Machine Learning Techniques for Text

Module 2: Detecting Spam Emails

Dr. Nikos Tsourakis

Course outline

- Module 0: Python Crash Course
- Module 1: Intro to Machine Learning
- **Module 2: Detecting Spam Emails**
- Module 3: Classifying Topics of Newsgroup Posts
- Module 4: Extracting Sentiments from Product Reviews
- Module 5: Recommending Music Titles
- Module 6: Teaching Machines to Translate
- Module 7: Summarizing Wikipedia Articles
- Module 8: Detecting Hateful and Offensive Language
- Module 9: Generating Text in Chatbots
- Module 10: Clustering Speech-to-Text Transcriptions

Overview

- Electronic mail is one of the most ubiquitous Internet services to exchange messages asynchronously
- The problem in this communication scheme is identifying and blocking unsolicited and unwanted messages
- We deal with this problem from a machine learning perspective and unfolds as a series of steps for developing and evaluating a typical ***spam detector***
  - We elaborate on the limitations of performing spam detection using traditional programming
  - Next, we introduce the basic techniques for text representation and preprocessing
  - Finally, we implement two classifiers using an open-source dataset and evaluate their performance on standard metrics

Module objectives

After completing this module, you should be able to:

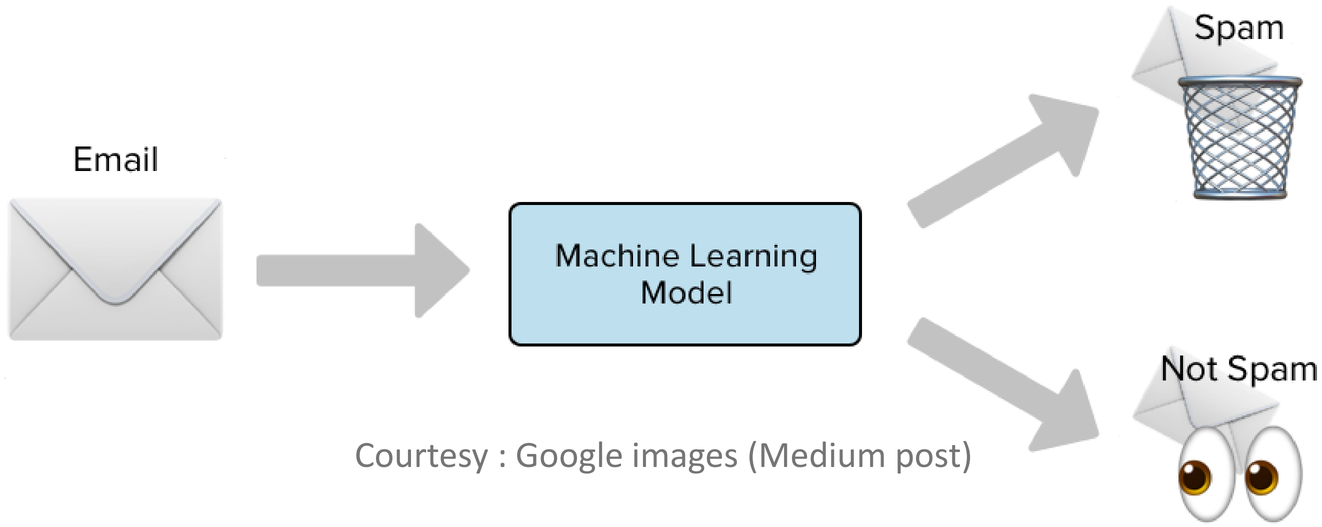
- Obtaining the data
- Understanding its content
- Preparing the datasets for the analysis
- Training the classification models
- Realizing the tradeoffs of the algorithms
- Assessing the performance of the models

Machine Learning Techniques for Text

Section 1: Understanding spam detection

Spam detection

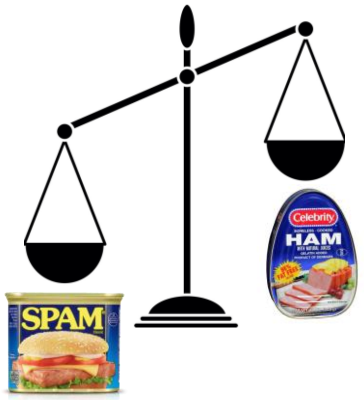
- Software inside the mail server that decides whether an incoming email is spam or not
- Without a spam detector the inbox will be flooded with irrelevant and possibly dangerous correspondence
- A serious technical, economic and even social threat!
- If you were to implement a detector program from scratch how would you do it?



Courtesy : Google images (Medium post)

Spam detection

- How can we build a spam detector for the following email?



From:	kmitnick@jakqd.com
To:	tjones@tsourakis.net
Subject:	Urgent!

Dear MR tjones,

You have noticed lately that your laptop is runing slow! This is because I gained access to your machine, and I installed a harmful VIRUS!!!

Even if you change your password my virus CANNOT BE intercept!

The only SOLUTION is to following my instractions here: https://bit.ly/33rhdNM

You have 48 hour before the virus is activated! ...OTHERWISE...GOOD LUCK!!!!

Spam detection

- Let's try to think of possible strategies for the detector

From:	kmitnick@jakqd.com
To:	tjones@tsourakis.net
Subject:	Urgent!

Dear MR tjones,

You have noticed lately that your laptop is runing slow! This is because I gained access to your machine, and I installed a harmful VIRUS!!!

Even if you change your password my virus CANNOT BE intercept!

The only SOLUTION is to following my instractions here: https://bit.ly/33rhdNM

You have 48 hour before the virus is activated! ...OTHERWISE...GOOD LUCK!!!!

Spam detection

- Let’s try to think of possible strategies for the detector

```
From:     kmitnick@jakqd.com
To:       tjones@tsourakis.net
Subject:  Urgent!
```

```
Dear MR tjones,

You have noticed lately that your laptop is runing slow!
This is because I gained access to your machine, and I
installed a harmful VIRUS!!!

Even if you change your password my virus CANNOT BE
intercept!

The only SOLUTION is to following my instractions here:
https://bit.ly/33rhdNM

You have 48 hour before the virus is activated!
...OTHERWISE...GOOD LUCK!!!!
```

➢ **T1** – The text in the subject field is typical for spam. It is characterized by a manipulative style that creates unnecessary urgency and pressure

Spam detection

- Let's try to think of possible strategies for the detector

```
From:     kmitnick@jakqd.com
To:       tjones@tsourakis.net
Subject:  Urgent!
```

```
Dear MR tjones,

You have noticed lately that your laptop is runing slow!
This is because I gained access to your machine, and I
installed a harmful VIRUS!!!

Even if you change your password my virus CANNOT BE
intercept!

The only SOLUTION is to following my instractions here:
https://bit.ly/33rhdNM

You have 48 hour before the virus is activated!
...OTHERWISE...GOOD LUCK!!!!
```

- **T2** – The message begins with the phrase *Dear MR tjones*. The last word was probably extracted automatically from the recipient's email address

Spam detection

- Let's try to think of possible strategies for the detector

```
From:     kmitnick@jakqd.com
To:       tjones@tsourakis.net
Subject:  Urgent!
```

```
Dear MR tjones,

You have noticed lately that your laptop is runing slow!
This is because I gained access to your machine, and I
installed a harmful VIRUS!!!

Even if you change your password my virus CANNOT BE
intercept!

The only SOLUTION is to following my instractions here:
https://bit.ly/33rhdNM

You have 48 hour before the virus is activated!
...OTHERWISE...GOOD LUCK!!!!
```

- **T3** – Bad spelling and the incorrect use of grammar are potential spam indicators

Spam detection

- Let’s try to think of possible strategies for the detector

```
From:     kmitnick@jakqd.com
To:       tjones@tsourakis.net
Subject:  Urgent!
```

```
Dear MR tjones,

You have noticed lately that your laptop is runing slow!
This is because I gained access to your machine, and I
installed a harmful VIRUS!!!

Even if you change your password my virus CANNOT BE
intercept!

The only SOLUTION is to following my instractions here:
https://bit.ly/33rhdNM

You have 48 hour before the virus is activated!
...OTHERWISE...GOOD LUCK!!!!
```

➢ **T4** – The text in the body of the message contains sequences with multiple punctuation marks or capital letters

Write the code

- We can now implement a program taking into account the four triggers
- When a new email arrives, the detector runs the following code and decides whether it is a spam

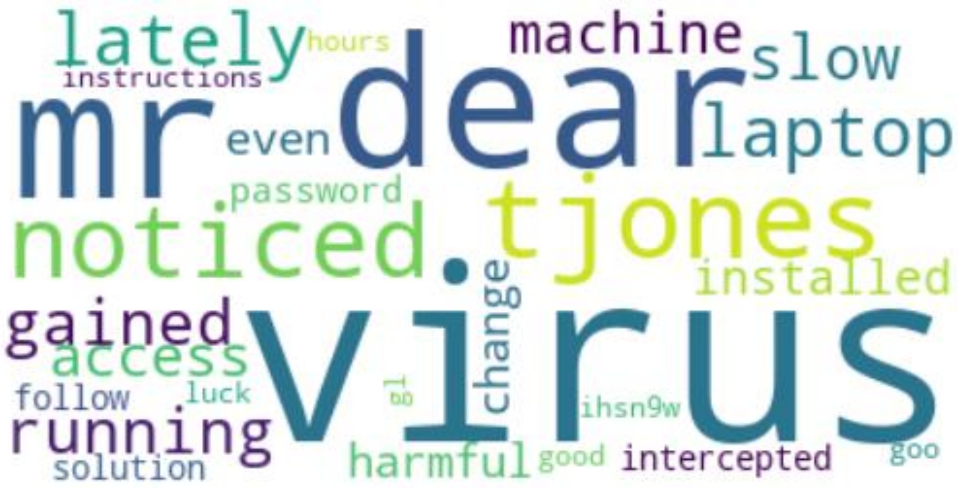
```
IF (subject is typical for spam)
    AND IF (message uses recipients email address)
        AND IF (spelling and grammar errors)
            AND IF (multiple sequences of marks-caps) THEN
                print("It's a SPAM!")
```

***Pseudocode** is an informal high-level description of a program or an algorithm.*

- Obviously, this is not the best detector ever built!

Add more triggers

- Words can also serve as part of our separation criteria
- Visualize the text data using ***word clouds*** (also known as tag clouds)



- The image suggests that the most common word in our spam message is "**virus**"

Add more triggers

- Next, we adapt the pseudocode

```
...
      AND IF (multiple sequences of marks-caps) THEN
          AND IF (common word = "virus") THEN
              print("It's a SPAM!")
```

Add more triggers

- Next, we adapt the pseudocode

```
...
      AND IF (multiple sequences of marks-caps) THEN
          AND IF (common word = "virus") THEN
              print("It's a SPAM!")
```

- Is this new version of the program better?

Add more triggers

- Next, we adapt the pseudocode

```
     ...
          AND IF (multiple sequences of marks-caps) THEN
               AND IF (common word = "virus") THEN
                    print("It's a SPAM!")
```

- Is this new version of the program better?
  - Slightly ☹
  - We can engineer even more criteria, but the problem becomes insurmountable at some point

Add more triggers

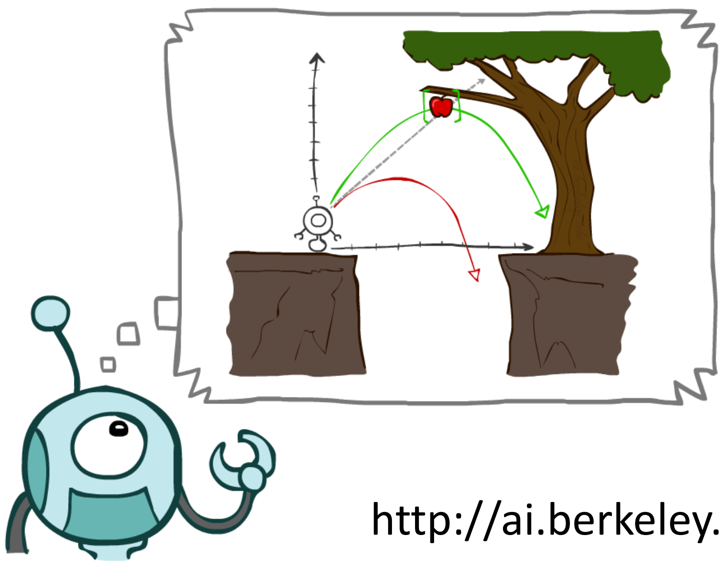
- Next, we adapt the pseudocode

```
    ...
            AND IF (multiple sequences of marks-caps) THEN
                AND IF (common word = "virus") THEN
                    print("It's a SPAM!")
```

- Is this new version of the program better?
  - Slightly ☹
  - We can engineer even more criteria, but the problem becomes insurmountable at some point
- This is where machine learning algorithms become handy and can help us to solve this problem

Learn by example

- As humans learn by example; touching something very hot (input) causes pain (output), we can train a machine to read emails (input) and classify them automatically into spam/non-spam (output)



http://ai.berkeley.edu/

Feature engineering

- The input to the learning algorithm was not the actual emails text but some kind of information extracted from those (***T1***-***T4***)
- These are called ***features*** and the process of creating them is called ***feature engineering***
- Identifying a suitable list of features for any ML task requires ***domain knowledge*** – comprehending the problem you want to solve in-depth.
- How you choose them directly impacts the algorithm's performance and determines its success to a significant degree:
  - certain features can overlap with others
  - specific features might be less relevant

Feature engineering

- Let’s see a few examples and propose good features

Problem	Features
Classify movie reviews as positive or negative	[redacted]
Separate images that contain either forests or the sea	[redacted]
Classify flowers into specific categories	[redacted]
Identify angry customers from telephone recordings	[redacted]
Predict the value of an apartment in a specific area	[redacted]

Feature engineering

- Let's see a few examples and propose good features

Problem	Features
Classify movie reviews as positive or negative	Count the number of times words such as `good`, `excellent`, `bad`, or `horrible` appear in a review
Separate images that contain either forests or the sea	[redacted]
Classify flowers into specific categories	[redacted]
Identify angry customers from telephone recordings	[redacted]
Predict the value of an apartment in a specific area	[redacted]

Feature engineering

- Let’s see a few examples and propose good features

Problem	Features
Classify movie reviews as positive or negative	Count the number of times words such as `good`, `excellent`, `bad`, or `horrible` appear in a review
Separate images that contain either forests or the sea	Count the number of pixels in an image with almost green colors versus blue ones
Classify flowers into specific categories	[redacted]
Identify angry customers from telephone recordings	[redacted]
Predict the value of an apartment in a specific area	[redacted]

Feature engineering

- Let's see a few examples and propose good features

Problem	Features
Classify movie reviews as positive or negative	Count the number of times words such as `good`, `excellent`, `bad`, or `horrible` appear in a review
Separate images that contain either forests or the sea	Count the number of pixels in an image with almost green colors versus blue ones
Classify flowers into specific categories	Calculate the sepal length, sepal width, petal length, and petal width of each flower
Identify angry customers from telephone recordings	[redacted]
Predict the value of an apartment in a specific area	[redacted]

Feature engineering

- Let's see a few examples and propose good features

Problem	Features
Classify movie reviews as positive or negative	Count the number of times words such as `good`, `excellent`, `bad`, or `horrible` appear in a review
Separate images that contain either forests or the sea	Count the number of pixels in an image with almost green colors versus blue ones
Classify flowers into specific categories	Calculate the sepal length, sepal width, petal length, and petal width of each flower
Identify angry customers from telephone recordings	Calculate the mean amplitude of the recorded signal
Predict the value of an apartment in a specific area	[redacted]

Feature engineering

- Let’s see a few examples and propose good features

Problem	Features
Classify movie reviews as positive or negative	Count the number of times words such as `good`, `excellent`, `bad`, or `horrible` appear in a review
Separate images that contain either forests or the sea	Count the number of pixels in an image with almost green colors versus blue ones
Classify flowers into specific categories	Calculate the sepal length, sepal width, petal length, and petal width of each flower
Identify angry customers from telephone recordings	Calculate the mean amplitude of the recorded signal
Predict the value of an apartment in a specific area	Extract the size in square meters, number of rooms, floor number, and postal code of each apartment

Feature engineering

- Let’s see a few examples and propose good features

Problem	Features
Classify movie reviews as positive or negative	Count the number of times words such as `good`, `excellent`, `bad`, or `horrible` appear in a review
Separate images that contain either forests or the sea	Count the number of pixels in an image with almost green colors versus blue ones
Classify flowers into specific categories	Calculate the sepal length, sepal width, petal length, and petal width of each flower
Identify angry customers from telephone recordings	Calculate the mean amplitude of the recorded signal
Predict the value of an apartment in a specific area	Extract the size in square meters, number of rooms, floor number, and postal code of each apartment

- But as we are dealing with text, and we need pertinent features

Machine Learning Techniques for Text

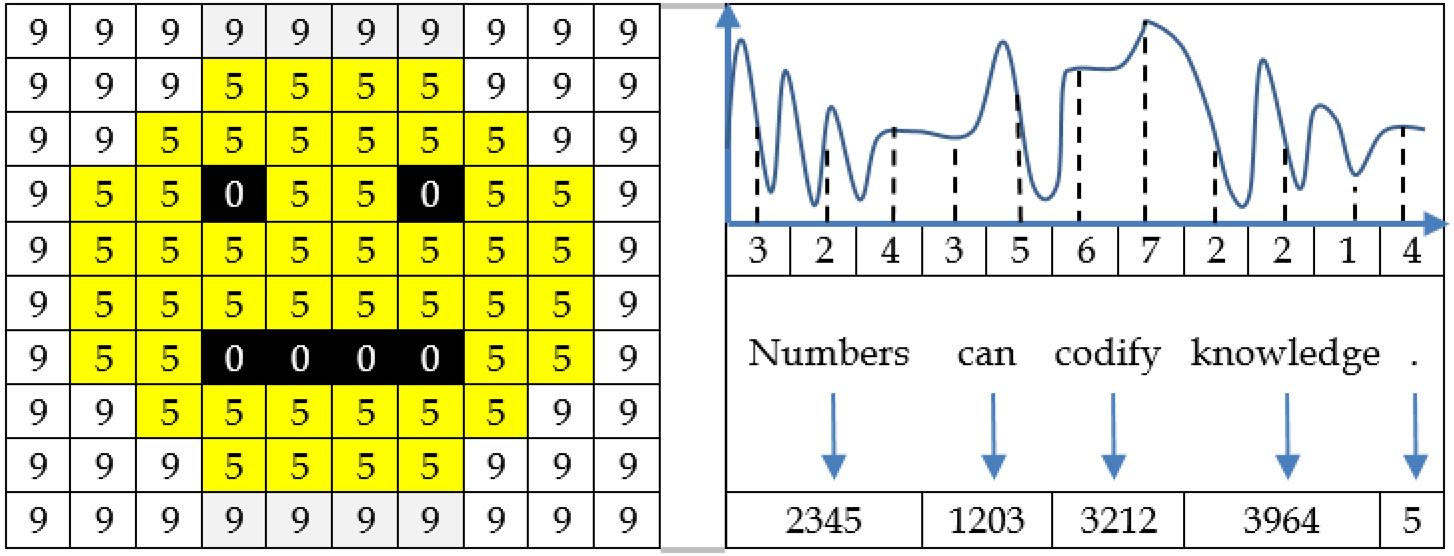
Section 2: Extracting word representations

Data representation

- What does a word mean to a computer? What about an image or an audio file?
  - To put it simply, nothing!
- A computer circuit can only process signals that contain two voltage levels or states, similar to an on-off switch
- This representation is the well-known binary system where every quantity is expressed as a sequence of ***1s*** (high voltage) and ***0s*** (low voltage)
- All data should be represented numerically

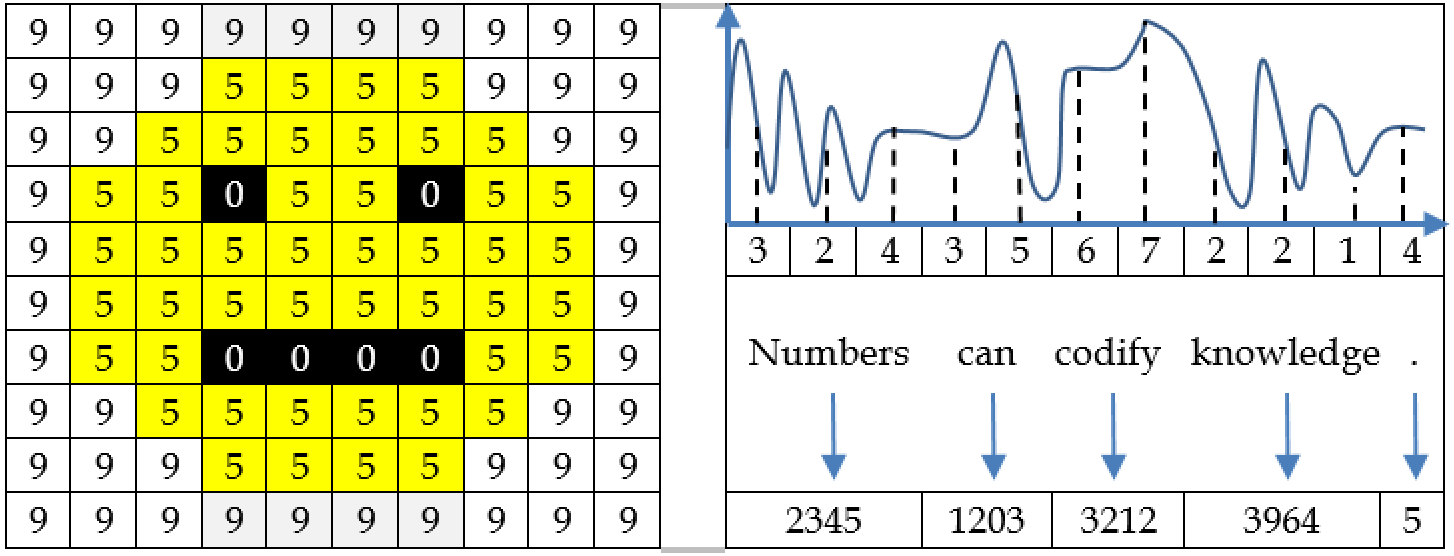
Word representations

- Numbers can codify a word, the pixels of an image, the audio samples of an audio file, etc.



Word representations

- Numbers can codify a word, the pixels of an image, the audio samples of an audio file, etc.



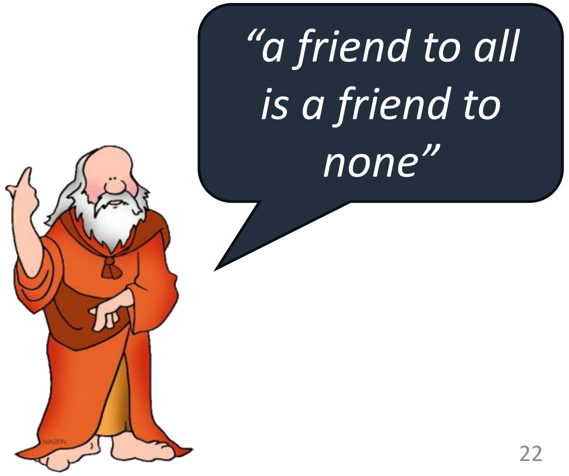
- There are various ways to represent words in machine learning problems

Using label encoding

- Consider this quote from Aristotle *“a friend to all is a friend to none”*
- Using ***label encoding*** we can produce the following mapping:

Input								
a	friend	to	all	is	a	friend	to	none
0	2	5	1	3	0	2	5	4

- This representation, however, implies some short of order (because 0 < 2 < 5), which is not true



Using one-hot encoding

- One-hot encoding codifies every word as a vector with zeros and a single one
  - no two words exist with the same one-hot vector

	Input								
Unique words	A	friend	to	all	is	a	friend	to	none
a	1	0	0	0	0	1	0	0	0
all	0	0	0	1	0	0	0	0	0
friend	0	1	0	0	0	0	1	0	0
is	0	0	0	0	1	0	0	0	0
none	0	0	0	0	0	0	0	0	1
to	0	0	1	0	0	0	0	1	0

Using one-hot encoding

- One-hot encoding codifies every word as a vector with zeros and a single one
  - no two words exist with the same one-hot vector

	Input								
Unique words	A	friend	to	all	is	a	friend	to	none
a	1	0	0	0	0	1	0	0	0
all	0	0	0	1	0	0	0	0	0
friend	0	1	0	0	0	0	1	0	0
is	0	0	0	0	1	0	0	0	0
none	0	0	0	0	0	0	0	0	1
to	0	0	1	0	0	0	0	1	0

- The majority of the elements in the array are zeros and can pose challenges due to the memory required to store them

Using one-hot encoding

- One-hot encoding codifies every word as a vector with zeros and a single one
    - no two words exist with the same one-hot vector

	Input								
Unique words	A	friend	to	all	is	a	friend	to	none
a	1	0	0	0	0	1	0	0	0
all	0	0	0	1	0	0	0	0	0
friend	0	1	0	0	0	0	1	0	0
is	0	0	0	0	1	0	0	0	0
none	0	0	0	0	0	0	0	0	1
to	0	0	1	0	0	0	0	1	0

tip

Matrixes of this kind are called sparse

- The majority of the elements in the array are zeros and can pose challenges due to the memory required to store them

Using token count encoding

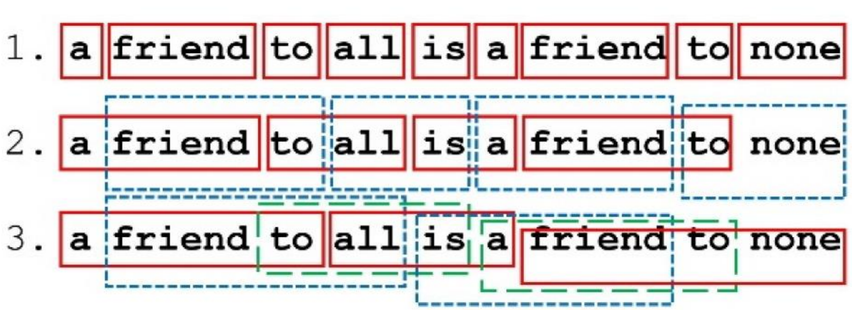
- Also known as ***bag-of-words*** (BoW) counts the absolute frequency of each word within a sentence or a document
- The input is represented as a bag of words without taking into account grammar or word order
- Create a table (***Term Document Matrix***) where each cell represents the number of times a word from the vocabulary appears the quote

Unique words							
a	all	friend	is	me	none	to	world
2	1	2	1	0	1	2	0

“a friend to all is a friend to none”

N-grams as tokens

- Certain word combinations are more frequent than others in any human language
- Consist of a single word (***unigrams***), two words (***bigrams***), three words (***trigrams***), etc.



1. Unigrams	2. Bigrams	3. Trigrams
a	a friend	a friend to
all	friend to	friend to all
friend	to all	to all is
is	all is	all is a
none	is a	is a friend
to	to none	friend to none

Using tf-idf encoding

- One limitation of BoW representations is that they do not consider the value of words inside the corpus
- If solely frequency were of prime importance, articles such as ***a*** or ***the*** would provide the most information for a document
- The ***term frequency-inverse document frequency*** (tf-idf) encoding scheme:
  - penalizes these frequent words
  - allows us to weigh each word in the text
- Heuristic where more common words tend to be less relevant for most semantic classification tasks, and the weighting reflects this approach

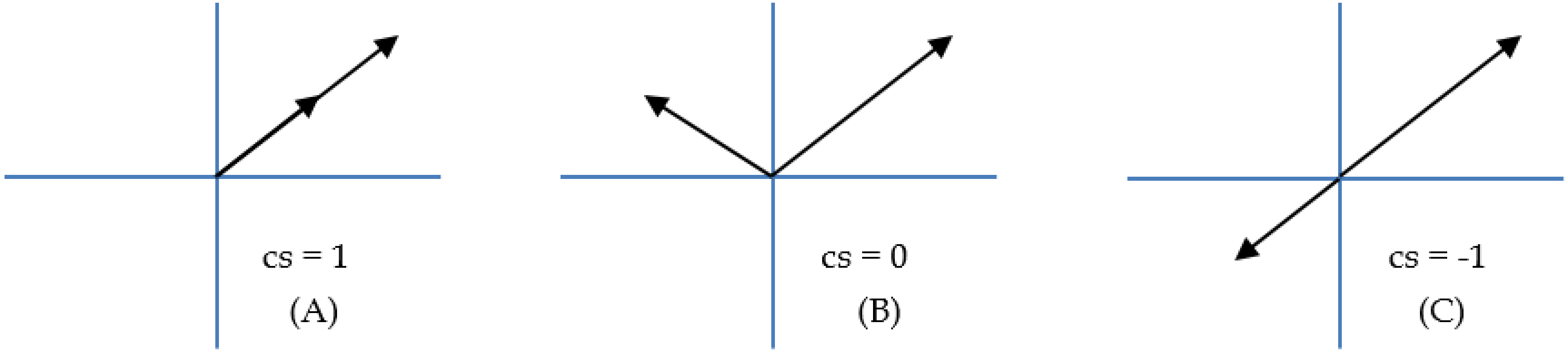
Using tf-idf encoding

- From a virtual dataset with 10 million emails, we randomly pick one containing 100 words
- Suppose that the word ***virus*** appears three times in this email, so its term frequency (tf) is: $\frac{3}{100} = 0.03$
- Moreover, the same word appears in 1000 emails in the corpus, so the inverse document frequency (idf) is: $\log(\frac{10000000}{1000}) = 4$
- The tf-idf weight is simply the product of these two statistics:

  0.03 * 4 = 0.12

Calculating vector similarity

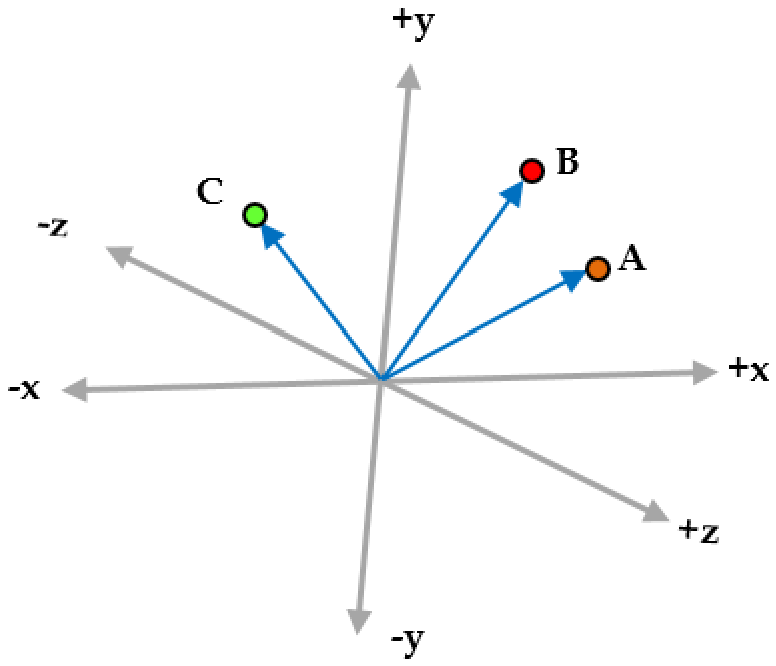
- ***Cosine similarity*** is the degree to which two vectors point in the same direction
- It targets orientation, rather than magnitude
- When the two vectors are aligned to the same direction cosine similarity is ***1***, when they are perpendicular it is ***0***, and when they point in opposite directions it is ***-1***



Cosine similarity

- Which vectors are more similar (A, B or C)?

A=(4, 4, 4),
B=(1, 7, 5)
C=(-5, 5, 1)



Cosine similarity

- Given that: $\cos(V, U) = \frac{V \bullet \mathrm{U}}{\|V\| \times \|U\|}$
- A=(4, 4, 4), B=(1, 7, 5) and C=(-5, 5, 1)
- We have:

$$A \bullet B = 4 \times 1 + 4 \times 7 + 4 \times 5 = 52$$
$$A \bullet C = 4 \times (-5) + 4 \times 5 + 4 \times 1 = 4$$
$$B \bullet C = 1 \times (-5) + 7 \times 5 + 5 \times 1 = 35$$

$\|A\| = \sqrt{4^2 + 4^2 + 4^2} = \sqrt{48}$, $\|B\| = \sqrt{75}$ and $\|C\| = \sqrt{51}$

- We obtain:

$\cos(A, B) = \frac{52}{\sqrt{48} \times \sqrt{75}} \cong 0.87$, $\cos(A, C) = \frac{4}{\sqrt{48} \times \sqrt{51}} \cong 0.08$ and $\cos(B, C) = \frac{35}{\sqrt{75} \times \sqrt{51}} \cong 0.57$

Machine Learning Techniques for Text

Section 3: Executing data preprocessing

Tokenizing the input

- During ***tokenization*** we split textual data into smaller components called ***tokens***
- These can be words, phrases, symbols, or other meaningful elements
- We can use ***regular expressions*** (regexp) that can assist with the creation of a tokenizer

“The best course ever taught about Text Mining”

↓

“The”, “best”, “course”, “ever”, “taught”, “about”, “Text”, “Mining”

Understanding regular expressions

- ***Regular expressions*** (regexp) are used to:
  - find a string in a document
  - replace part of the text with something else
  - examine the conformity of some textual input

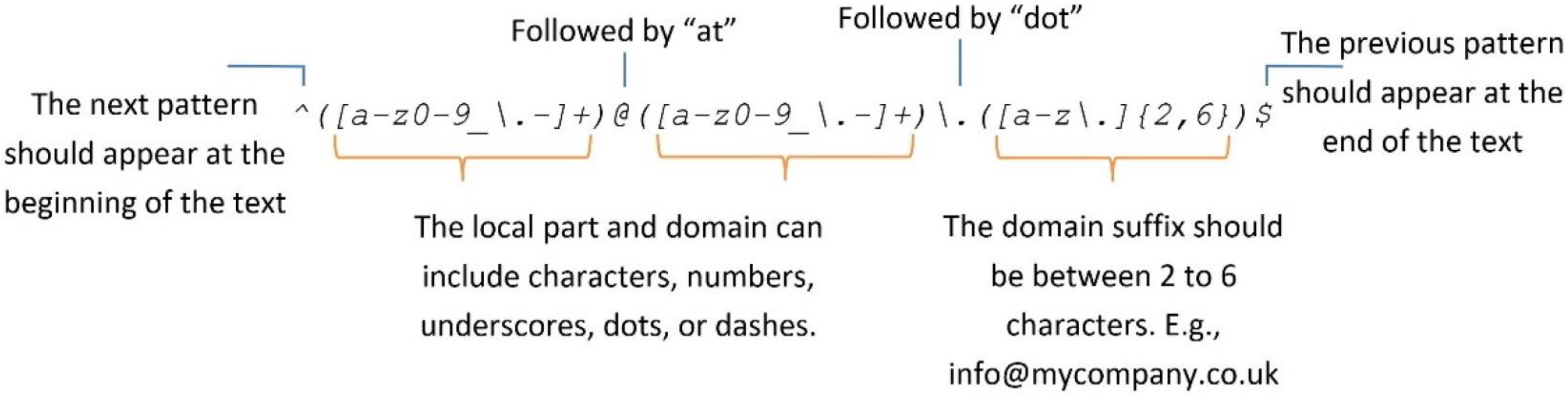
Understanding regular expressions

- ***Regular expressions*** (regexp) are used to:
  - find a string in a document
  - replace part of the text with something else
  - examine the conformity of some textual input

regexp	**matches**
[A-Za-z0-9]	Any character, letter, or digit, in the range A to Z, a to z, or 0 to 9 – for example, `G`.
[A-Za-z0-9]+	Any sequence of characters, letters, or digits, in the range A to Z, a to z, or 0 to 9. The plus sign signifies at least one occurrence – for example, `Hello2You`.
[a-z]{2,5}	Any sequence of lowercase letters containing two to five characters – for example, `hello`.
^([0-9][a-z]+)	Any sequence of characters beginning with a number and followed by lowercase letters – for example, `4ever`.
([A-Z]+[0-9])$	Any sequence of characters with uppercase letters that ends with a number – for example, `ME2`.

Understanding regular expressions

- Check the validity of an email address



\. = the dot character needs to be escaped otherwise it will match any character
[ ] = defines character ranges
( ) = defines a group

Removing stop words

- A typical task during the preprocessing phase is removing all the words that presumably help us focus on the most important information in the text
- These are called ***stop words*** and there is no universal list in English or any other language
- Examples of stop words include determiners (such ***as*** another and ***the***), conjunctions (such as ***but*** and ***or***), and prepositions (such as ***before*** and ***in***)

tokenization *"The", "best", "course", "ever", "taught", "about", "Text", "Mining"*

↓ ↓

stop word removal *"best", "course", "ever", "taught", "Text", "Mining"*

Stemming the words

- Map words with the same core meaning to a central word or symbol
- During ***stemming*** we cut off the end (suffix) or the beginning (prefix) of an inflected word and ending up with its stem (the root word)
- For example, the stem of the word ***plays*** is ***play***

stop word removal — *"best", "course", "ever", "taught", "Text", "Mining"*

↓

stemming or/and lemmatization — *"best", "course", "ever", "teach", "text", "mine"*

Lemmatizing the words

- Another approach for reducing the inflectional forms of a word to a base root is called ***lemmatization***
- The method performs morphological analysis of the word and obtains its proper lemma (the base form under which it appears in a dictionary)
- For example, the lemma of ***led*** is ***lead***
- Lemmatization differs from stemming, as it requires detailed dictionaries to look up a word

stop word removal	*“best”, “course”, “ever”, “taught”, “Text”, “Mining”*
↓	↓
stemming or/and lemmatization	*“best”, “course”, “ever”, “teach”, “text”, “mine”*

Let’s practice!

Tasks

- Word representations
- Data preprocessing

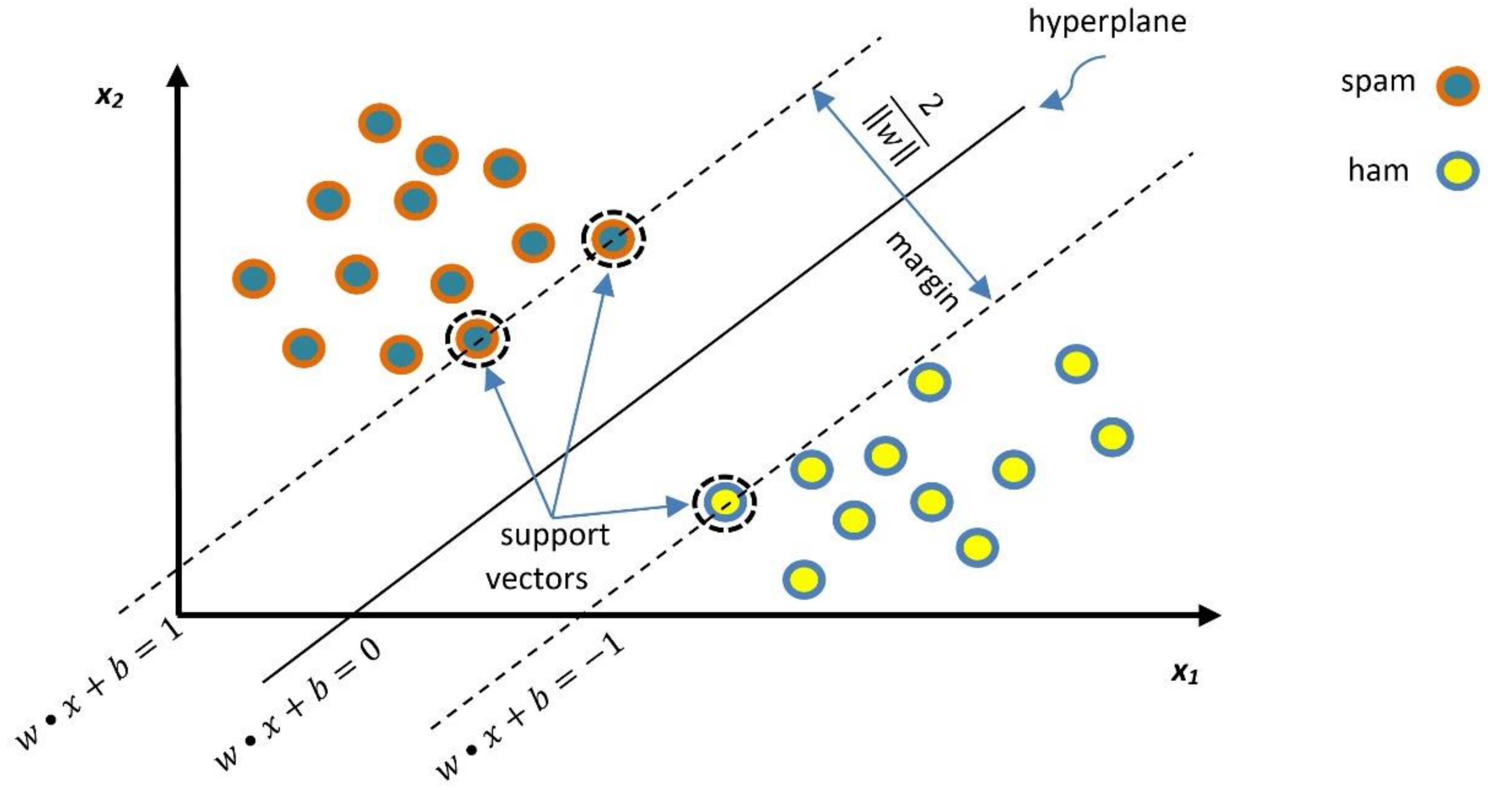
https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-02/spam-detection.ipynb

Machine Learning Techniques for Text

Section 4: Performing classification

Support Vector Machines

- ***Support Vector Machines*** is a supervised machine learning algorithm



Support Vector Machines

- The data below is separated with two different lines. Which separation is better?

Support Vector Machines

- The data below is separated with two different lines. Which separation is better?

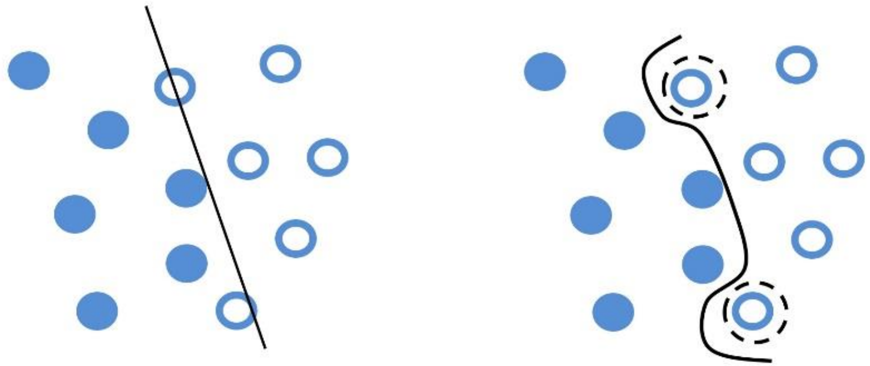
- The line in the left plot has a higher classification error
- The margin in the second is small and therefore the model lacks in generalization

Hyperparameter ***C***

- The SVM algorithm permits the adjustment of the training accuracy versus generalization tradeoff using the hyperparameter ***C***
- A frequent point of confusion is that hyperparameters and model parameters are the same things, but this is not true
  - Hyperparameters are parameters whose values are used to control the learning process
  - On the other hand, model parameters update their value in very training iteration until we obtain a good classification model
- We can direct the SVM to create the most efficient model for each problem by adjusting the hyperparameter ***C***

Support Vector Machines

- Which one seems to work better this time?



- At first glance, the curved line in the plot on the right perfectly separates the data into two classes
- But getting too specific boundaries entails the risk of ***overfitting***
  - The model learns the training data perfectly but fails to classify a slightly different example correctly

Overfitting

- Most of us have grown in a certain cultural context, trained (overfit) to interpret social signals like body posture, facial expressions, voice tone, etc. in a certain way
- When socializing with people of diverse social cultures we might fail to interpret similar social signals correctly (generalization)

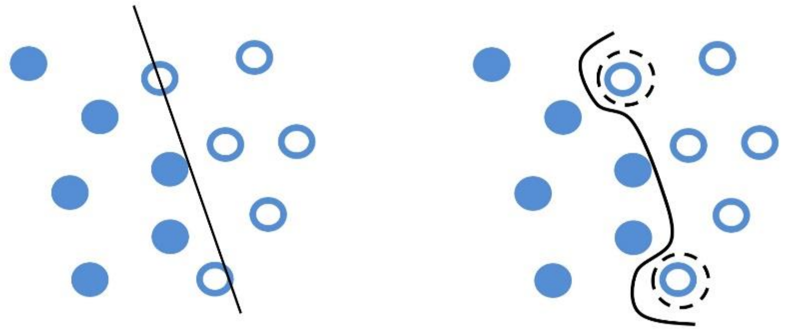
Overfitting

- Most of us have grown in a certain cultural context, trained (overfit) to interpret social signals like body posture, facial expressions, voice tone, etc. in a certain way
- When socializing with people of diverse social cultures we might fail to interpret similar social signals correctly (generalization)

There is always a tradeoff between ***accuracy*** during training and ***generalization*** during inference

Hyperparameter ***gamma***

- We can also prevent overfitting by adjusting the ***gamma*** hyperparameter
- Defines how far the influence of a single training example reaches
- The curvature of the decision boundary can also be affected by points that are pretty far from it.
- The takeaway here is that both ***C*** and ***gamma*** hyperparameters help us create more efficient models but identifying their best values demands experimentation



Data pipeline

- Getting the data
  - Use the ***SpamAssassin*** public mail corpus is a selection of mail messages, suitable for use in testing spam filtering systems
  - https://spamassassin.apache.org/old/publiccorpus/
- Creating the train and test sets
  - We choose a 75:25 split between the training and test sets, attributing the larger proposition to the training data
- Preprocessing the data
  - tokenization, stop word removal, and lemmatization
- Extracting the features
  - tf-idf vectorization

Let’s practice!

Tasks

- Word representations
- Data preprocessing
- Classification

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-02/spam-detection.ipynb

Machine Learning Techniques for Text

Section 5: Measuring classification performance

Accuracy

- ***Accuracy*** is the percentage of correctly classified examples by an algorithm divided by the total number of examples, defined as:

$$Accuracy = \frac{Number\ of\ correctly\ classified\ examples}{Total\ number\ of\ examples}$$

- If we had to choose the best algorithm this should probably be the one with the highest accuracy
- The argument is that the algorithm with the highest number of correct classifications should be the right choice
- Although this is not far from the truth, it is not always the case

Accuracy

- A model classifies 1000 emails that we already know whether they spam or ham
- A possible result of classifying these emails is shown in the table below, known as ***confusion matrix***

	spam	ham
spam	True Positive (TP) Reality: spam Prediction: spam Total number: 15	False Positive (FP) Reality: ham Prediction: spam Total number: 20
ham	False Negative (FN) Reality: spam Prediction: ham Total number: 85	True Negative (TN) Reality: ham Prediction: ham Total number: 880

Accuracy

- Rewrite the f*ormula*

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{15 + 880}{15 + 880 + 20 + 85} = 0.895$$

	spam	**ham**
spam	True Positive (TP) Reality: spam Prediction: spam Total number: 15	False Positive (FP) Reality: ham Prediction: spam Total number: 20
ham	False Negative (FN) Reality: spam Prediction: ham Total number: 85	True Negative (TN) Reality: ham Prediction: ham Total number: 880

Accuracy

- Rewrite the formula

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{15 + 880}{15 + 880 + 20 + 85} = 0.895$$

	spam	ham
spam	True Positive (TP) Reality: spam Prediction: spam Total number: 15	False Positive (FP) Reality: ham Prediction: spam Total number: 20
ham	False Negative (FN) Reality: spam Prediction: ham Total number: 85	True Negative (TN) Reality: ham Prediction: ham Total number: 880

- Out of the 1000 spam emails (TP + FN) only 15 were correctly identified and the other 85 were considered as ham emails

Accuracy

- To assess the performance of a model correctly, we need to make this analysis and consider the ***type of errors*** that are most important within the task

- Is it better to have a ***strict*** model that can block a legitimate email for the sake of fewer spam ones (increased FPs)?
- Or is it preferable to have a ***lenient*** model that doesn't block most ham emails but allows more undetected spam in your mailbox (increased FNs)?

	spam	**ham**
spam	True Positive (TP) Reality: spam Prediction: spam Total number: 15	False Positive (FP) Reality: ham Prediction: spam Total number: 20
ham	False Negative (FN) Reality: spam Prediction: ham Total number: 85	True Negative (TN) Reality: ham Prediction: ham Total number: 880

Accuracy

- Similar questions arise in all ML problems and generally in many real-world situations
- For example, wrong affirmative decisions (FPs) in a fire alarm system are preferable to wrong negative ones (FNs)
  - In the first case, we get a false alert of a fire that didn't occur
- Conversely, declaring innocent a guilty prisoner implies higher FNs, which is preferable to finding guilty an innocent one (higher FPs)
- Accuracy is a good metric when the test data is balanced and the classes are equally important

Precision and Recall

- ***Precision*** us the proportion of positive identifications that are, in reality, correct, given by:

$$Precision = \frac{TP}{TP + FP} = \frac{15}{15 + 20} = 0.43$$

- ***Recall*** tells us the proportion of the actual positives that are identified correctly, given by:

$$Recall = \frac{TP}{TP + FN} = \frac{15}{15 + 85} = 0.15$$

- Improving precision often deteriorates recall and vice versa

F-score

- We can combine precision and recall (harmonic mean) in one more reliable ***F-score*** metric

$$\text{F-score} = 2 * \frac{precision * recall}{precision + recall} = 2 * \frac{0.43 * 0.15}{0.43 + 0.15} = 0.22$$

- When precision and recall reach their perfect score (equal to 1), the F-score becomes 1
- Based on this metric we can evaluate more reliably different ML models

ROC and AUC

- When the classifier returns some kind of confidence score for each prediction, we can use another technique for evaluating performance called the ***Receiver Operator Characteristic*** (ROC) curve
- A ROC curve is a graphical plot that shows the model's performance at all classification thresholds
- It utilizes two rates, namely the ***True Positive Rate*** (TPR), the same as recall, and the ***False Positive Rate*** (FPR)

ROC and AUC

- ***True Positive Rate (TPR)***

$$TPR = \frac{TP}{TP + FN}$$

- ***False Positive Rate (FPR)***

$$FPR = \frac{FP}{FP + TN}$$

- The benefit of ROC curves is that they help us visually identify the trade-offs between the TPR and FPR
- In this way, we can find which classification threshold better suits the problem under study

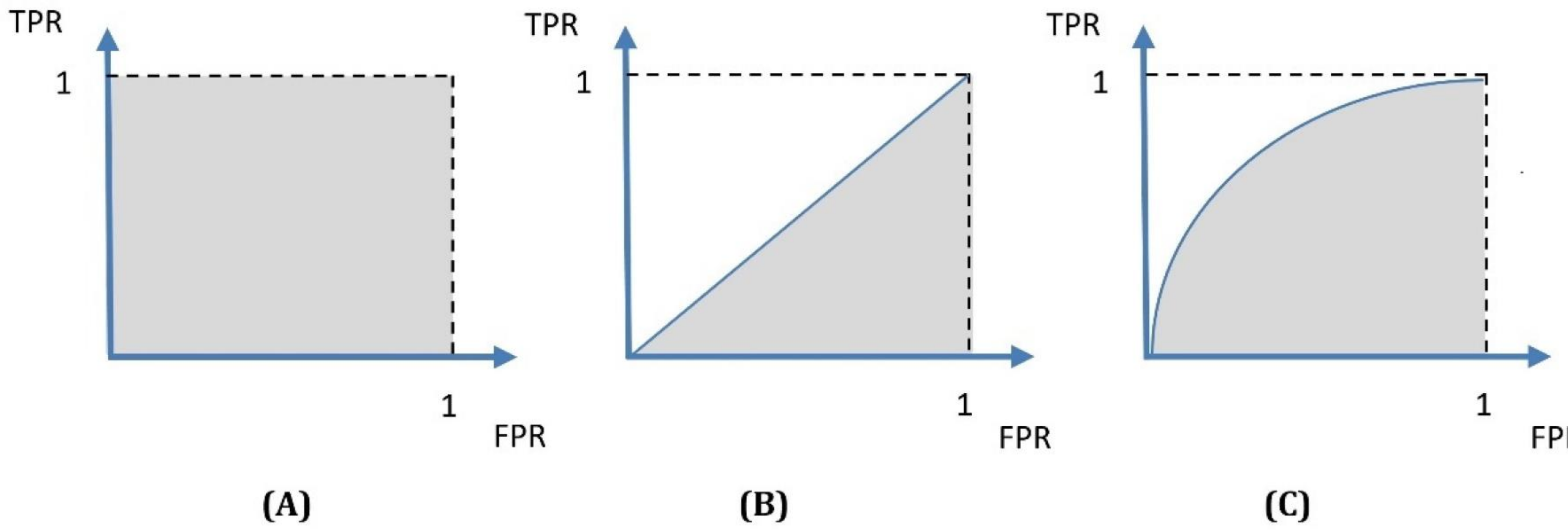
ROC and AUC

- A simplified example with 10 emails
- 7 thresholds are used
- For each prediction a probability score is generated by the model

Reality ● ham ○ spam	Pred. prob.	Threshold 0.0	0.55	0.65	0.75	0.85	0.95	1.0
●	0.1	○	●	●	●	●	●	●
●	0.2	○	●	●	●	●	●	●
●	0.3	○	●	●	●	●	●	●
●	0.4	○	●	●	●	●	●	●
●	0.5	○	●	●	●	●	●	●
○	0.6	○	○	●	●	●	●	●
●	0.7	○	○	○	●	●	●	●
●	0.8	○	○	○	○	●	●	●
○	0.9	○	○	○	○	○	●	●
○	0.99	○	○	○	○	○	○	●
		3 7	3 2	2 2	2 1	2 0	1 0	0 0
		0 0	0 5	1 5	1 6	1 7	2 7	3 7
	TPR	1	1	0.67	0.67	0.67	0.33	0
	FPR	1	0.29	0.29	0.14	0	0	0

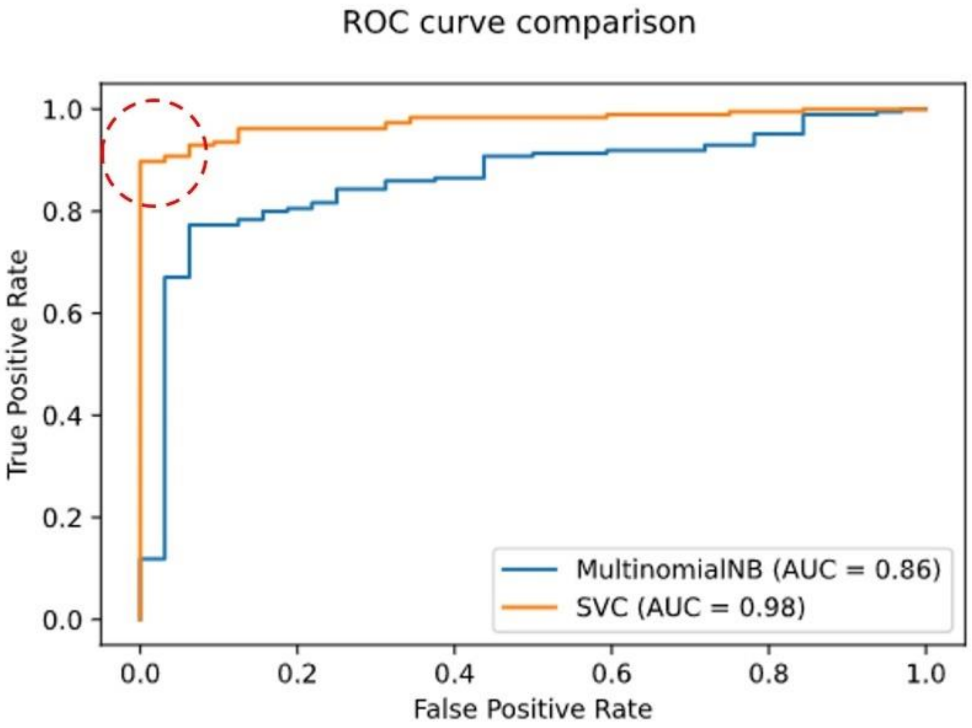
ROC and AUC

- The grayed area in these plots, called the ***Area Under the ROC Curve*** (AUC), is related to the quality of our model; the higher its surface, the better it is



AUC for two models

- The model based the support vector machines provides better results



Precision-recall curves

- ROC curves can sometimes perform too optimistically with imbalanced datasets
- For example, using the TN factor during the FPR calculation can skew the results
- The solution, in this case, is to generate another visualization called the ***Precision-Recall curve***
- Fortunately, this factor is not part of the precision or recall formulas
- We must scrutinize both ROC and precision-recall curves to understand the models' performance and the differences between the classifiers

Let's practice!

Tasks

- Word representations
- Data preprocessing
- Classification
- Performance

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-02/spam-detection.ipynb

Machine Learning Techniques for Text

Section 6: Typical pipeline

input “*The best course ever taught about Text Mining*” input → "The best course ever taught about Text Mining"

tokenization → *"The", "best", "course", "ever", "taught", "about", "Text", "Mining"*

input	*“The best course ever taught about Text Mining”*
↓	↓
tokenization	*“The”, “best”, “course”, “ever”, “taught”, “about”, “Text”, “Mining”*
↓	↓
stop word removal	*“best”, “course”, “ever”, “taught”, “Text”, “Mining”*

input	*“The best course ever taught about Text Mining”*
↓	↓
tokenization	*“The”, “best”, “course”, “ever”, “taught”, “about”, “Text”, “Mining”*
↓	↓
stop word removal	*“best”, “course”, “ever”, “taught”, “Text”, “Mining”*
↓	↓
stemming or/and lemmatization	*“best”, “course”, “ever”, “teach”, “text”, “mine”*

input → *“The best course ever taught about Text Mining”*

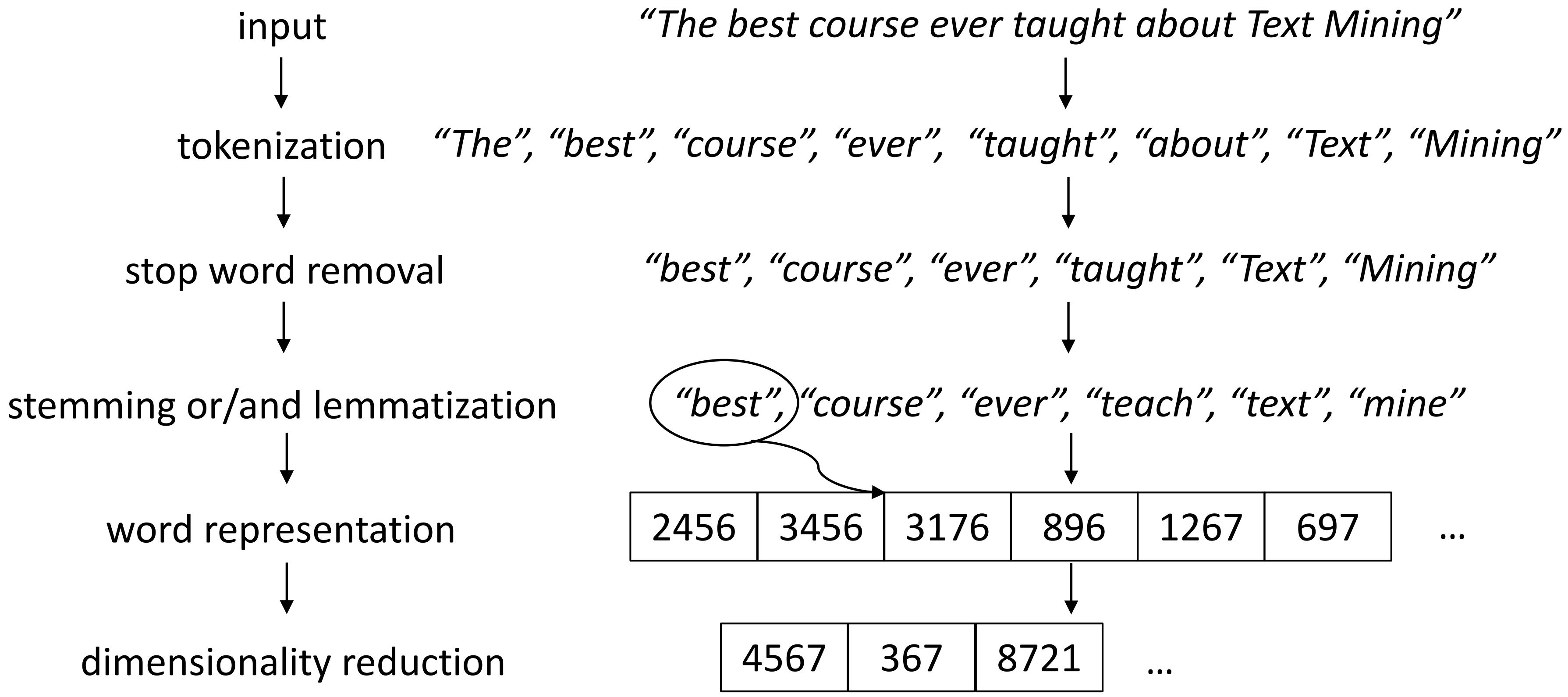
tokenization → *“The”, “best”, “course”, “ever”, “taught”, “about”, “Text”, “Mining”*

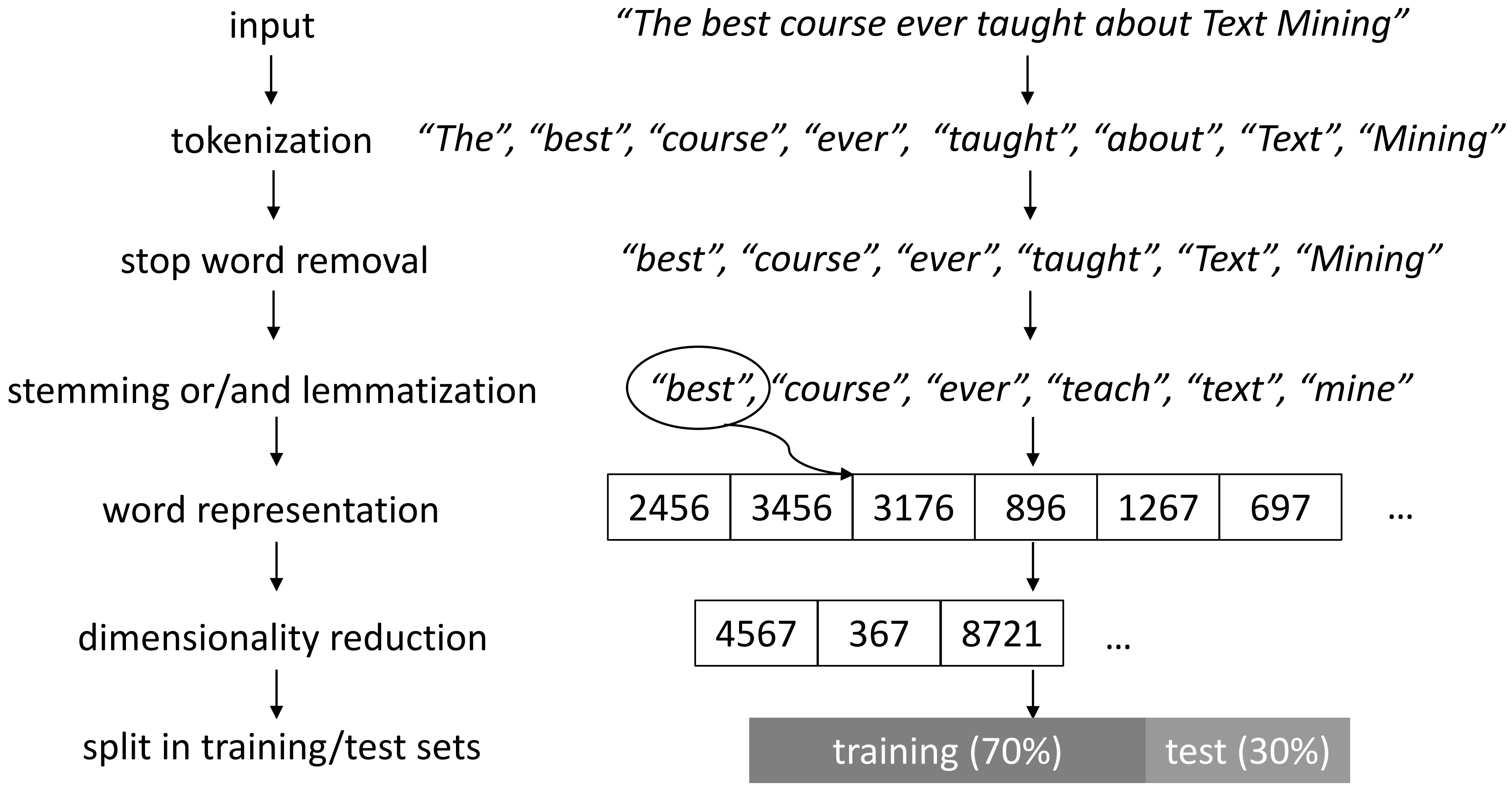
stop word removal → *“best”, “course”, “ever”, “taught”, “Text”, “Mining”*

stemming or/and lemmatization → *“best”, “course”, “ever”, “teach”, “text”, “mine”*

word representation →

2456	3456	3176	896	1267	697	…


input
"The best course ever taught about Text Mining"
tokenization
"The", "best", "course", "ever", "taught", "about", "Text", "Mining"
stop word removal
"best", "course", "ever", "taught", "Text", "Mining"
stemming or/and lemmatization
"best", "course", "ever", "teach", "text", "mine"
word representation
2456
3456
3176
896
1267
697
…
dimensionality reduction
4567
367
8721
…


input
“The best course ever taught about Text Mining”
tokenization
“The”, “best”, “course”, “ever”, “taught”, “about”, “Text”, “Mining”
stop word removal
“best”, “course”, “ever”, “taught”, “Text”, “Mining”
stemming or/and lemmatization
“best”, “course”, “ever”, “teach”, “text”, “mine”
word representation
2456 | 3456 | 3176 | 896 | 1267 | 697 …
dimensionality reduction
4567 | 367 | 8721 …
split in training/test sets
training (70%)
test (30%)

input	*“The best course ever taught about Text Mining”*
↓	↓
tokenization	*“The”, “best”, “course”, “ever”, “taught”, “about”, “Text”, “Mining”*
↓	↓
stop word removal	*“best”, “course”, “ever”, “taught”, “Text”, “Mining”*
↓	↓
stemming or/and lemmatization	*“best”, “course”, “ever”, “teach”, “text”, “mine”*
↓	↓
word representation	2456 | 3456 | 3176 | 896 | 1267 | 697 …
↓	↓
dimensionality reduction	4567 | 367 | 8721 …
↓	↓
split in training/test sets	training (70%) | test (30%)
↓	↓
classification	algorithm=KNN, k=9

input	*“The best course ever taught about Text Mining”*
tokenization	*“The”, “best”, “course”, “ever”, “taught”, “about”, “Text”, “Mining”*
stop word removal	*“best”, “course”, “ever”, “taught”, “Text”, “Mining”*
stemming or/and lemmatization	*“best”, “course”, “ever”, “teach”, “text”, “mine”*
word representation	2456 | 3456 | 3176 | 896 | 1267 | 697 …
dimensionality reduction	4567 | 367 | 8721 …
split in training/test sets	training (70%) | test (30%)
classification	algorithm=KNN, k=9
performance assessment	accuracy=90%

input	*“The best course ever taught about Text Mining”*
tokenization	*“The”, “best”, “course”, “ever”, “taught”, “about”, “Text”, “Mining”*
stop word removal	*“best”, “course”, “ever”, “taught”, “Text”, “Mining”*
stemming or/and lemmatization	*“best”, “course”, “ever”, “teach”, “text”, “mine”*
word representation	2456 | 3456 | 3176 | 896 | 1267 | 697 …
dimensionality reduction	4567 | 367 | 8721 …
split in training/test sets	training (70%) | test (30%)
classification	algorithm=KNN, k=9
performance assessment	accuracy=90%
hyperparameter tuning	k=1-100

input

input

↓

tokenization

input

↓

tokenization

↓

stop word removal

input

↓

tokenization

↓

stop word removal

↓

stemming or/and lemmatization

Porter Stemmer, Wordnet

input

↓

tokenization

↓

stop word removal

↓

stemming or/and lemmatization — Porter Stemmer, Wordnet

↓

word representation — Label, One-hot, Token count, tf-idf, Word Embedding

input

↓

tokenization

↓

stop word removal

↓

stemming or/and lemmatization — Porter Stemmer, Wordnet

↓

word representation — Label, One-hot, Token count, tf-idf, Word Embedding

↓

dimensionality reduction — PCA, LDA, SVD, t-SNE

input

↓

tokenization

↓

stop word removal

↓

stemming or/and lemmatization — Porter Stemmer, Wordnet

↓

word representation — Label, One-hot, Token count, tf-idf, Word Embedding

↓

dimensionality reduction — PCA, LDA, SVD, t-SNE

↓

split in training/test sets

input

↓

tokenization

↓

stop word removal

↓

stemming or/and lemmatization — Porter Stemmer, Wordnet

↓

word representation — Label, One-hot, Token count, tf-idf, Word Embedding

↓

dimensionality reduction — PCA, LDA, SVD, t-SNE

↓

split in training/test sets

↓

classification — SVM, Naïve Bayes, K-Nearest Neighbor, Random Forest, Logistic Regression, Neural Networks

input	
tokenization	
stop word removal	
stemming or/and lemmatization	Porter Stemmer, Wordnet
word representation	Label, One-hot, Token count, tf-idf, Word Embedding
dimensionality reduction	PCA, LDA, SVD, t-SNE
split in training/test sets	
classification	SVM, Naïve Bayes, K-Nearest Neighbor, Random Forest, Logistic Regression, Neural Networks
performance assessment	Accuracy, Precision, Recall, F-score, Confusion matrix

input	
↓	
tokenization	
↓	
stop word removal	
↓	
stemming or/and lemmatization	Porter Stemmer, Wordnet
↓	
word representation	Label, One-hot, Token count, tf-idf, Word Embedding
↓	
dimensionality reduction	PCA, LDA, SVD, t-SNE
↓	
split in training/test sets	
↓	
classification	SVM, Naïve Bayes, K-Nearest Neighbor, Random Forest, Logistic Regression, Neural Networks
↓	
performance assessment	Accuracy, Precision, Recall, F-score, Confusion matrix
↓	
hyperparameter tuning	Grid search

Key takeaways

Visualizations

- Word clouds

Text representations

- Label encoding
- One-hot encoding
- Token count encoding
- Tf-idf encoding

ML algorithms & models

- Support Vector Machines
- Naïve Bayes

Text preprocessing

- Tokenization
- Stop words removal
- Stemming
- Lemmatization
- Regular expressions

ML concepts

- Supervised learning
- Creating train and test sets
- Feature engineering
- Overfitting

Performance metrics

- Accuracy
- Precision
- Recall
- F-score
- ROC and AUC
- True Positive Rate

Machine Learning Techniques for Text

Questions?