FOLT Lecture 4,Text Classification

Intro

● Text classification is the task of choosing the correct **class label** for a given **textual instance**
○ Text categorization ○ Spam detection ○ Sentiment classification
● Classification variants ○ **Binary classification** (2 classes) vs. **multi-class classification** (>2 classes)
- Binary classification: classify the input data instance into one of two of possible classes or categories (Example: spam detection (Yes/No)
)
- Multi-class classification: classify the input data instance into one of several possible categories
(Example: sentiment classification (Positive/Negative/Neutral))
○ Single-label classification vs. multi-label classification
- Single-label classification: assign a single label to each input instance (Example: spam detection (Yes/No), sentiment classification
(Positive/Negative/Neutral))
- Multi-label classification: assign one or more labels to each input instance (Example: text categorization
(Politics/Sports/Arts/Tech/Business/Entertainment/…))

Rule-based approaches

- Handcrafted linguistic rules for spam detection

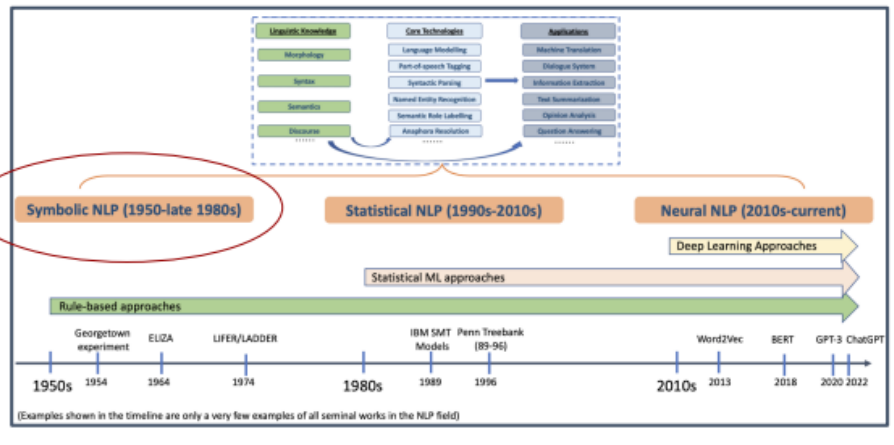
UBIQUITOUS KNOWLEDGE PROCESSING

From: MSTeam-Outlook Message Center <no-reply@office365protectionservices.co.uk>
Sent: 19 September 2018 11:44
To: Bob Smith <Bob.Smith@Company.com>
Subject: Account Verification

Fake domain

```
>>> def is_spam_mail(mail):
...       if is_fake_domain(mail):
...             return True
...       elif is_a_threat(mail):
...             return True
...       ...
```

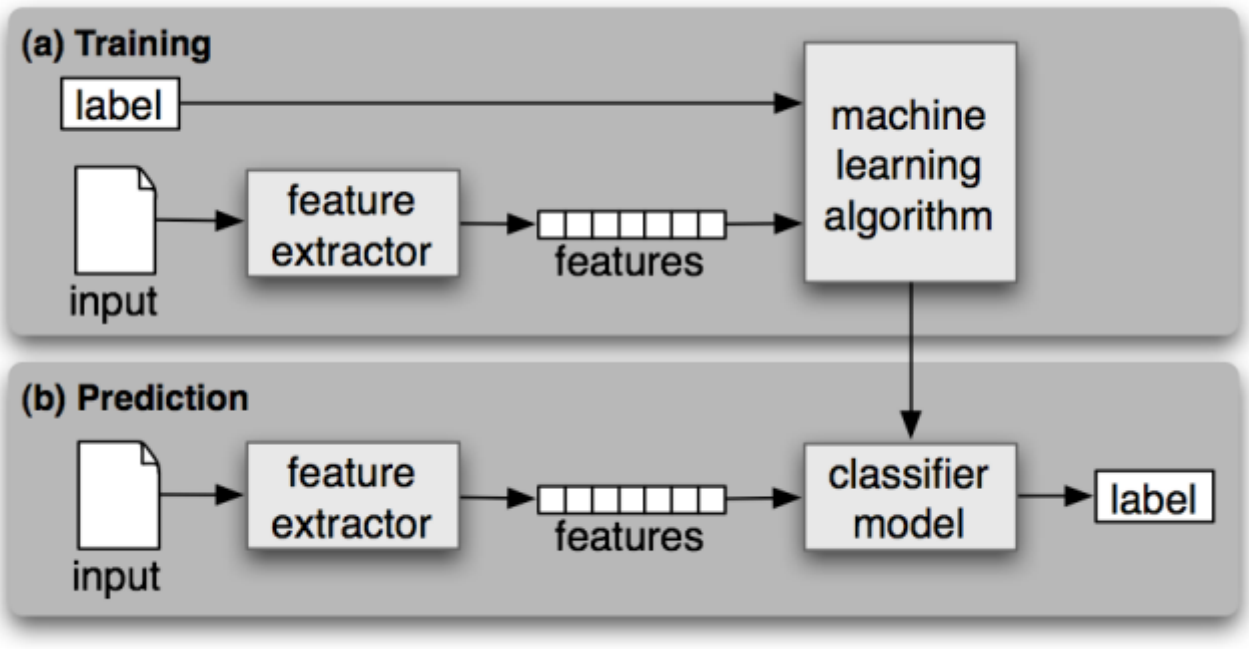
- Advantages:
  - Accurate
  - Quick
- Disadvantages:
  - Requires knowledge and experience to apply heuristics effectively
  - Can be time-consuming
  - Cannot generalize to unseen cases



(Lecture 2)

PART1: Supervised Machine Learning Process for Text Classification

Goal: build a supervised classifier to predict labels for the given input.



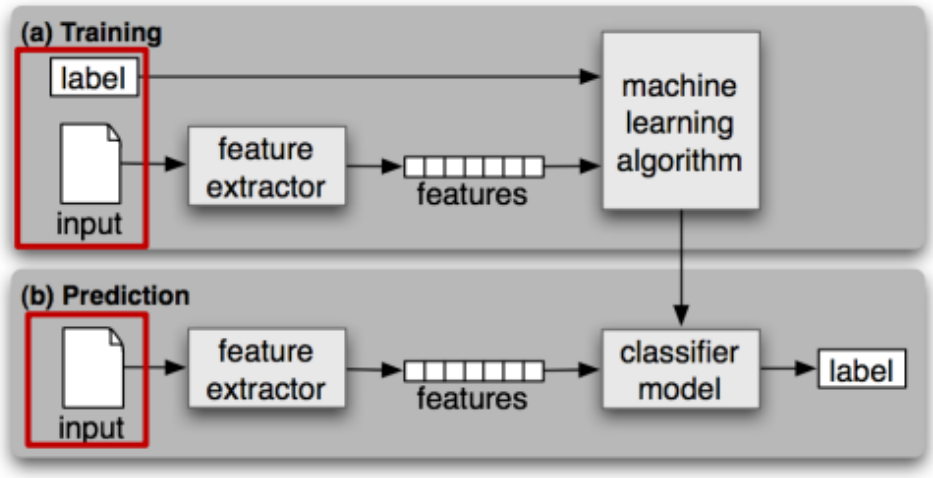
Data Collection

Data Pre-processing

Model Training and Prediction

Model Evaluation

Deployment, Monitoring and Maintenance



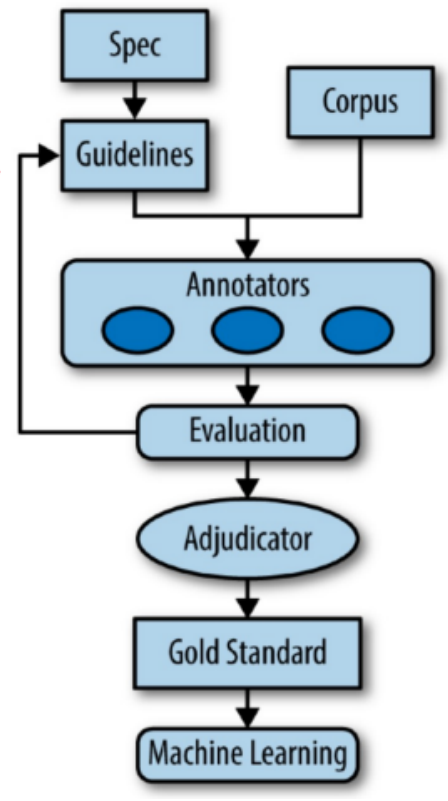
1) Data Collection

Data Collection

Define the problem (e.g., Spam detection)

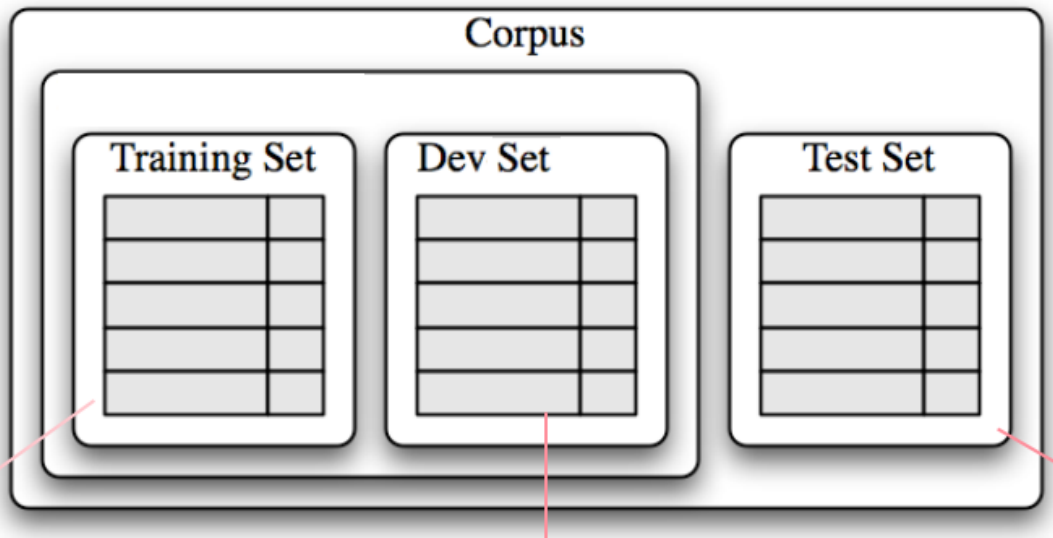
Obtain labelled dataset

- Data annotation (Lecture 2)
  - Automatic annotation (model prediction)
  - Human annotation
    - Natural/existing annotation
    - New/human annotation
      - ✓ IAA: Inter-annotator Agreement



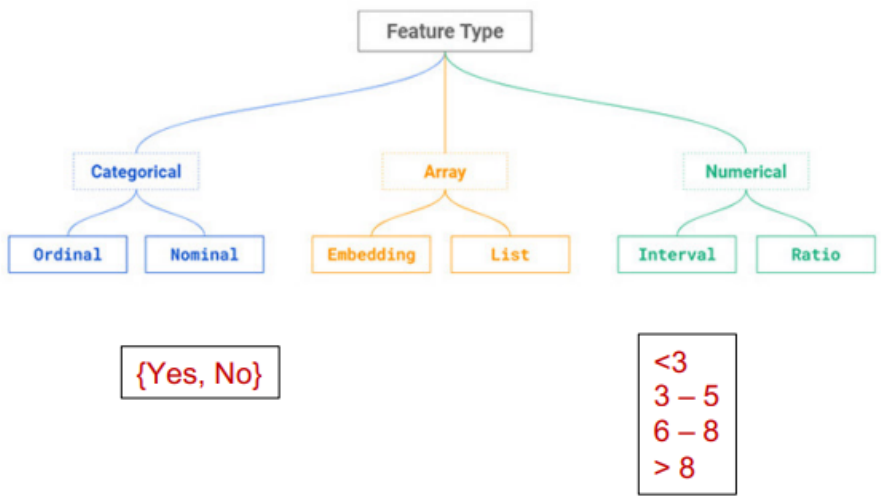
2) Data Pre-processing

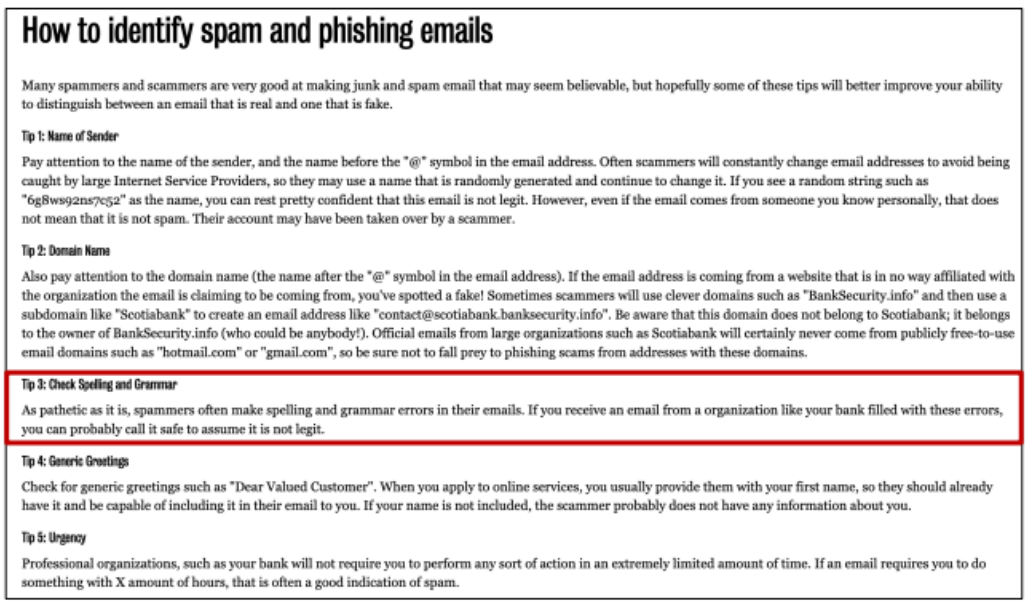
a)Split the Data: Train/(Dev)elopment/Test Sets



b) Extract features

o Which **features** are relevant to identify spam emails?
o How to **extract** and **encode** these features?





How to identify spam and phishing emails

Many spammers and scammers are very good at making junk and spam email that may seem believable, but hopefully some of these tips will better improve your ability to distinguish between an email that is real and one that is fake.

Tip 1: Name of Sender

Pay attention to the name of the sender, and the name before the "@" symbol in the email address. Often scammers will constantly change email addresses to avoid being caught by large Internet Service Providers, so they may use a name that is randomly generated and continue to change it. If you see a random string such as "6g8ws92ns7c52" as the name, you can rest pretty confident that this email is not legit. However, even if the email comes from someone you know personally, that does not mean that it is not spam. Their account may have been taken over by a scammer.

Tip 2: Domain Name

Also pay attention to the domain name (the name after the "@" symbol in the email address). If the email address is coming from a website that is in no way affiliated with the organization the email is claiming to be coming from, you've spotted a fake! Sometimes scammers will use clever domains such as "BankSecurity.info" and then use a subdomain like "Scotiabank" to create an email address like "contact@scotiabank.banksecurity.info". Be aware that this domain does not belong to Scotiabank; it belongs to the owner of BankSecurity.info (who could be anybody!). Official emails from large organizations such as Scotiabank will certainly never come from publicly free-to-use email domains such as "hotmail.com" or "gmail.com", so be sure not to fall prey to phishing scams from addresses with these domains.

Tip 3: Check Spelling and Grammar

As pathetic as it is, spammers often make spelling and grammar errors in their emails. If you receive an email from a organization like your bank filled with these errors, you can probably call it safe to assume it is not legit.

Tip 4: Generic Greetings

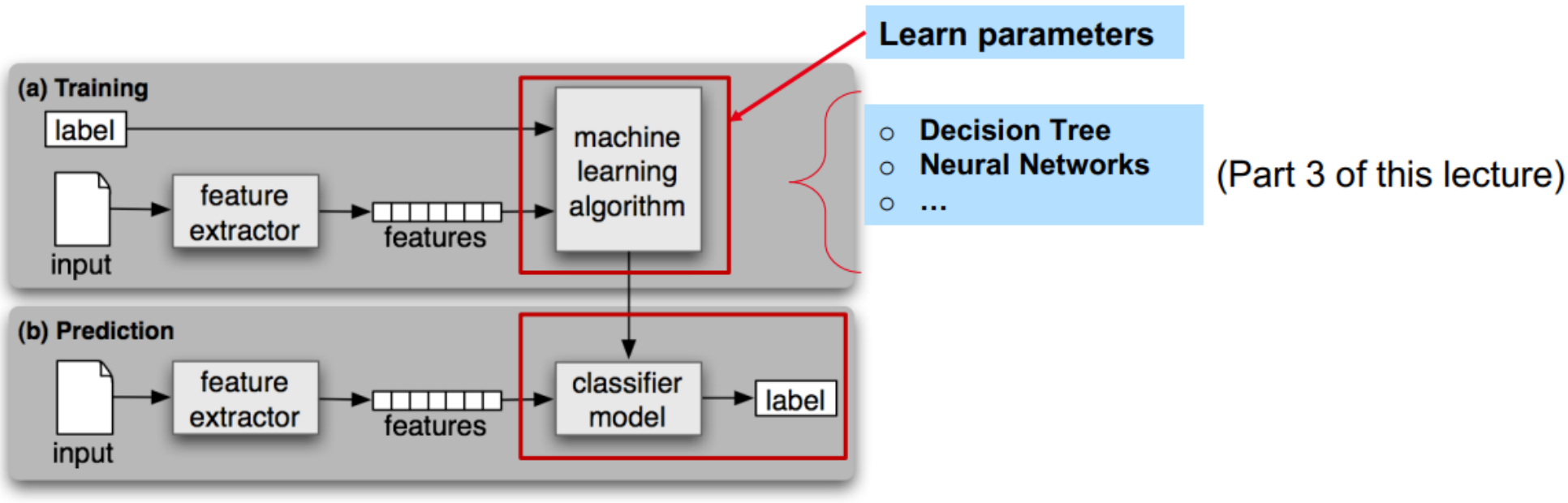
Check for generic greetings such as "Dear Valued Customer". When you apply to online services, you usually provide them with your first name, so they should already have it and be capable of including it in their email to you. If your name is not included, the scammer probably does not have any information about you.

Tip 5: Urgency

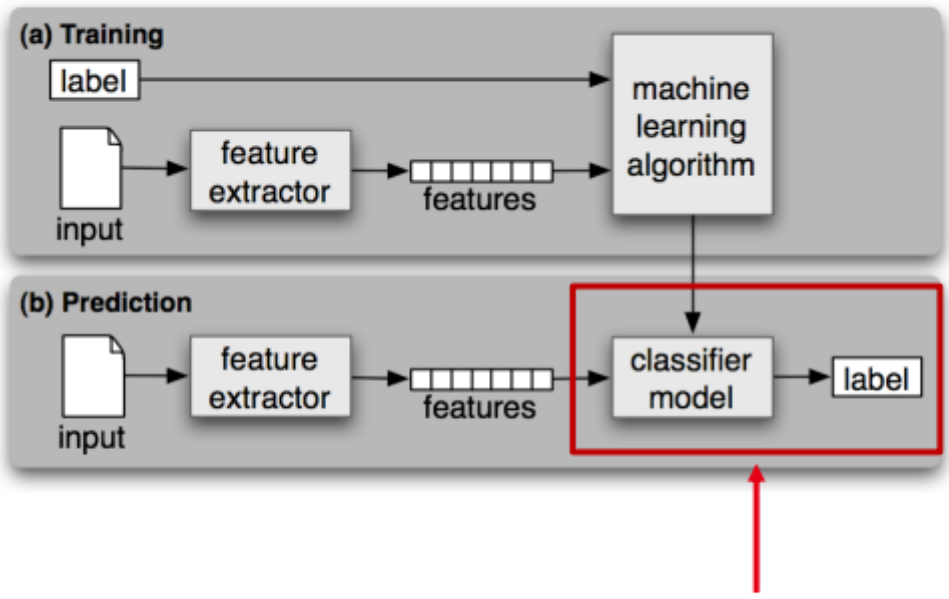
Professional organizations, such as your bank will not require you to perform any sort of action in an extremely limited amount of time. If an email requires you to do something with X amount of hours, that is often a good indication of spam.

3) Model Training and Prediction

o Choose a suitable machine learning algorithm for a specific task
o The choice of the model depends on the nature of the problem and the characteristics of the data

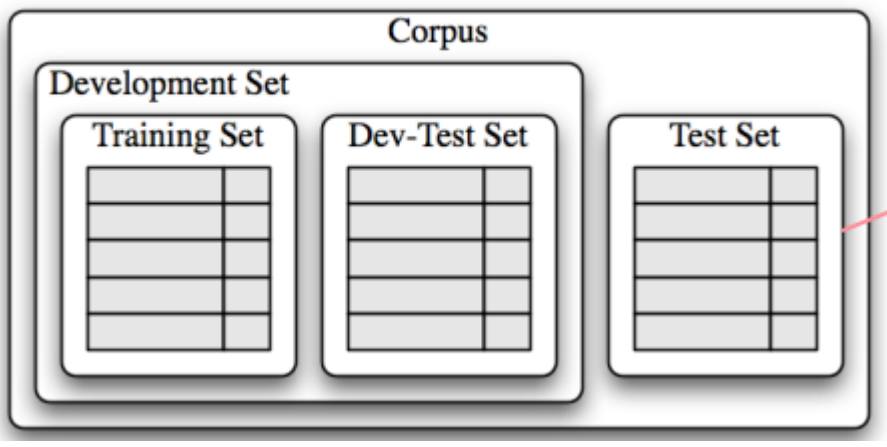


4)Model Evaluation



How good is the trained model?
How can we further improve the model's performance?

o Assess the performance of a learned model by estimating the error on a given reference dataset
o A reference dataset annotated by humans is called a gold standard dataset
o To evaluate, we compare the classifier's prediction with the (gold) labels of the (gold standard) dataset (This yields the performance/prediction error)

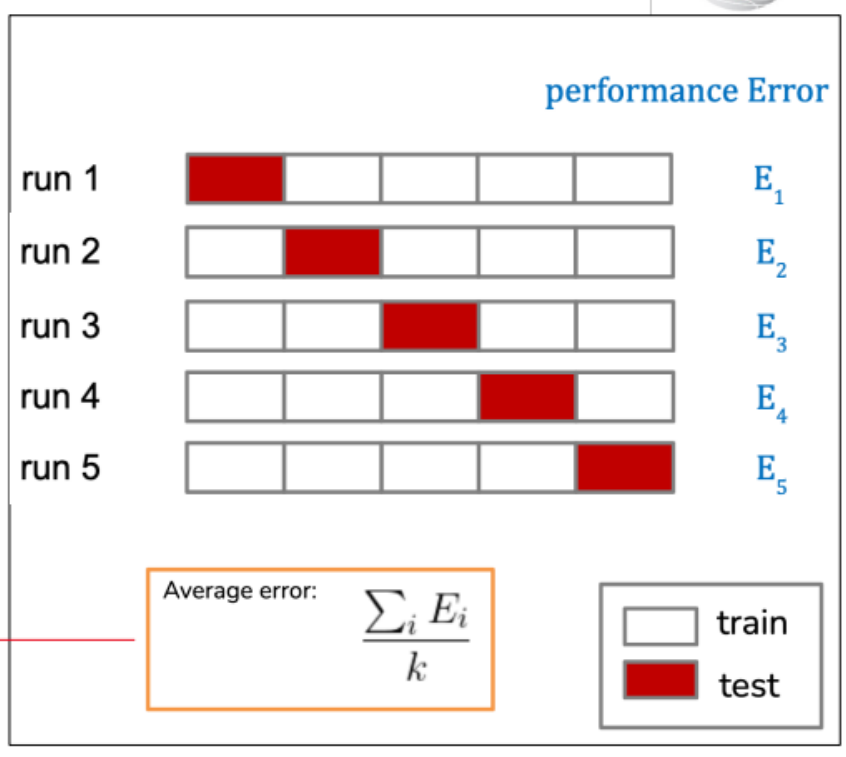


- Different from training
- Held-out data
- Assess generalization
- Don't optimize model on test set

Cross Validation

o Often, data is very scarce (= small)
o Good train / dev / test split is not always possible
o Test set is too small to get reliable results at all
→ Solution: Use "running" test sets to increase test data!
→ This is called cross validation

- *k*-fold cross validation
- Divide the data into *k* equal parts (folds)
- Train on *k-1* folds
- Test on the remaining fold
- Repeat *k* times such that each fold is used for evaluation once



- ✓ Each fold might be too small to give accurate evaluation scores on their own
- ✓ The **combined evaluation score** is based on a large amount of data → reliable

Two Goals of Evaluation

1. **Model selection:** choose the best model.
    - Dev set
    - **Feature tuning**: feature engineering
    - **(Hyper)parameter tuning**: Typically, our model will have tuning (hyper-)parameters and we wish to find the values of these (hyper)parameters that minimize the error.

2. **Model assessment:** estimating the trained model's **performance on new data**.
    - Test set
    - The prediction error on new data is called **generalization error**.

Evaluation Metric: Accuracy

o **Accuracy** measures the percentage of correctly classified instances in the test set.

$$\frac{\#\text{ correctly classified instances}}{\#\text{ all instances}}$$

- **Problem with accuracy**:
    - Imagine we have 1 million emails
        - 100 of them are spams
        - 999,900 are normal emails (non-spam)
    - We could build a majority-class classifier that just labels every email as "non-spam"
        - It would get 99.99% accuracy
        - But it doesn't return the results we are looking for (spam emails)
        - That's why we use ***precision*** and ***recall*** instead

Confusion Matrix: Binary Classification

Spam detection: Spam (positive) vs. Non-spam (negative)

		gold standard labels	
		gold positive	gold negative
system output labels	system positive	(tp) **true positive**	(fp) **false positive**
	system negative	(fn) **false negative**	(tn) **true negative**

- **Precision** measures percentage of positive instances the **system detected** that are **in fact positive/correct** (according to the human gold labels).

		gold standard labels		
		gold positive	gold negative	
system output labels	system positive	(tp) **true positive**	(fp) **false positive**	**precision** $= \frac{tp}{tp+fp}$
	system negative	(fn) **false negative**	(tn) **true negative**	

- **Recall** measures **percentage** of **positive** instances present in the input that **were correctly identified** by the system.

		gold standard labels	
		gold positive	gold negative
system output labels	system positive	(tp) **true positive**	(fp) **false positive**
	system negative	(fn) **false negative**	(tn) **true negative**

recall $= \frac{tp}{tp+fn}$

Why Precision and Recall

- Our majority-class classifier just labels nothing as “spam”
- Get accuracy = 99.99%
- But recall = 0
- Precision and recall, unlike accuracy, emphasize ***true positives***

A Combined Measure: F-measure

- F-measure combines precision (P) and recall (R) to give a single number

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- We almost **always** use **balanced F_1** (i.e., $\beta = 1$)

$$F_1 = \frac{2PR}{P + R}$$

Confusion Matrix for 3-class Classification

System Predictions	Gold labels		
	Urgent	Normal	Spam
Urgent	8	10	1
Normal	5	60	50
Spam	3	30	200

$$\text{Accuracy} = \frac{8+60+200}{8+5+3+10+60+30+1+50+200} = 0.73$$

Urgent class:

$$\text{Precision} = \frac{8}{8+10+1} = 0.42$$

$$\text{Recall} = \frac{8}{8+5+3} = 0.5$$

$$F_1 = \frac{2*0.42*0.5}{0.42+0.5} = 0.46$$

Normal class:

$$\text{Precision} = \frac{60}{5+60+50} = 0.52$$

$$\text{Recall} = \frac{60}{10+60+30} = 0.6$$

$$F_1 = \frac{2*0.52*0.6}{0.52+0.6} = 0.58$$

Spam class:

$$\text{Precision} = \frac{200}{3+30+200} = 0.86$$

$$\text{Recall} = \frac{200}{1+50+200} = 0.80$$

$$F_1 = \frac{2*0.85*0.8}{0.86+0.8} = 0.82$$

Combine Pre. and Rec. for Multiple Classes

- Macro-averaging:
  - compute the **performance for each class**, and then **average** over classes

- Micro-averaging:
  - collect **predictions for all classes** into one confusion matrix
  - compute precision and recall from that table

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

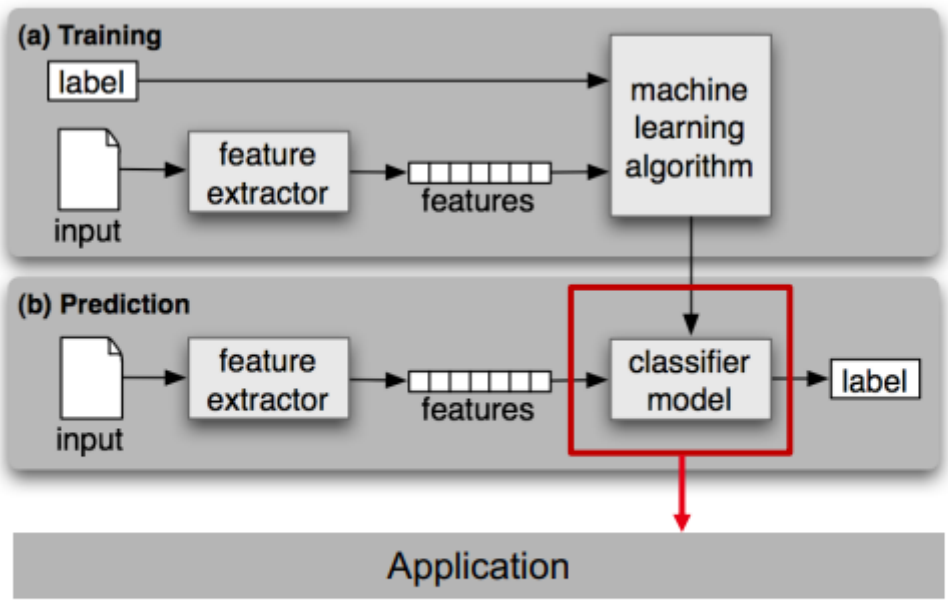
Error Analysis

- Manually examine a small subset of incorrectly-classified examples
  - ✓ E.g., 100 normal emails misclassified as “spam”

- Categorize them based on the common themes
  - ✓ Misspellings (e.g., w4tches, med1cine): 50
  - ✓ Grammatical errors: 25
  - ✓ Special phrases (e.g., *turn off, shut down, close*): *68*

- Improve the performance by
  - ✓ Collect more training data
  - ✓ Improve methods

5)Deployment, Monitoring and Maintenance



Does the model’s performance drop over time?

Does the model contain potential bias?

Harms in Text Classification

- **Sentiment Classifiers**
  - ✓ Assign lower sentiment and more *negative* emotion to sentences with African American names in them (Kiritchenko and Mohammad, 2018)
  - ✓ This perpetuates **negative stereotypes** that associate African Americans with negative emotions
- **Toxicity detection** is the task of detecting hate speech, abuse, harassment, or other kinds of toxic language.
  - ✓ Incorrectly classify non-toxic sentences as being *toxic*
  - ✓ These sentences simply mention identities like blind people, women, or gay people
  - ✓ This could lead to *censorship* of discussion about these groups

What Causes Harms in Text Classification?

o Problems in the training data à machine learning systems are known to amplify the biases in their training data

o Problems in the human labels

o Problems in the resources used (lexicons)

o Problems in model architecture (what model is trained/how is it optimized)

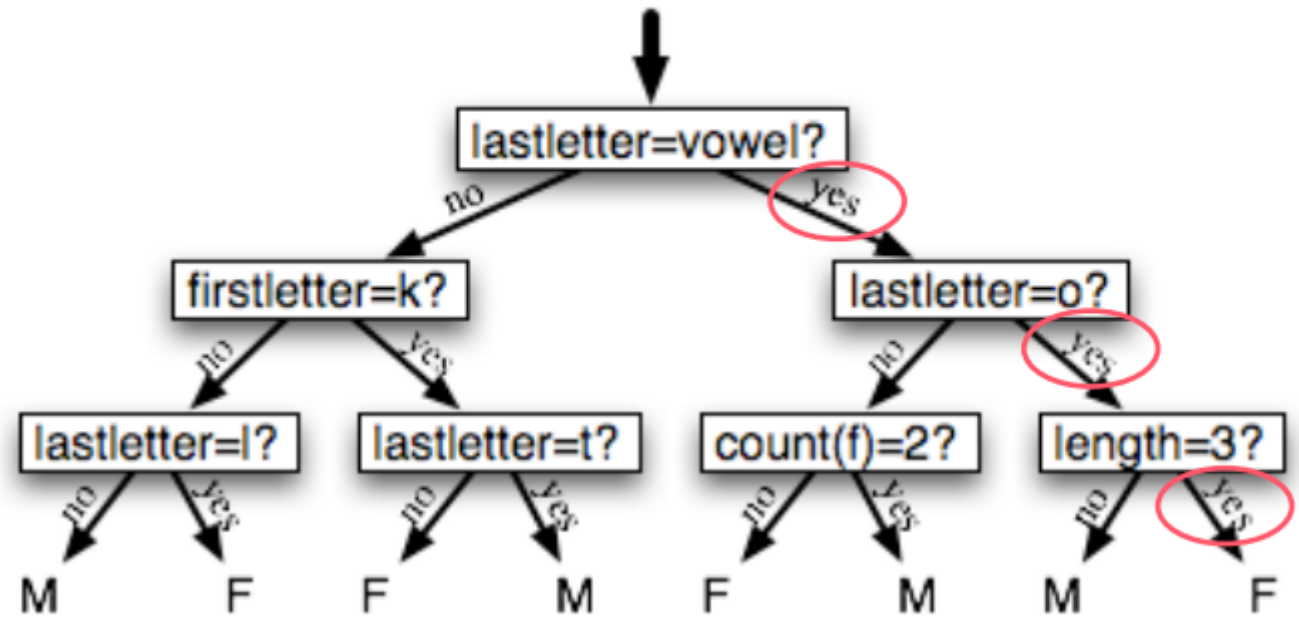
Mitigation of these harms is an **open research** area

→ Take into consideration when building text classification models

PART2: Some Machine Learning Models for Text Classification

1) Decision Tree

- Identifying gender of a given name



o Decision tree

makes a statement – **decision nodes**

makes a decision based on whether the statement is True or False –**leaf nodes**

o Classification: decision tree classifies things into categories

o Decision stump: one single feature for decision

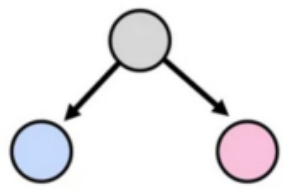
o Different feature types in the same tree

Yes/no: last letter=vowel

Numerical: length=3, count(f)=2

Build a Decision Tree

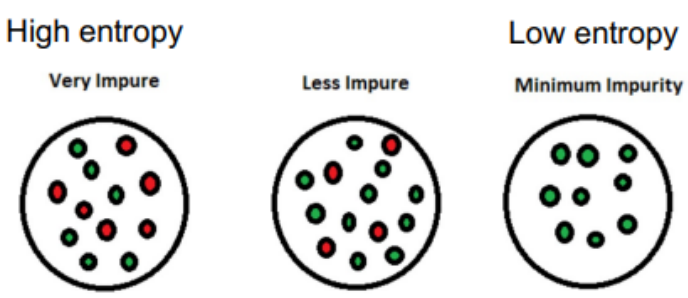
1. Select a feature and split data into a binary tree

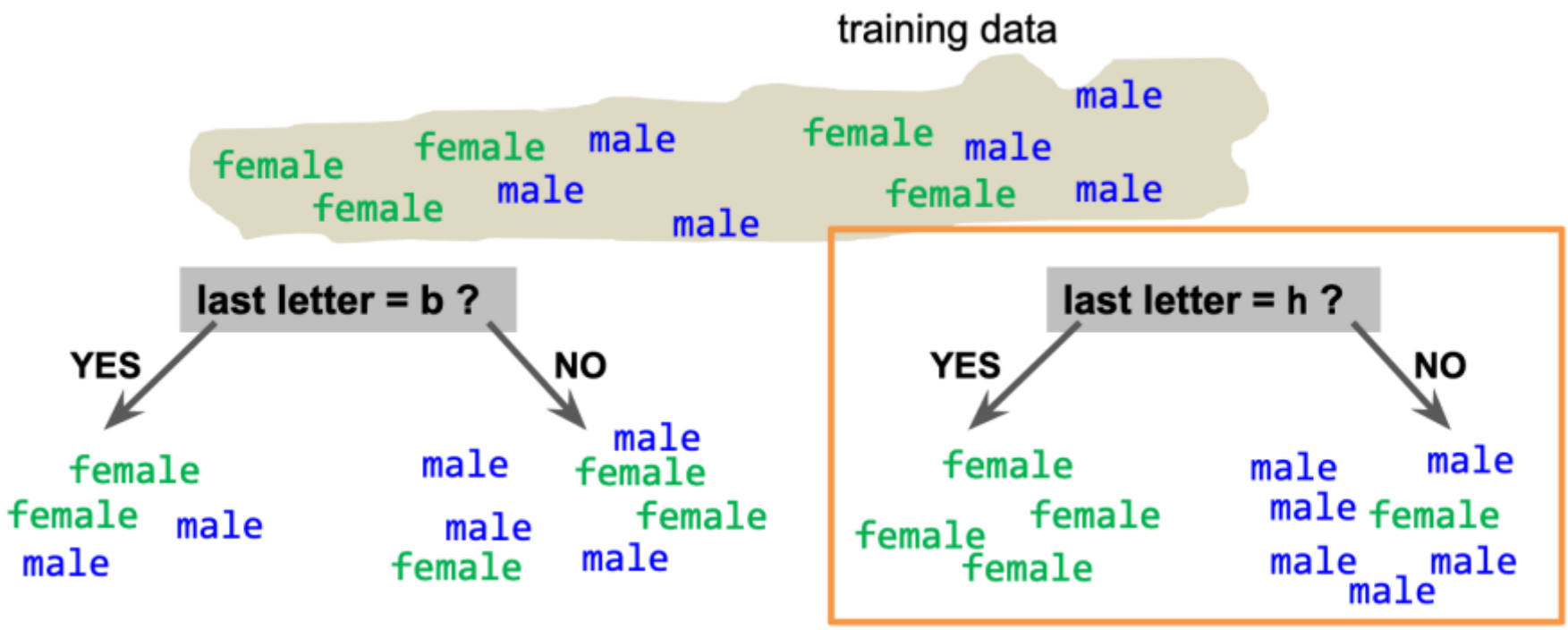


- ✓ When to split?
- ✓ Split that maximizes the **purity** (minimize **impurity**)
  - Entropy as impurity measure → learn more here
  - **Entropy** is an information theory metric that measures the impurity or uncertainty in a group of observations.
  - Consider a dataset with N classes. The entropy can be calculated using the formula below:

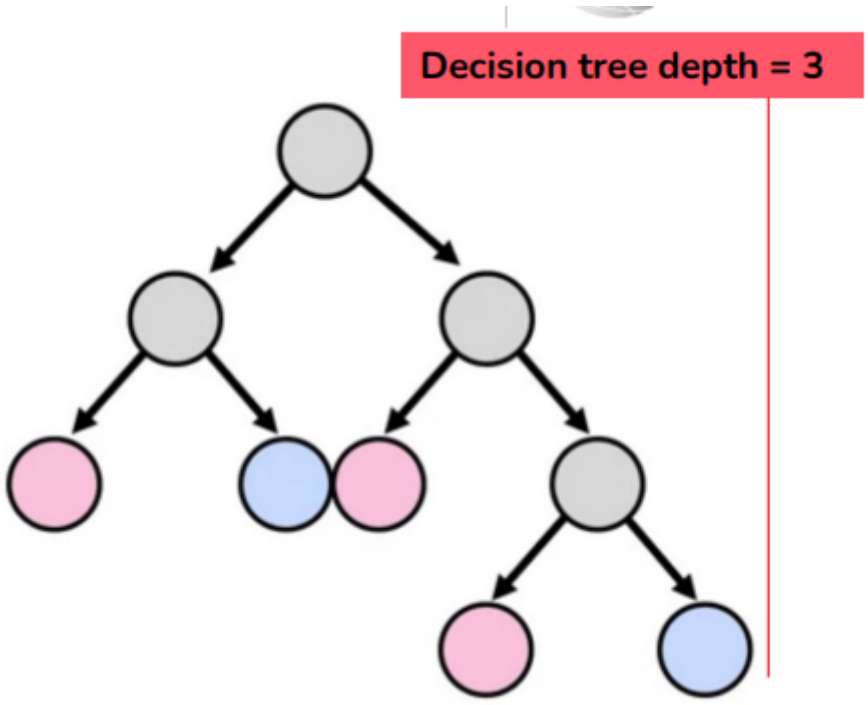
$$E = -\sum_{i=1}^{N} p_i log_2 p_i$$

P_i is the probability of randomly selecting an example in class *i*.





2. Continue splitting with available features
   - ✓ How long to keep splitting?
   - ✓ Until:
     - ✓ Leaf nodes are **pure** (only one class remains)
     - ✓ A **maximum depth** is reached
     - ✓ A performance metric is achieved



Quiz: Decision Tree

Take a decision tree learning to classify between spam and non-spam emails. There are 20 training examples at the root note, comprising 10 spam and 10 non-spam emails. If the algorithm can choose from among four features, resulting in four corresponding splits, which would it choose (i.e., which has highest purity)?

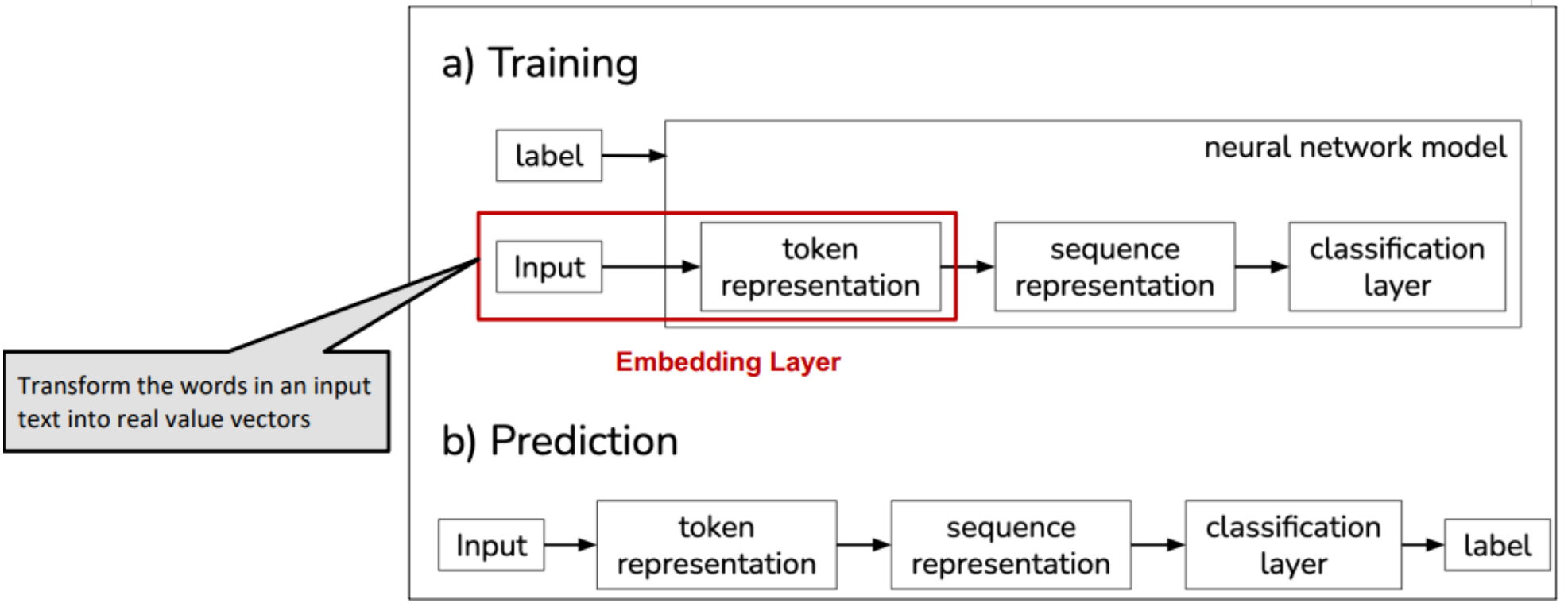
A. Left split: 10 of 10 emails are spam. Right split: 0 of 10 emails are spam.
B. Left split: 2 of 2 emails are spam. Right split: 8 of 18 emails are spam.
C. Left split: 5 of 10 emails are spam. Right split: 5 of 10 emails are spam.
D. Left split: 7 of 8 emails are spam. Right split: 3 of 12 emails are spam.

Decision Tree: Pro and Con

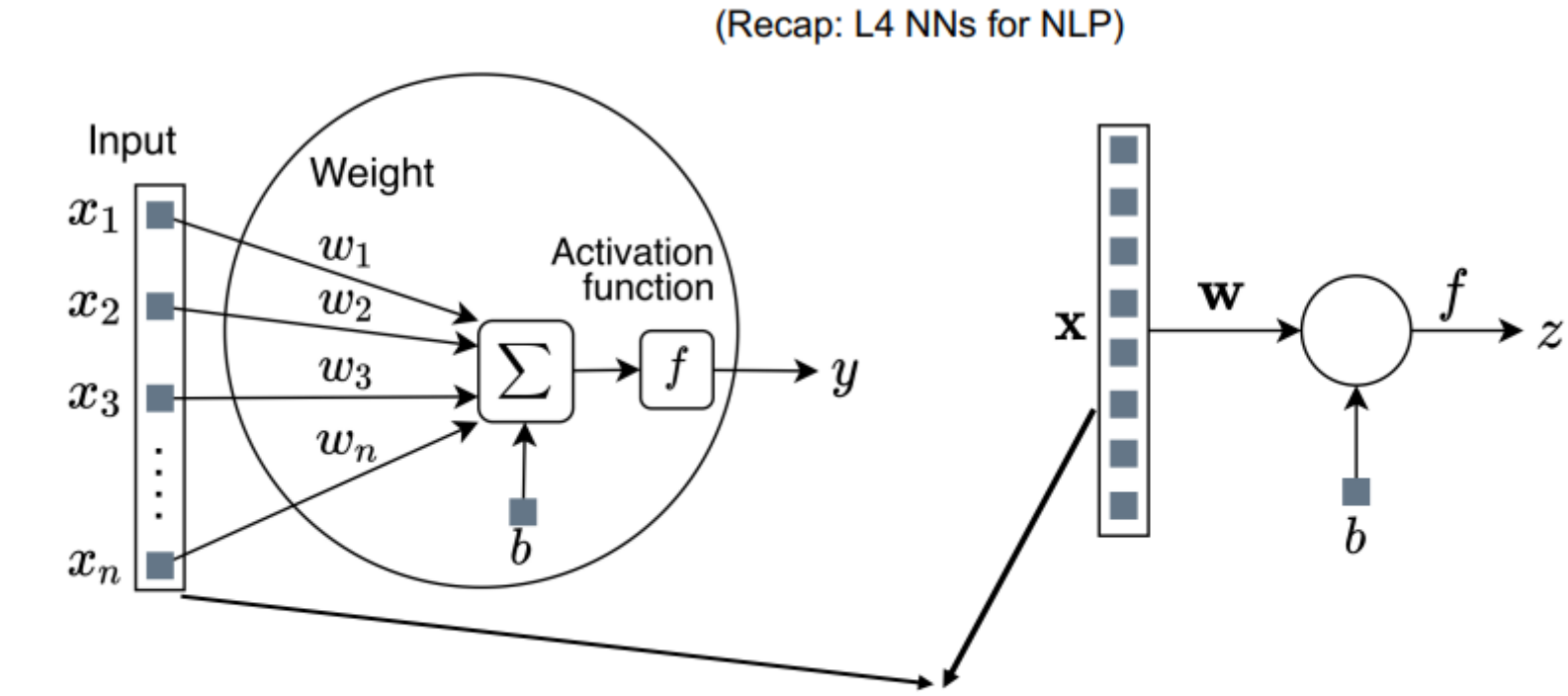
- **Advantages**
  - ✓ Simple to interpret
  - ✓ Especially useful to learn models for data, which can be hierarchically categorized

- **Problems**
  - ✓ Amount of training data in lower nodes is quite small (because it is split at every decision node)
    - ✓ we need to make sure that we do not **overfit the training data**
    - ✓ Force features to be checked in a specific order, even if features are independent of each other

2)Supervised Neural Text Classification



a)Input Representation



- Neural networks work well with **real-value** input

- We need to turn word sequences into real-value vectors
  → This is called **encoding** or **embedding** or **representation**

The Simplest Representation: One-hot Vectors

o **One-hot vector** is a **sparse** vector whose dimension is equal to the number of words in the vocabulary, all values are equal to **zero** except the word index that is equal to **one**, which **indicates the occurrence** of the word.

$$\boldsymbol{x}_{cat} = [0,0,\ldots,0,1,0,\ldots,0,0,0] \in \mathrm{R}^{|V|}$$

$$\boldsymbol{x}_{dog} = [0,0,\ldots,0,0,0,\ldots,0,1,0] \in \mathrm{R}^{|V|}$$

- Formally
  - $V = \{w_1, w_2, \ldots, w_{|V|}\}$: vocabulary of a language
  - $|V|$: size of the vocabulary

w_1: first word in the vocabulary; w_i: word i^{th} in the vocabulary

Example

- ✓ $\boldsymbol{x}_1 = [1, 0, 0, 0, \ldots, 0, 0]$: vector of the *first* word w_1 in the vocabulary
- ✓ $\boldsymbol{x}_2 = [0, 1, 0, 0, \ldots, 0, 0]$: vector of the *second* word w_2 in the vocabulary
- ✓ $\boldsymbol{x}_{|V|} = [0, 0, 0, 0, \ldots, 0, 1]$: vector of the *last* word $w_{|V|}$ in the vocabulary

- Vocabulary of a language $V = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$
- Size of the vocabulary $|V| = 8$
- Documents $text_1 = \{w_2, w_3, w_1, w_5\}$, $text2 = \{w_2, w_2, w_4, w_7, w_8\}$
- Vector representation of *text1 and text2*

$$\boldsymbol{x}_{text1} = [1, 1, 1, 0, 1, 0, 0, 0]$$

$$\boldsymbol{x}_{text2} = [0, 1, 0, 1, 0, 0, 1, 1]$$

- Count-based vectors: instead of 1s, replaced by counts of the words

$$\boldsymbol{x}_{text1} = [1, 1, 1, 0, 1, 0, 0, 0]$$
$$\boldsymbol{x}_{text2} = [0, 2, 0, 1, 0, 0, 1, 1]$$

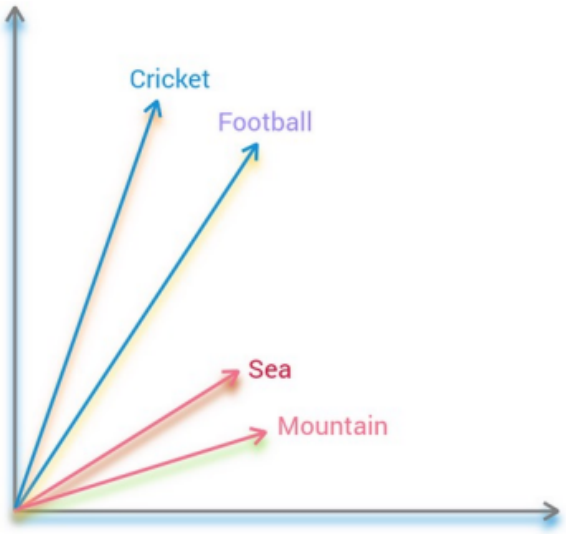
Problems With One-hot Vectors

- One-hot vectors are very **sparse**
  - ✓ Very high dimension → huge weight matrix

- Semantic relations between words are not encoded [Lecture 2]
  - ✓ E.g., dog and cat are pets, orange and apple are fruits

- **Solution**: represent words using *low dimensional* **dense vectors** / **word representations** → work well for neural networks
  - ✓ Instead of using $|V|$ dimensional vectors
  - ✓ Use **real-value dense vectors** with **smaller dimensions**, such as 50, 100, 300, 1024 < $|V|$
    - ✓ The Oxford English Dictionary contains over 300k entries
    - ✓ Adults who are native English speakers tend to have a vocabulary of 15k to 30k words
  - ✓ But getting larger, e.g., GPT-3 ~ > 10k dimensions (12,288)

- Word representations | dense word vectors | word embeddings | word vectors
  - Each word is a **dense** vector of size *d*
  - Dense: values are real-valued numbers, can be negative
  - Dimensions *d* much less than |*V*|, ranging from 50 to 1,000



Word embeddings provide similar vector representations for words with similar meanings

- Word representations | dense word vectors | word embeddings | word vectors
  - Each word is a **dense** vector of size *d*
  - Dense: values are real-valued numbers, can be negative
  - Dimensions *d* much less than |*V*|, ranging from 50 to 1,000

- Formally
  - $V = \{w_1, w_2, \ldots, w_{|V|}\}$: vocabulary of a language
  - $|V|$: size of the vocabulary
  - w_1: first word in the vocabulary; w_i: word i^{th} in the vocabulary
  - $\boldsymbol{x}_i \in R^d$: vector of the i^{th} word w_i in the vocabulary
  - Example
    - ✓ $\boldsymbol{x}_1$ = [0.6, 0.9, 0.1, 0.4, -0.7, -0.3, -0.2]
    - ✓ $\boldsymbol{x}_2$ = [0.5, 0.8, -0.1, 0.2, 0.7, -0.5, -0.3]

Word Embeddings(part of Input representation but so big)

How can we get Word embeddings?

- Most approaches follow the distributional hypothesis (Firth, 1957) to learn word embeddings

"You shall know a word by the company it keeps"

- In other words, we can guess the meaning of a word by its context, i.e., other words in the text containing the current word.

Maybe I will see a reindeer with red nose this Christmas.

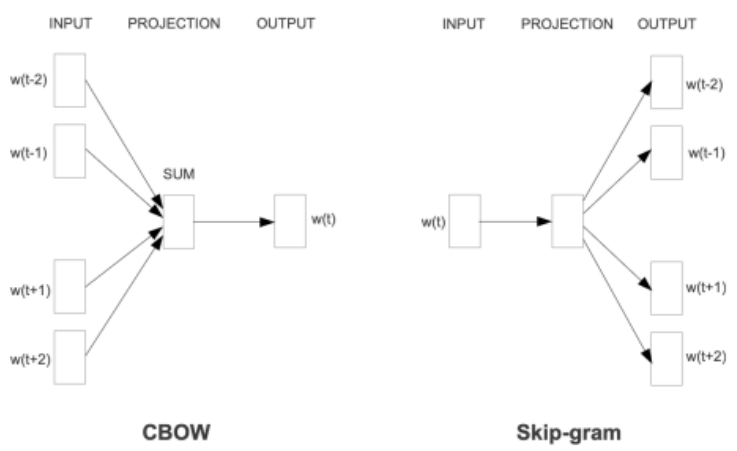
if reindeer is the current word, what is the context?

- Word2vec
  - SkipGram: given a word, predicts the possible context words

? ? ? ? reindeer ? ? ? ?

  - CBOW: given a context, predicts a possible word in the context

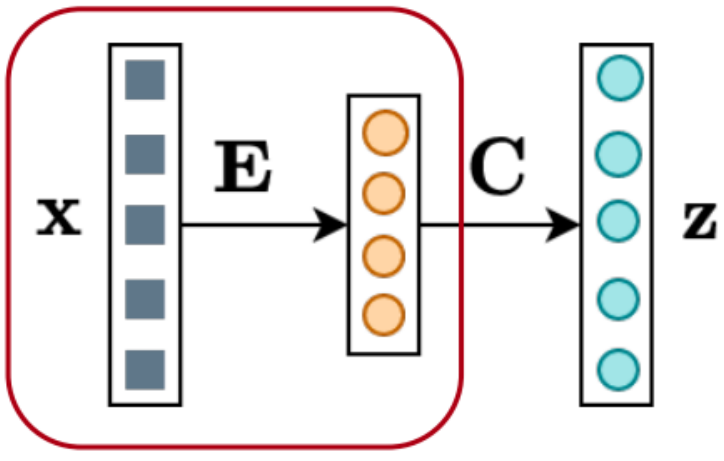
Maybe I will see a ? with red nose this Christmas



Word analogies: Paris - France + Italy = Rome

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

1) Skip-Gram



- $\mathbf{x} \in R^{|V|}$: one-hot word vector
- $\mathbf{E} \in R^{|V| \times d}$: word embedding matrix
- $\mathbf{z} \in R^{|V|}$: predicted context words
- $\mathbf{C} \in R^{d \times |V|}$: context embedding matrix

1. Get one-hot vector ***x*** of the target word *t*
2. Get the word vector ***e*** for the target word

$$\underbrace{\begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix}}_{\text{one-hot vector}} \times \begin{bmatrix}8 & 2 & 1 & 9\\ 6 & 5 & 4 & 0\\ 7 & 1 & 6 & 2\\ 1 & 3 & 5 & 8\\ 0 & 4 & 9 & 1\end{bmatrix} = \underbrace{\begin{bmatrix}1 & 3 & 5 & 8\end{bmatrix}}_{\text{Hidden layer output}}$$

Embedding weight matrix **(E)**

3. Generate a score vector ***z***

$$\underbrace{\begin{bmatrix}1 & 3 & 5 & 8\end{bmatrix}}_{\text{Hidden layer output}} \times \begin{bmatrix}6 & 0 & 2 & 4 & 4\\ 3 & 6 & 5 & 2 & 4\\ 4 & 4 & 4 & 2 & 4\\ 1 & 3 & 3 & 3 & 1\end{bmatrix} = \underbrace{\begin{bmatrix}43 & 62 & 61 & 44 & 44\end{bmatrix}}_{\text{score vector}}$$

Context embedding matrix **(C)**

4. Turn the score vector into probabilities with softmax

   y'=softmax(***z***)

$$\begin{bmatrix}4.1e-09 & 0.731 & 0.269 & 1.1e-08 & 1.1e-08\end{bmatrix}$$

5. ***y'*** should match the true probabilities of context words

- After training is finished, we only care about the embedding matrix **E**
- We multiply a word's one-hot vector with E, to get the embedding of the word

$|V|$ is huge

→ negative sampling

→ binary classification: whether a word is in the context

✓ For a target word t, treat surrounding words from the context are positive examples

✓ Randomly select other words in vocabulary as negative examples

✓ Train a classifier to distinguish those two cases

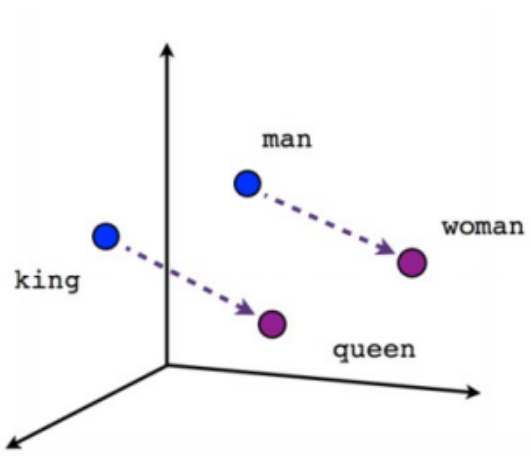
✓ Use the learned weights E as the embeddings

2) Properties of Word Embeddings

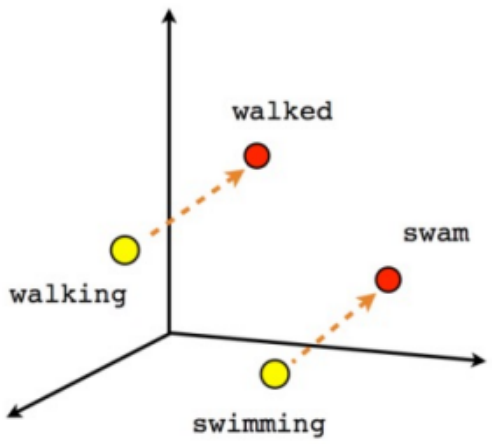
- Semantically similar words are near in the vector space

A two-dimensional (t-SNE) projection of 60 dimensional embeddings trained for sentiment analysis

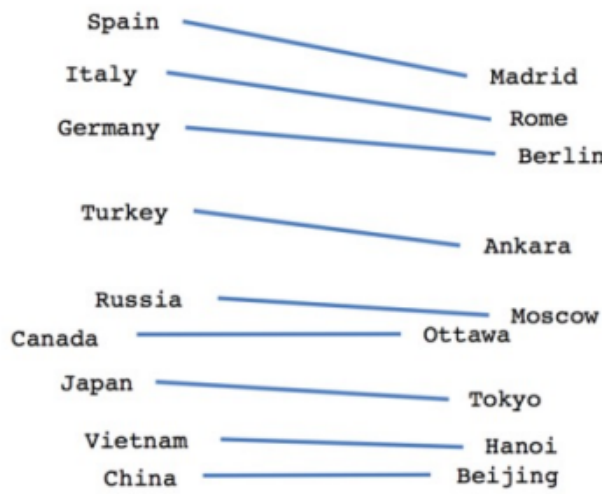
- Syntax/Morphology properties are (sometimes) also preserved



Male-Female



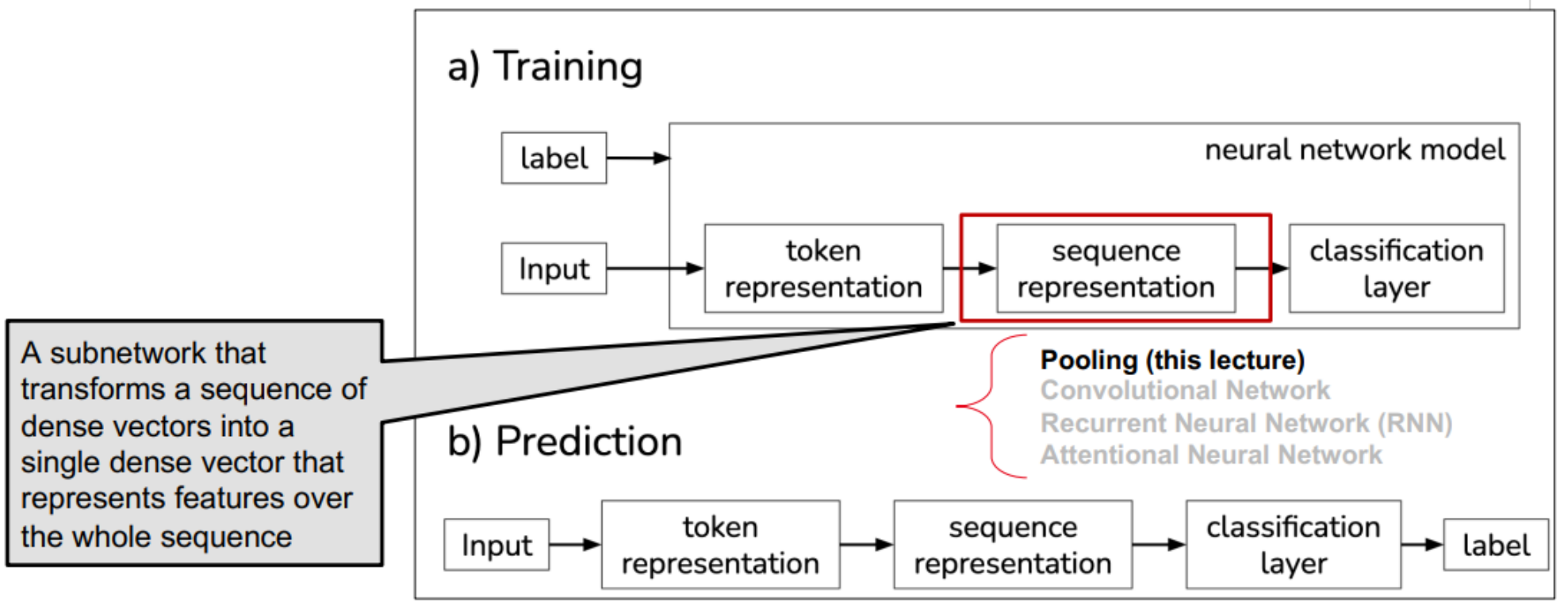
Verb tense



Country-Capital

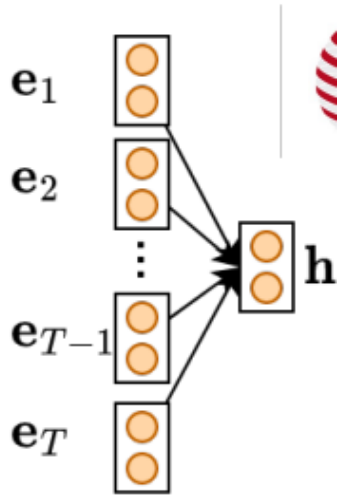
End Of Word Embeddings

b) sequence representation



Pooling

- Pooling based sequence representation



  - Sum pooling

$$\text{sum}(\text{emb}(\mathbf{X}_{1:T})) = \sum_{i=1}^{T} \mathbf{e}_i$$

  - Average pooling

$$\text{avg}(\text{emb}(\mathbf{X}_{1:T})) = \frac{1}{T} \sum_{i=1}^{T} \mathbf{e}_i$$

  - Max pooling

$$\max(\text{emb}(\mathbf{X}_{1:T})) = \left\langle \max_{i=1}^{T} \mathbf{e}_i[1] \max_{i=1}^{T} \mathbf{e}_i[2] \ldots \max_{i=1}^{T} \mathbf{e}_i[d] \right\rangle$$

  - Min pooling

$$\min(\text{emb}(\mathbf{X}_{1:T})) = \left\langle \min_{i=1}^{T} \mathbf{e}_i[1] \min_{i=1}^{T} \mathbf{e}_i[2] \ldots \min_{i=1}^{T} \mathbf{e}_i[d] \right\rangle$$

Stop Words

- Stop words are very frequent words but uninformative
  - ✓ Because they are too frequent, they are insignificant for any classification task
  - ✓ Common stop words in English

    | a | the | on | of | with | about | and | in | at | to | " | , | ? | oh | . |
- How to get stop words?
  - From scratch: the top 10 or 50 frequent words in the training set
  - Using existing word lists

Unknown Words

- Are there unknown words?
  - ✓ Usually, appear in our test data but not in the training data
- Sometimes we ignore them
  - ✓ Remove them from test data
  - ✓ Why not include unknown words?
    - Not helpful for classification
- Use an unknown word vector
  - ✓ Randomly initialized
  - ✓ Set as the average all word vectors

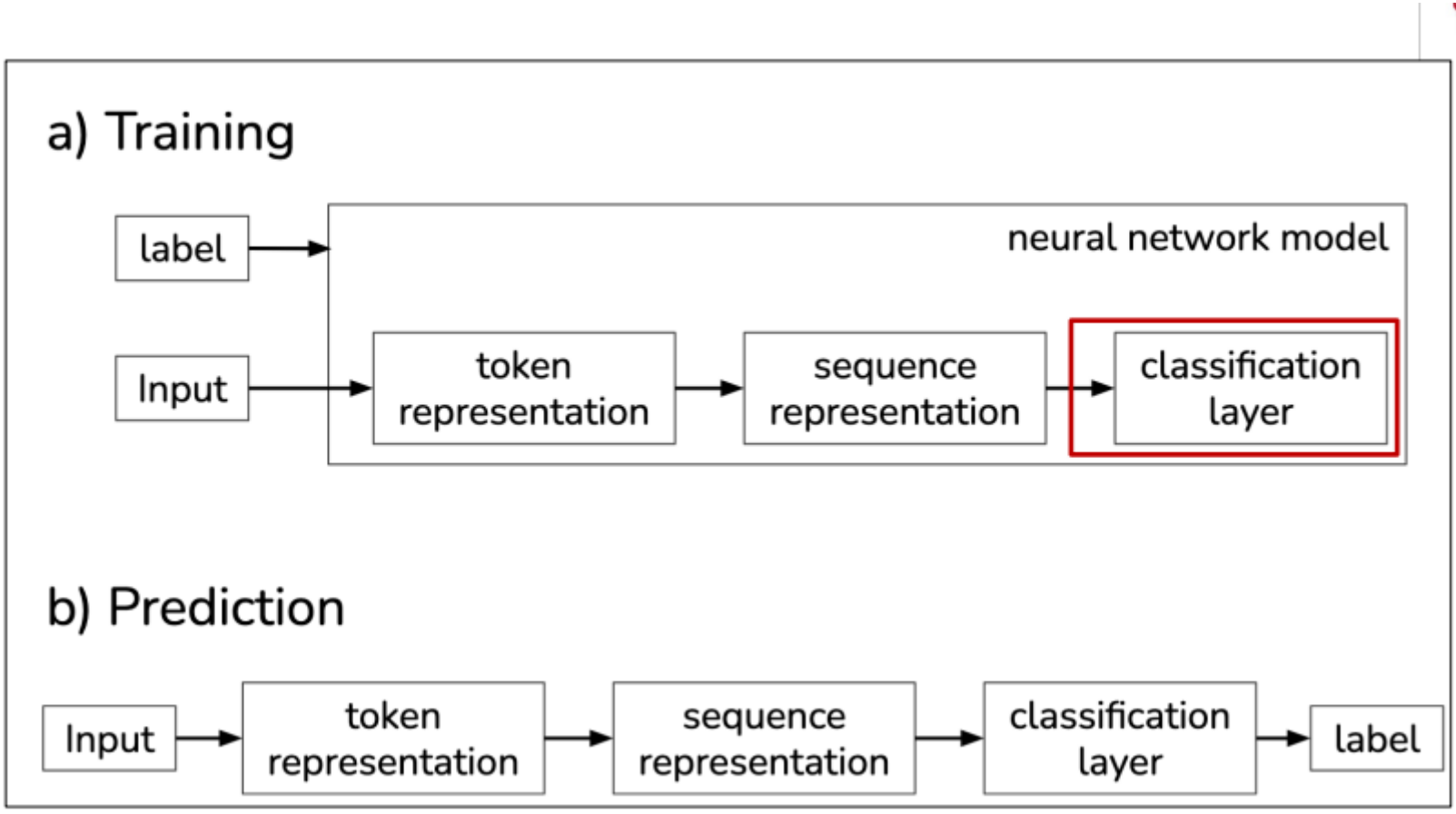
Fasttext word embeddings

- provides embeddings for character n-grams
- can handle out-of-vocabulary (OOV) words

  e.g., fan-bloody-tastic

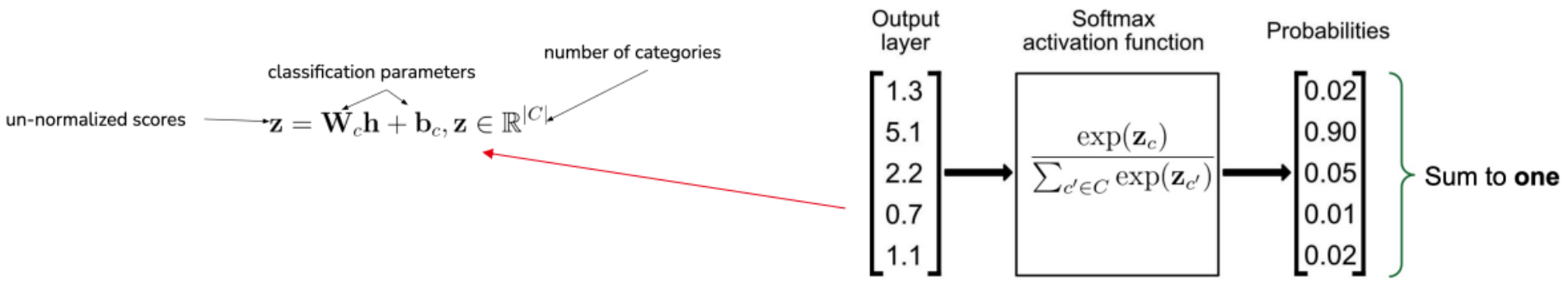
→ by summing up vectors of the corresponding char-n-grams

c) classification layer



Active Function: Softmax (Lecture 4)

- Take a vector and compute a probability distribution out of the vector
  - Each score resides in [0, 1]
- Normally is used as the activation function in the output layer that predicts a multinomial probability distribution
  - Sum to **one**



Summary :Supervised Neural Text Classification

o Token representation - Get from self-supervised tasks
o Sequence representation - Simplest operation: pooling
o Classification - Softmax function