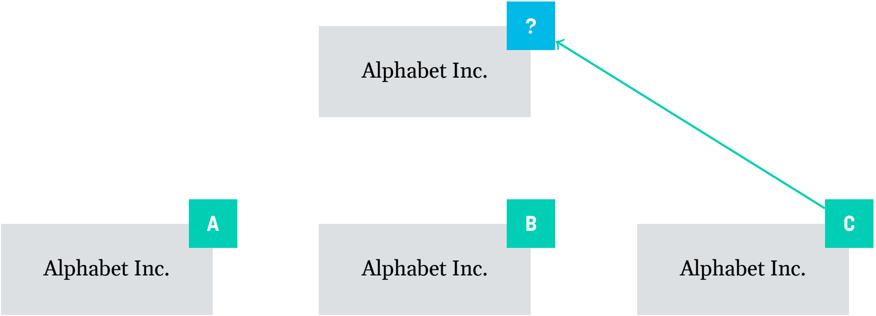
```
clf = make_pipeline(
            CountVectorizer(),
            MultinomialNB()
      )
clf.fit(X, y)
...
```

DecisionTreeClassifier
LogisticRegression
RidgeClassifier
SGDClassifier
SVC
...

Naive Bayes

- **Naive Bayes** is a simple probabilistic classifier.
  - Models $P(\text{class}, \text{document})$, then infers $P(\text{class}|\text{document})$ via Bayes' rule.
- It is **"naive"** in the sense that it makes strong (= unrealistic) independence assumptions about probabilities.
  - Its estimated probability values are usually not very accurate.
- For text classification, you typically want to use `MultinomialNB` from scikit-learn.

Naive Bayes decision rule (informally)



$\text{score}(A) =$
$P(\text{A})\, P(\text{Alphabet}|\text{A})\, P(\text{Inc.}|\text{A})$

10%

$\text{score}(B) =$
$P(\text{B})\, P(\text{Alphabet}|\text{B})\, P(\text{Inc.}|\text{B})$

20%

$\text{score}(C) =$
$P(\text{C})\, P(\text{Alphabet}|\text{C})\, P(\text{Inc.}|\text{C})$

70%

Generative vs. discriminative models

- **Logistic regression** is a simple discriminative classifier.
  - Models $P(\text{class}|\text{document})$ directly
- It is primarily used for **binary classification**.
  - But can be extended to multi-class classification as well
- Logistic regression can be considered the simplest form of a **neural network**.

The logistic model

- Logistic regression learns a **weight matrix** and **bias vector**.

$$\underbrace{\hat{\boldsymbol{y}} = f}_{\text{logistic function, e.g. softmax}}(\boldsymbol{x}\,\overbrace{\boldsymbol{W}}^{\text{weight matrix} \in \mathbb{R}^{n \times k}} + \overbrace{\boldsymbol{b}}^{\text{bias vector} \in \mathbb{R}^{k}})$$

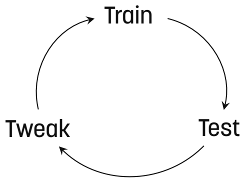
- It requires a **training algorithm** to find parameters that maximize the likelihood of the training data.
  - No closed-form solution like with Naive Bayes.

Finding the best classifier

- A **hyperparameter** is a parameter of a machine learning model that *you* need to set before the training starts.
  - *Example:* Naive Bayes has a "smoothing constant α"
- **Hyperparameter tuning** means trying different values to see which works "best."
  - Can have a big impact on your classifier's performance!
- Neural networks tend to have significantly more hyperparameters than statistical models, so tuning them becomes even more crucial.

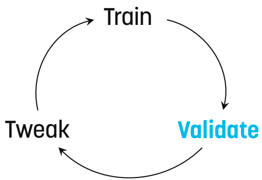
A problem when searching for the best classifier

- If you train & test repeatedly, your **test set** effectively **becomes part of the training** procedure!



Validating

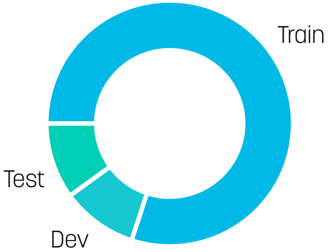
- One solution: use a separate **validation set**, also called **dev(elopment) set**, while tweaking the model.
  - Another solution is cross-validation.



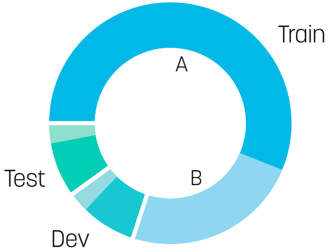
- Use the test set only for the **final evaluation**.

Creating a data split

- **Randomly shuffle & split** your data into train/dev/test partitions.
  - Typically, you want to reserve most of your data for training.



- **Stratified sampling** is a method of shuffling & splitting so that classes are represented equally in each split.



Important concepts

- train, test, & validation set
- Naive Bayes classifier
- logistic regression classifier
- hyperparameter tuning

Fine-Tuning Language Models for Text Classification

3

Language modelling

- What is the **probability** of a **sequence of words**?

$$p(\text{“I like books”}) > p(\text{“books I like”})$$
$$p(\text{“my comfort food is pizza”}) > p(\text{“my comfort food is chairs”})$$

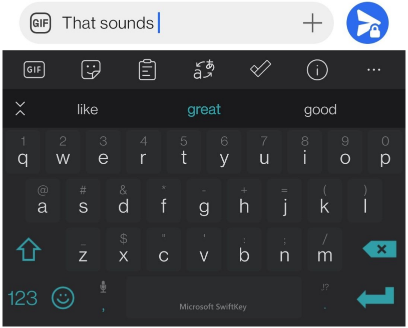
- What is the **conditional probability** of a word given context?

$$p(\text{“pizza”} | \text{“my comfort food is”}) = \ ?$$

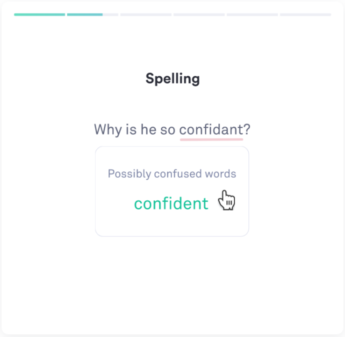
- These two formulations are equivalent.

Language modelling in practice

- Next word prediction



- Spelling correction



Source: Grammarly

Masked language modelling

1. *The capital of **Germany** is Berlin.* — WORLD KNOWLEDGE
2. *Kenya's athlete broke the world **record** in long jump.* — LEXICAL KNOWLEDGE
3. *I know this man, I've seen **him** before!* — CO-REFERENCE
4. *This movie was so **boring** that I almost fell asleep.* — SENTIMENT
5. *Yesterday we met **the/our** new neighbours.* — SYNTACTICAL CONSTRAINTS

Idea

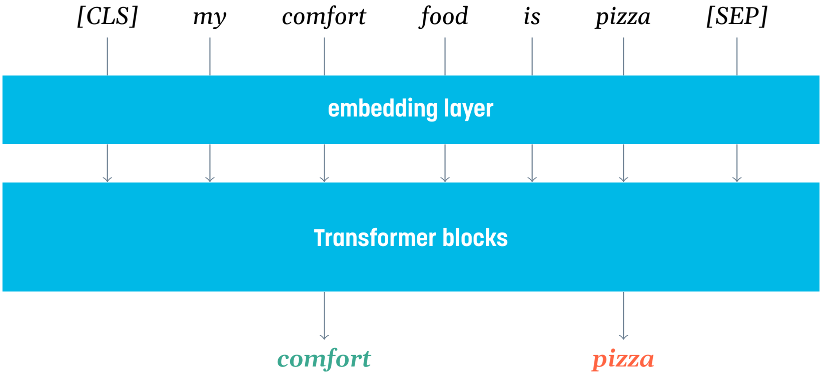
Training a neural network on the masked language modelling task will result in all of these types of knowledge being "encoded" in its parameters.

BERT: Bidirectional Encoder Representations from Transformers

- **BERT** is a neural network trained primarily on **masked language modelling**.
  - Based on the Transformer architecture
  - Outperformed all other models at the time of its release (2018)
- Spawned an entire "family" of similar models for specific tasks, languages, etc.
  - RoBERTa, NewsBERT, MusicBERT, CamemBERT, GottBERT, WangchanBERTa, BERTopic, ImageBERT, DistilBERT, TinyBERT, SpanBERT, ChemBERTa, ...
- Many of these models can be accessed through the **HuggingFace Model Hub**.

Devlin et al. (2019), List of BERT-related papers

Masked language modelling in BERT



Using BERT for text classification

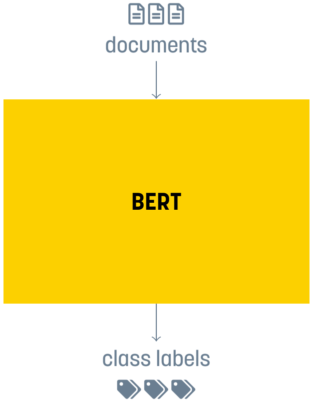
- Option 1: Feed a sentence into BERT and **extract embedding vectors** from it.
  - We could then use these vectors instead of bag-of-words to represent documents.

Idea

BERT is already a prediction model. We can simply **re-use** the entire model!

- Option 2: **Fine-tune the BERT model** on the target task.
  - *fine-tuning* = continuing to train, but with other data or tasks

Fine-tuning a BERT model



1. Download a pre-trained BERT model, e.g. from the **HuggingFace Model Hub**.
   - You can use any masked language model (MLM) here, but almost all of them are named after BERT.

2. Fine-tune it on your text classification task.
   - The **text classification guide** from HuggingFace is a good place to get started!

Advantages and downsides of using BERT models

- Much better at **encoding the meaning** of a linguistic expression.
  - Compared to e.g. counting words
- Pre-trained models exist for **almost any language & domain**.

- Need **more time and resources** to train.
  - Efficient fine-tuning requires a GPU
- Expressive power of neural networks **not always needed**!

Rule of thumb

Try "simple" methods first; if they don't work well, try fine-tuning a BERT model.

Important concepts

- (masked) language modelling
- BERT models
- fine-tuning

Evaluation of Text Classifiers

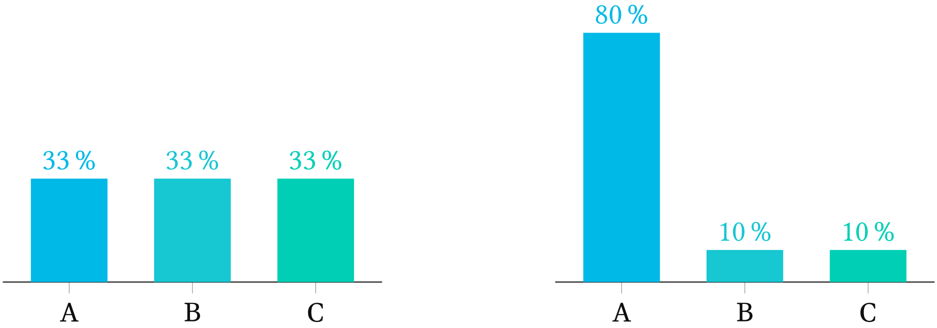
4

Accuracy

- **Accuracy** is the proportion of documents for which the classifier was “correct”.

$$\text{accuracy} = \frac{\text{\# of correctly classified documents}}{\text{\# of all documents}}$$

Is an accuracy of 80% good?



It depends!

The importance of baselines

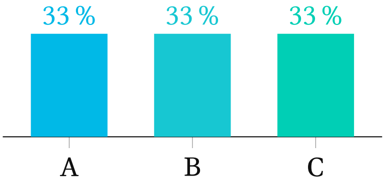
- Evaluation metrics are **no absolute measures** of performance.
  – What is "good" depends on the task!

If you hear:	You should ask:
"This classifier performs very well!"	*"...compared to what?"*

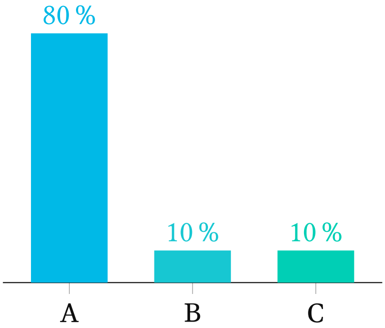
- We should judge a classifier's performance by **comparing it** against something else.
  – "Logistic regression achieves better accuracy than Naive Bayes."
- The point of comparison is often called the **baseline**.

Most-frequent-class baseline

- A simple baseline is to always predict the **most frequent class** in the training data.



A classifier with 80% accuracy could be pretty good here!



A classifier with 80% accuracy is not better than the MFC baseline…

Confusion matrix

		predicted		
		A	B	C
gold-standard	A	58	6	1
	B	5	11	2
	C	0	7	43

Documents labelled “B” in the gold-standard
where the model predicted “C”

Accuracy

	A	B	C
A	58	6	1
B	5	11	2
C	0	7	43

$$\text{accuracy} = \frac{\text{\# of correctly classified documents}}{\text{\# of all documents}}$$

Precision and recall

Precision and **recall** “zoom in” on how good a system is at identifying documents of a specific class.

Precision

When the model predicts class x, how often is it correct?

Recall

When the document has class x, how often does the model predict it?

- The proportion of correctly classified documents among all documents for which **the model predicts** class x.

- The proportion of correctly classified documents among all documents for which **the gold-standard class** is x.

Precision and recall with two classes

- Precision and recall are always computed **with respect to a class**.
- In a two-class setting, they are usually defined with respect to the **positive class**.
  - assumes two classes 'positive' and 'negative'

$$\text{precision} = \frac{\text{\# true positives}}{\text{\# true positives} + \text{\# false positives}}$$

$$\text{recall} = \frac{\text{\# true positives}}{\text{\# true positives} + \text{\# false negatives}}$$

Precision with respect to class B

	A	B	C
A	58	6	1
B	5	11	2
C	0	7	43

$$\text{precision} = \frac{\text{\# of true positives for “B”}}{\text{\# of all documents predicted to be “B”}}$$

Recall with respect to class B

	A	B	C
A	58	6	1
B	5	11	2
C	0	7	43

$$\text{recall} = \frac{\text{\# of true positives for “B”}}{\text{\# of all documents labelled “B” in the gold-standard}}$$

F1-measure

- A good system should **balance** between precision and recall.
- The **F1-measure** is the harmonic mean of the two values:

$$\text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Classification report in scikit-learn

	precision	recall	f1-score	support
C	0.63	0.04	0.07	671
KD	0.70	0.02	0.03	821
L	0.92	0.02	0.04	560
M	0.36	0.68	0.47	1644
MP	0.36	0.25	0.29	809
S	0.46	0.84	0.59	2773
SD	0.57	0.12	0.20	1060
V	0.59	0.15	0.24	950
accuracy			0.43	9288
macro avg	0.57	0.26	0.24	9288
weighted avg	0.52	0.43	0.34	9288

two ways of averaging!

Reporting averages

- The **macro average** is the arithmetic mean of the individual per-class scores.

$$\mathrm{F1}_{macro} = \frac{\sum_c \mathrm{F1}(c)}{\#\text{ of classes}}$$

- The **micro average** weights the per-class scores by the **number of documents** for the respective class.
  - also “weighted average”

$$\mathrm{F1}_{weighted} = \frac{\sum_c \mathrm{F1}(c) \cdot \boxed{\#(c)}}{\#\text{ of documents}}$$

Reporting averages: Intuition

- Macro average: *"each **class** is equally important"*

$$\mathrm{F1}_{macro} = \frac{\sum_c \mathrm{F1}(c)}{\text{\# of classes}}$$

- Micro average: *"each **document** is equally important"*

$$\mathrm{F1}_{weighted} = \frac{\sum_c \mathrm{F1}(c) \cdot \#(c)}{\text{\# of documents}}$$

Important concepts

- importance of baselines
- most-frequent class baseline
- confusion matrix
- precision, recall, F1-score (in multi-class classification)
- macro & micro/weighted average

LINKÖPING
UNIVERSITY