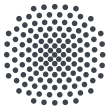


Universität Stuttgart
Institut für Maschinelle
Sprachverarbeitung

Gerhard Kremer
Kerstin Jung

Hackatorial: Maschinelles Lernen lernen

CRETA

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Überblick

Hackatorial „Maschinelles Lernen lernen“

- Entstanden im Projekt CRETA
- Realisiert durch Mitarbeiter und Studierende

CRETA

CENTER FOR REFLECTED TEXT ANALYTICS

- Interdisziplinär:
  Mediävistik, Neuere Deutsche Literatur, Digital Humanities, Sozialwissenschaften, Visualisierung, Computerlinguistik
- Ziele:
  - Reflektierte Textanalyse
  - Einsatz von quantitativen Verfahren
    für geistes- und sozialwissenschaftliche Fragestellungen
  - Entitäten und Relationen
  - Annotation als passende Methodik
- Demnächst: Coaching-Stipendien

Überblick

Hackatorial „Maschinelles Lernen lernen“

- Entstanden im Projekt CRETA
- Realisiert durch Mitarbeiter und Studierende
- Hackatorial
  = Hackathon + Tutorial
  = (Hacking + Marathon) + Tutorial
- Basiert auf der Shared Task-Idee

Shared Task

- Mehrere Gruppen treten an
- Gleiche, klar definierte Aufgabe
- Gleiche Daten
- Daten für Evaluation vorerst unbekannt
- Evaluation nach vorab definierten Kriterien
- Kompetitives Format,
  große Bedeutung in der Computerlinguistik

Aufgaben

- Entwicklung und Training eines maschinellen Lernverfahrens zur Erkennung von Entitätenreferenzen in Texten (Datengrundlage: annotierte Korpora)
- Auswahl einer Kombination von Trainings-Merkmalen … durch Verändern eines Python-Skripts
- Lokales Training eines Modells für eines von drei Korpora und Performanz-Test
- Am Ende: Anwendung des finalen Modells auf die (noch geheimen) Testdaten
- Gemeinsame Auswertung der Ergebnisse

Grundideen

- Realistisches Szenario
  - Skripte sind in Gemeinschaftsarbeit gewachsen
    (und daher tlw. komplexer als nötig)
  - Echter Code:
    Skript führt genau aus, was eingegeben wird
  - Kann auch über das Hackatorial hinaus genutzt werden
- Für Programmiereinsteiger:
  - Nur Mut!
    (fast alles kann rückgängig gemacht werden)
  - Wir sind da, um zu helfen

Ablauf

- Theorie
  - Korpus und Annotationen
  - Grundlagen maschinellen Lernens
- Praxis
  - Technisches Setup und Starthilfe
  - Änderungen an und Aufrufen der Skripte
- Entwicklung
- Test: Upload der annotierten Daten auf den Server
- Evaluationsmaße und Besprechen der Ergebnisse

Einschub: Technisches Setup

Installations-Test

Bei Schwierigkeiten: Kaffeepause

Korpus und Annotationen

Korpus

Subkorpus	Beschreibung
Parzival	Mittelhochdeutsche Texte – Reimpaarverse (12./13. Jhd.)
Werther	Goethe's *Die Leiden des jungen Werther* – Briefroman, veröff. 1774
Bundestagsdebatten	Protokolle der Reden von Abgeordneten – Stenographische Mitschrift (1996-2015)

- Heterogenes Korpus (Inhalt und Form/Stil)
- Überlappende Forschungsinteressen (Beziehungen, Orte, Konflikte)

Entitäten

... relevant im multidisziplinären CRETA-Szenario
... gemeinsamer Workflow Grundlage für Zusammenarbeit

- Parzival:
  Personen und Orte
- Werther:
  Personen, Orte, Werke
- Bundestagsdebatten:
  Personen, Orte, Organisationen

Annotationen

Text

Abbildung: Entitäten und Entitätsreferenzen

Annotationen

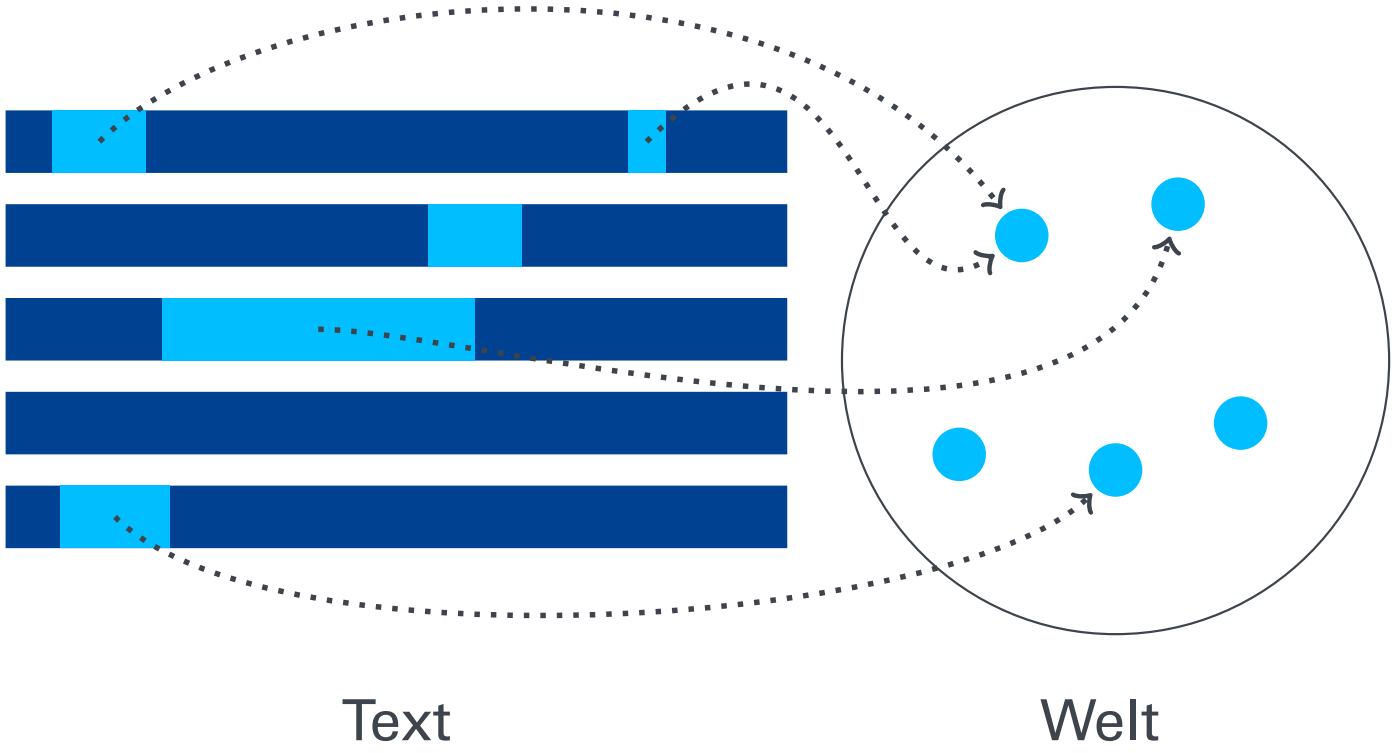


Abbildung: Entitäten und Entitätsreferenzen

Annotationen

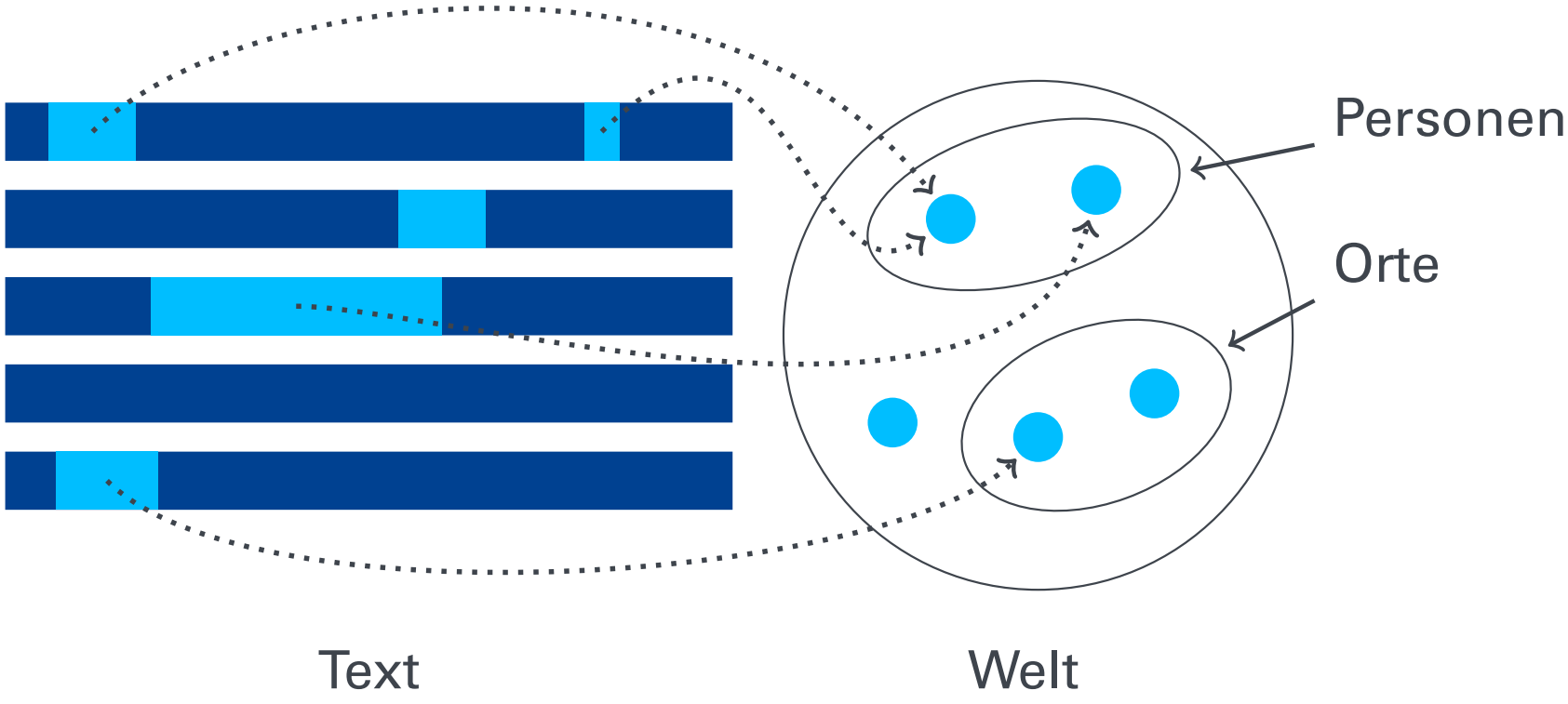


Abbildung: Entitäten und Entitätsreferenzen

Annotationsrichtlinien

… für Referenzausdrücke im Text

Annotiert wurden …

- Eigennamen („Werther“)
- Appellative („der junge Mann“) – wenn sie referieren –
  - Gruppen („die beiden Ritter“)
  - Anreden („Mein Liebster“)

Jedoch nicht annotiert wurden …

- generische Ausdrücke („der Kanzler hat das Recht …“)
  (vgl. spezifischer Ausdruck: „der Kanzler fliegt morgen nach …“)
- Pronomen

Annotationsrichtlinien

Wie wurde annotiert?

- Maximale Nominalphrasen
  - inkl. Relativsätze:
    „der Kanzler, der sich zu dieser Zeit in Berlin aufhielt“
  - inkl. Appositionen: „Wilhelm, mein Freund“
- Artikel inkl. Präposition, wenn verschmolzen: „im Land“
- Eingebettete Phrasen: [Wolfram von [Eschenbach]$_{LOC}$]$_{PER}$
- Kontextabhängiges Annotieren:
  „Ich wollte schon immer mal nach [Europa]$_{LOC}$“
  „[Europa]$_{ORG}$ zwingt Griechenland einen Sparkurs auf“

Annotationsbeispiele

Text	Klasse	Beispiele
Parzival	PER	Parzival, der ritter, die tafelrunde
	LOC	Nantes, der wald Braziljan, der palas
Werther	PER	Werther, die Kinder aus dem Dorfe, Liebster Wilhelm
	LOC	Die Schweiz, Dem Dorfe
	WRK	Emilia Galotti
Bundestags-debatten	PER	Angela Merkel, die Abgeordneten
	LOC	Großbritannien, das Land, Europa
	ORG	SPD, die Union, Europa

Textspezifische Besonderheiten

- Parzival
  - Mhd. Sprache
  - Eigennamen mit großem Anfangsbuchstaben
  - 30-Vers-Segmente (jeweils 1. Wort großgeschrieben)
  - fast alle anderen Wörter kleingeschrieben
- Werther
  - 1878: „veraltete“ Sprache
  - Briefform: Ich-Erzähler
  - Emotional gefärbte Erzählweise
    (z.B. lange Sätze, Interjektionen)
- Bundestagsdebatten
  - nicht-erzählender Text
  - direkte Reden
  - zeitgenössischer Stil und Inhalt

Korpus: Einsicht in Datei

Grundlagen maschinellen Lernens

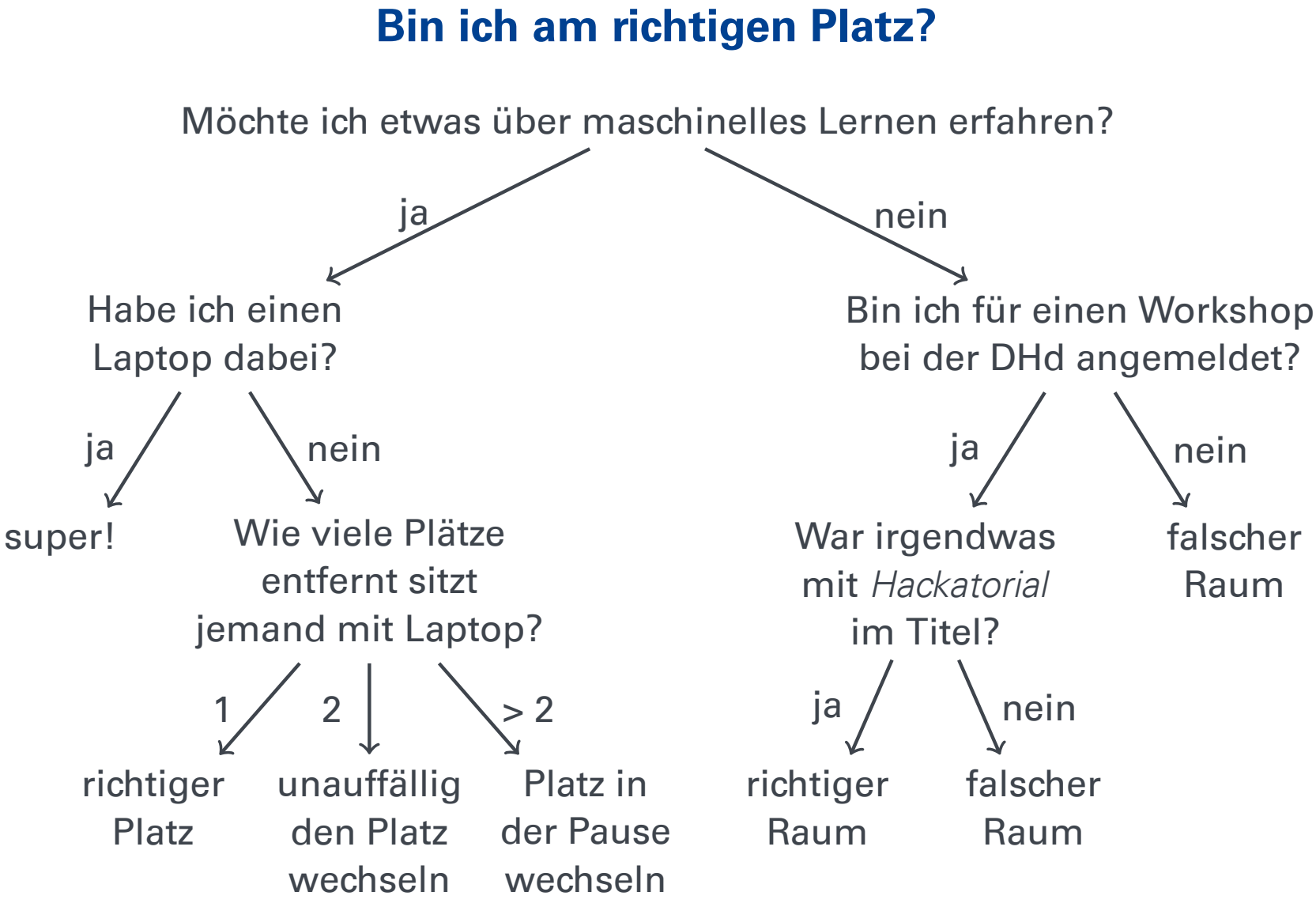
Maschinelles Lernen – Basics

- Mustererkennung in Daten
  - Erkennung von Mustern, versteckten Zusammenhängen und unbekannten Beziehungen in Daten
- Vielfältige Anwendungen
  - Börsenhandel, Suchmaschinen, Überwachung, datengetriebene Wissenschaft, automatische Übersetzung/Textanalyse
- Relevanz für die Digital Humanities
  - *Big data*-Analysen
  - Finden von interessanten Fällen, die sich nicht so verhalten wie die anderen
- Analyse von Daten: Inspektion von Fehlern und/oder Abhängigkeiten zwischen Merkmalen

Klassifikation

- Zuweisung von Kategorien zu Objekten
  - Wörter $\rightarrow$ Wortarten
  - Tierfotos $\rightarrow$ Tierart
  - Texte $\rightarrow$ Gattungen
- Vorhersagemodell: Verantwortlich für die Zuweisung
- Viele Möglichkeiten, z.B.
  - Entscheidungsbäume
  - Naive Bayes
  - Neuronale Netze
  - ...

Beispiel – Entscheidungsbaum



Merkmale (= Features)

- Entscheidung basiert auf Merkmalen
- **Merkmale sind alles, was der Vorhersagealgorithmus sieht**

Richtiger Platz?

Etwas über ML erfahren?

Wie viele Plätze entfernt?

Hackatorial im Titel?

...

Ist das Wort ein Nomen?

Ist es großgeschrieben?

Wie viele Buchstaben lang?

Welche Wortart hat das Wort davor?

...

- Datentypen: Binär (ja/nein), numerisch, kategorial
- Merkmalsextraktion:
  Extraktion von Merkmalsausprägungen aus Daten

Was bedeutet dann „Lernen“?

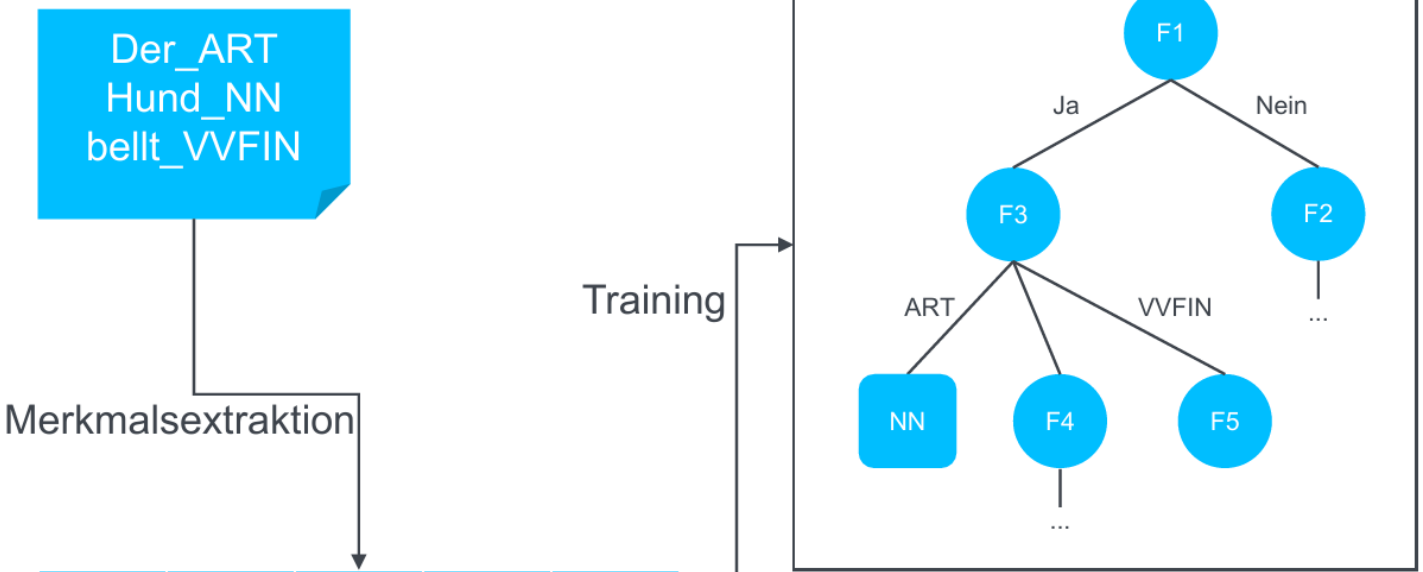
- Vorhersagemodelle können auf beliebige Daten angewendet werden, um neue Vorhersagen zu bekommen
- Manuell erzeugte Modelle (insb. Entscheidungsbäume)
- Automatisches Erzeugen von Modellen aus Daten (= Training) $\Rightarrow$ **Maschinelles Lernen**

ML-Zutaten:

- Trainingsdaten = Daten mit bekannter Klassifikation (durch Annotation)
- Lernalgorithmus ermittelt ein Modell, das die Trainingsdaten möglichst gut vorhersagt

Maschinelles Lernen – Ablauf

Wortartenerkennung

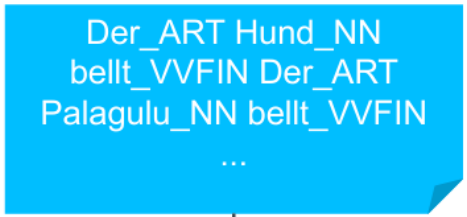


Wort	F1	F2	F3	K
Der	Ja	3		ART
Hund	Ja	4	ART	NN
bellt	Nein	5	NN	VVFIN

Was haben wir gelernt?

- Reproduktion der Trainingsdaten funktioniert gut, ist aber nicht sehr interessant: Computer können Dinge speichern
- Ziel: Generalisierung
  - Zusammenhänge, die in ähnlichen Situationen (= bei ähnlichen Featurewerten) auch funktionieren:
    Der Palagulu bellt. $\rightarrow$ *Palagulu* ist das Nomen, auch wenn das Wort nicht im Duden steht
- Test auf Trainingsdaten ist **kein realistisches Szenario**
  $\Rightarrow$ **Separate Testdaten**

Maschinelles Lernen – Training, Test

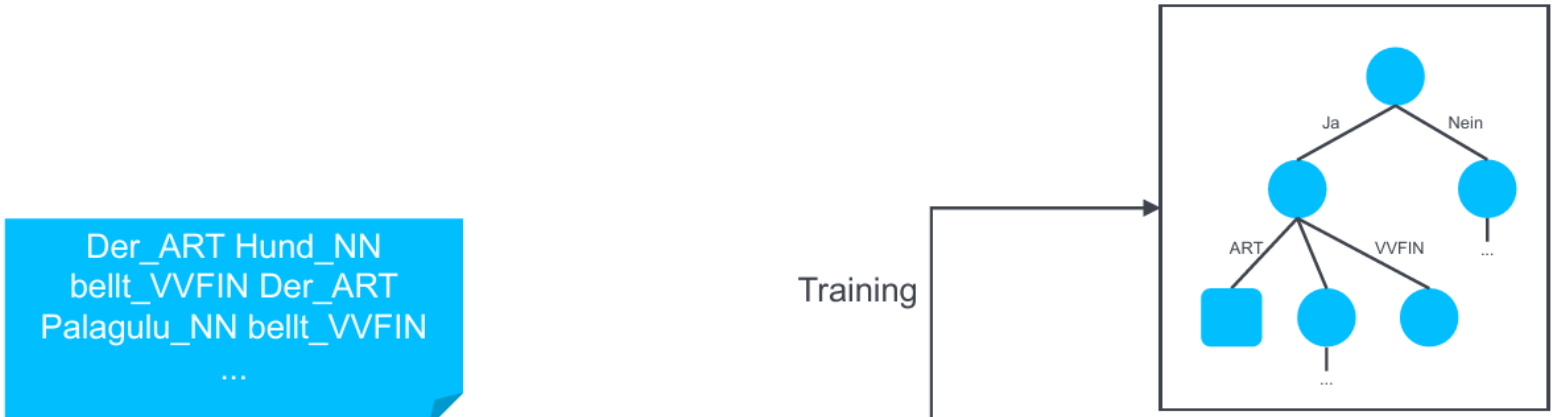


Trainingsdaten

Testdaten

Wort	F1	F2	F3	K
Der	Ja	3		ART
Hund	Ja	4	ART	NN
bellt	Nein	5	NN	VVFIN
Der	Ja	3		ART
Palagulu	Ja	8	ART	NN
bellt	Nein	5	NN	VVFIN

Maschinelles Lernen – Training, Test



Training

Der_ART Hund_NN bellt_VVFIN Der_ART Palagulu_NN bellt_VVFIN ...

Trainingsdaten

Wort	F1	F2	F3	K
Der	Ja	3		ART
Hund	Ja	4	ART	NN
bellt	Nein	5	NN	VVFIN

Testdaten

Der	Ja	3		ART
Palagulu	Ja	8	ART	NN
bellt	Nein	5	NN	VVFIN

Maschinelles Lernen – Training, Test

Der_ART Hund_NN bellt_VVFIN Der_ART Palagulu_NN bellt_VVFIN ...

Training

Trainingsdaten

Wort	F1	F2	F3	K
Der	Ja	3		ART
Hund	Ja	4	ART	NN
bellt	Nein	5	NN	VVFIN

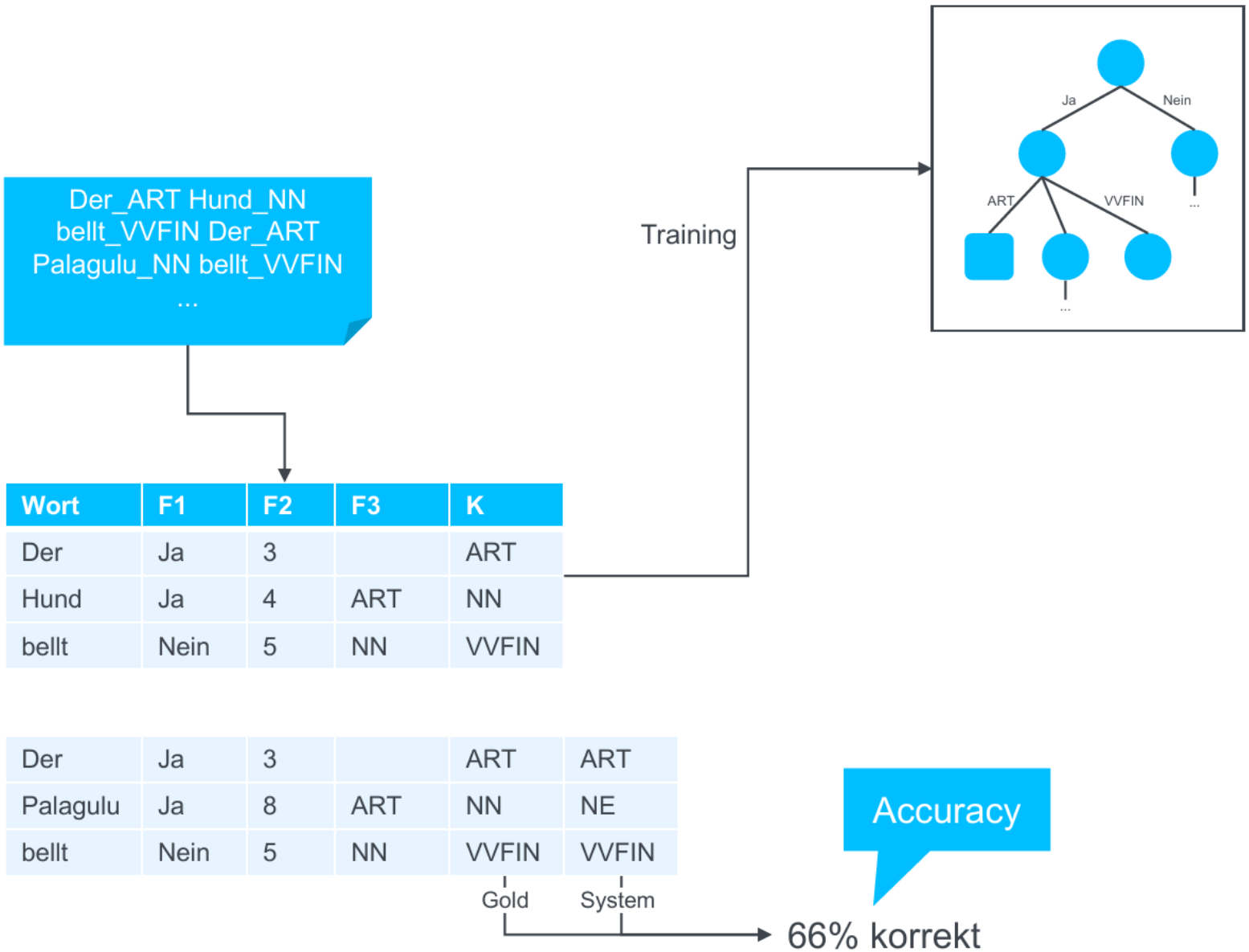
Vorhersage (= system output, silver)

ART
NE
VVFIN

Testdaten

Der	Ja	3		ART
Palagulu	Ja	8	ART	NN
bellt	Nein	5	NN	VVFIN

Maschinelles Lernen – Training, Test



Maschinelles Lernen – Kochrezept

1. Annotiere Daten (oder besorge annotierte Daten)
2. Teile annotierte Daten in Trainings- und Testdaten auf
3. Extrahiere Features aus Trainings- und Testdaten
4. Trainiere ein Vorhersagemodell mit den Trainingsdaten
5. Wende das Modell auf die Testdaten an
6. Überprüfe die Übereinstimmung der Vorhersagen mit den Annotationen
   Ergebnis: z.B. Prozentzahl
7. Vergleiche Prozentzahl mit Baseline, Forschungsstand oder der Konkurrenz

Maschinelles Lernen – Kochrezept

1. Annotiere Daten (oder besorge annotierte Daten)
2. Teile annotierte Daten in Trainings- und Testdaten auf
3. Extrahiere Features aus Trainings- und Testdaten
4. Trainiere ein Vorhersagemodell mit den Trainingsdaten
5. Wende das Modell auf die Testdaten an
6. Überprüfe die Übereinstimmung der Vorhersagen mit den Annotationen
   Ergebnis: z.B. Prozentzahl
7. Vergleiche Prozentzahl mit Baseline, Forschungsstand oder der Konkurrenz

Das machen wir gleich!

Maschinelles Lernen – Kochrezept

1. Annotiere Daten (oder besorge annotierte Daten)
2. Teile annotierte Daten in Trainings- und Testdaten auf
3. Extrahiere Features aus Trainings- und Testdaten
4. Trainiere ein Vorhersagemodell mit den Trainingsdaten
5. Wende das Modell auf die Testdaten an
6. Überprüfe die Übereinstimmung der Vorhersagen mit den Annotationen
   Ergebnis: z.B. Prozentzahl
7. Vergleiche Prozentzahl mit Baseline, Forschungsstand oder der Konkurrenz

Das machen wir am Ende!

Maschinelles Lernen – Entscheidungsbäume

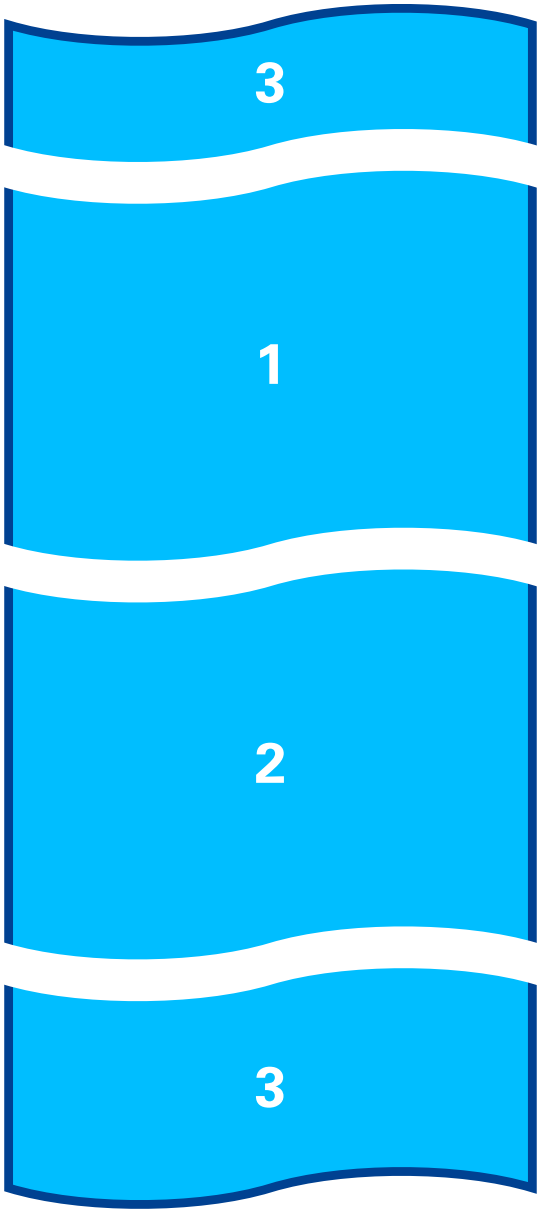
- Lernalgorithmus (rekursiv):
  Schrittweise Aufteilung der Daten in Teilmengen, bis nur noch Teilmengen übrig sind, in denen alle Objekte der gleichen Klasse angehören
- Vorhersagemodell: Folge dem Baumpfad

Maschinelles Lernen – Naive Bayes

- Lernalgorithmus:
  Ermittelt Wahrscheinlichkeit, dass eine Klasse vorliegt, gegeben die Feature-Werte
- Vorhersagemodell:
  Berechne für jede Zielklasse die Wahrscheinlichkeit, indem die Einzel-Wahrscheinlichkeiten für die vorliegenden Feature-Werte multipliziert werden

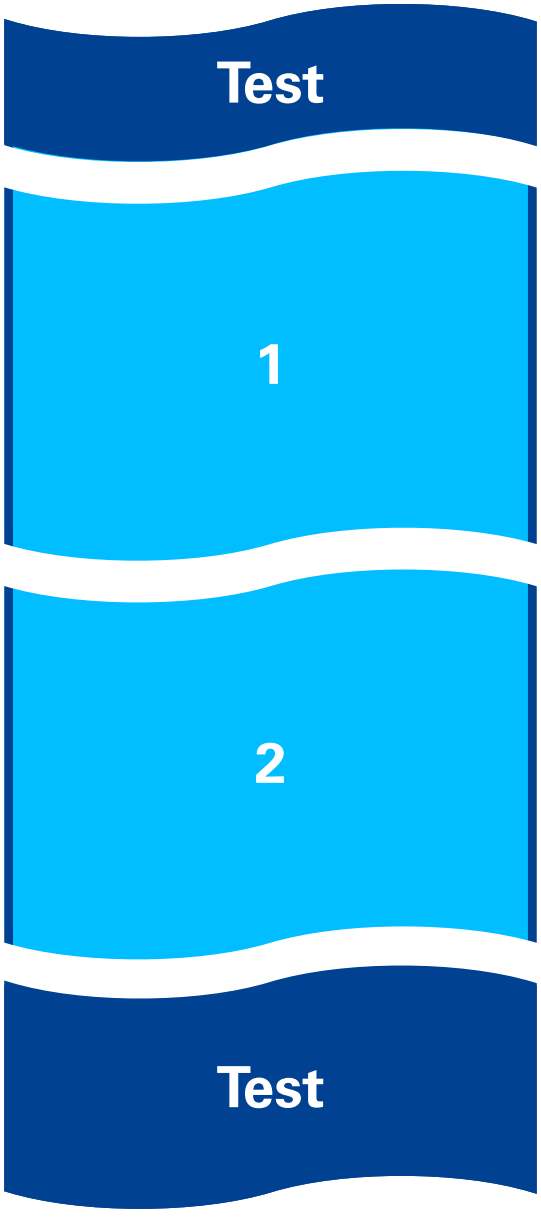
Kreuzvalidierung (cross validation)

Kreuzvalidierung (cross validation)



- zufällige Abschnitte gleicher Größe

Kreuzvalidierung (cross validation)



- zufällige Abschnitte gleicher Größe
- 1 Teil: Test des trainierten Modells,
  Rest als Trainingsdaten
- Dreifach:
  Dreiteilung, 3 Durchgänge

Kreuzvalidierung (cross validation)

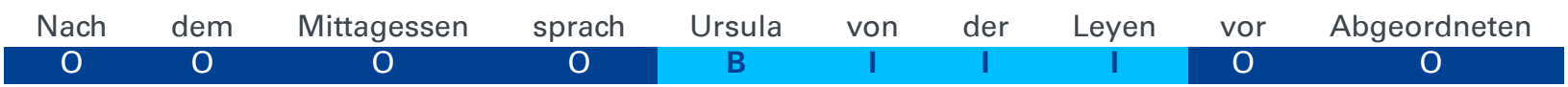
- zufällige Abschnitte gleicher Größe
- 1 Teil: Test des trainierten Modells,
  Rest als Trainingsdaten
- Dreifach:
  Dreiteilung, 3 Durchgänge

Kreuzvalidierung (cross validation)

- zufällige Abschnitte gleicher Größe
- 1 Teil: Test des trainierten Modells, Rest als Trainingsdaten
- Dreifach:
  Dreiteilung, 3 Durchgänge
- Arithmetisches Mittel der Einzelwerte

Erkennung von Entitätsreferenzen

- Entitätsreferenzen können mehrere Wörter umfassen:
  der starke Ritter, *Ursula von der Leyen*
- Einzelwörter nicht ausreichend
- BIO-Schema
  - Begin, Inside, Outside
  - Eine Entitätsreferenz wird auf die Wörter verteilt, die sie umfasst



Wie trainiere ich ein Vorher-sagemodell?

Finde den richtigen Ordner

- Wo ist der Ordner `hackatorial` abgelegt?
- Terminal oder Eingabeaufforderung öffnen

Eingabeaufforderung

- `cd Pfad/zum/hacktorial/code`
  um in den Unterordner `code` zu wechseln

Das Skript zum Trainieren mit python ausführen

- Abhängig von Betriebssystem und Version gibt es folgende Möglichkeiten um python aufzurufen:
  - py
  - python
  - python3
- Beispiele:

```
python train.py ../data/Bundestag_train.tsv

py train.py ..\data\Bundestag_train.tsv
```

Herzlichen Glückwunsch!

Sie haben gerade Ihr erstes Maschine Learning Modell trainiert!

Jetzt verbessern wir das Modell indem wir

- die gewählten Features verändern
- den Trainingsalgorithmus austauschen

Verfügbare Trainingskorpora:

- Parzival_train.tsv
- Werther_train.tsv
- Bundestag_train.tsv

Features auswählen

Schritt 1: Datei `feature_extractor.py` öffnen

Schritt 2: Features nach Wahl zuschalten (= aktivieren) bzw. auskommentieren (= deaktivieren)

- Features, die mit # beginnen sind inaktiv
- Durch Löschen des #-Symbols wird das Feature aktiviert
- Mehrere Features können gleichzeitig aktiv sein

```
        ##############################################################################
        # THIS IS WHERE ALL THE DIFFERENT POSSIBLE FEATURE EXTRACTION FUNCTIONS ARE CALLED #
        # COMMENT THEM IN OR OUT DEPENDING ON WHICH FEATURES YOU FIND USEFUL          #
        ##############################################################################

            # structure of feature function for example of the feature "capitalized":
            # -1 calls the last word that has been appended to the featureset
            # 0 accesses the dictionary which is the first element of the tupel
            # "capitalized" is the feature name

            featureset[-1][0]["surface"] = self.surface(token_dic)
            featureset[-1][0]["surface_backwards"] = self.surface_backwards(token_dic)
            featureset[-1][0]["surface_every_second"] = self.surface_every_second(token_dic)
            featureset[-1][0]["word_length"] = self.word_length(token_dic)
            featureset[-1][0]["pos"] = self.pos(token_dic)
            featureset[-1][0]["lemma"] = self.lemma(token_dic)
            featureset[-1][0]["lemma_backwards"] = self.lemma_backwards(token_dic)
            featureset[-1][0]["lemma_every_second"] = self.lemma_every_second(token_dic)
            featureset[-1][0]["segment_id"] = self.segment_id(token_dic)
            featureset[-1][0]["sent_id"] = self.sent_id(token_dic)
            featureset[-1][0]["word_id"] = self.word_id(token_dic)
            featureset[-1][0]["next_word"] = self.next_word(index, corpus)
            featureset[-1][0]["previous_word"] = self.previous_word(index, corpus)
            featureset[-1][0]["next_lemma"] = self.next_lemma(index, corpus)
            featureset[-1][0]["previous_lemma"] = self.previous_lemma(index, corpus)
            featureset[-1][0]["next_pos"] = self.next_pos(index, corpus)
            featureset[-1][0]["previous_pos"] = self.previous_pos(index, corpus)
            featureset[-1][0]["find_word_in_sentence74"] = self.find_word_in_sentence74(token_dic)
```

Features auswählen

Schritt 1: Datei `feature_extractor.py` öffnen

Schritt 2: Features nach Wahl zuschalten (= aktivieren) bzw. auskommentieren (= deaktivieren)

- Features, die mit # beginnen sind inaktiv
- Durch Löschen des #-Symbols wird das Feature aktiviert
- Mehrere Features können gleichzeitig aktiv sein

```
featureset[-1][0]["word_length"] = self.word_length
featureset[-1][0]["pos"] = self.pos(token_dic)
featureset[-1][0]["lemma"] = self.lemma(token_dic)
```

```
########
# THIS
# COMME
########
    # s
    # -1 calls the last word that has been appended to the featureset
    # 0 accesses the dictionary which is the first element of the tupel
    # "capitalized" is the feature name

    featureset[-1][0]["surface"] = self.surface(token_dic)
    featureset[-1][0]["surface_backwards"] = self.surface_backwards(token_dic)
    featureset[-1][0]["surface_every_second"] = self.surface_every_second(token_dic)
    featureset[-1][0]["word_length"] = self.word_length(token_dic)
    featureset[-1][0]["pos"] = self.pos(token_dic)
    featureset[-1][0]["lemma"] = self.lemma(token_dic)
    featureset[-1][0]["lemma_backwards"] = self.lemma_backwards(token_dic)
    featureset[-1][0]["lemma_every_second"] = self.lemma_every_second(token_dic)
    featureset[-1][0]["segment_id"] = self.segment_id(token_dic)
    featureset[-1][0]["sent_id"] = self.sent_id(token_dic)
    featureset[-1][0]["word_id"] = self.word_id(token_dic)
    featureset[-1][0]["next_word"] = self.next_word(index, corpus)
    featureset[-1][0]["previous_word"] = self.previous_word(index, corpus)
    featureset[-1][0]["next_lemma"] = self.next_lemma(index, corpus)
    featureset[-1][0]["previous_lemma"] = self.previous_lemma(index, corpus)
    featureset[-1][0]["next_pos"] = self.next_pos(index, corpus)
    featureset[-1][0]["previous_pos"] = self.previous_pos(index, corpus)
    featureset[-1][0]["find_word_in_sentence74"] = self.find_word_in_sentence74(token_dic)
```

Trainingsalgorithmus austauschen

Schritt 1: Datei `train.py` öffnen

Schritt 2: Eine der Zeilen beginnend mit `trainer =` auskommentieren

- DTTrainer: Entscheidungsbaum
- NBTrainer: Naive Bayes

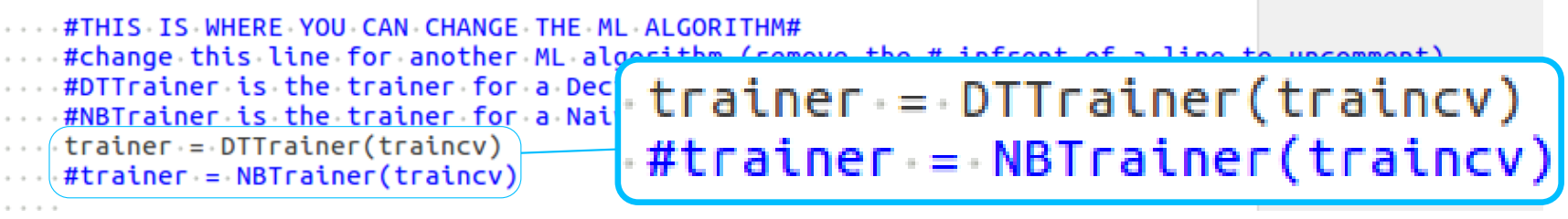
```
    #THIS IS WHERE YOU CAN CHANGE THE ML ALGORITHM#
    #change this line for another ML algorithm (remove the # infront of a line to uncomment)
    #DTTrainer is the trainer for a Decision Tree classifier
    #NBTrainer is the trainer for a Naive Bayes classifier
    trainer = DTTrainer(traincv)
    #trainer = NBTrainer(traincv)
    
```

Trainingsalgorithmus austauschen

Schritt 1: Datei `train.py` öffnen

Schritt 2: Eine der Zeilen beginnend mit `trainer =` auskommentieren

- DTTrainer: Entscheidungsbaum
- NBTrainer: Naive Bayes



Viel Erfolg beim Training!

Test auf externen Daten und Upload

Schritt 1: Datei `test.py` öffnen

Schritt 2: Algorithmus auswählen (`trainer =`)

- DTTrainer: Entscheidungsbaum
- NBTrainer: Naive Bayes

Schritt 3: Aufruf anpassen

Beispiel:

```
python3 test.py ../data/Parzival_train.tsv pa TeamA
```

Schritt 1: Datei `test.py` öffnen

Schritt 2: Algorithmus auswählen (`trainer =`)

- DTTrainer: Entscheidungsbaum
- NBTrainer: Naive Bayes

Schritt 3: Aufruf anpassen

Beispiel:

```
python3 test.py ../data/Parzival_train.tsv pa TeamA
```

Schritt 1: Datei `test.py` öffnen

Schritt 2: Algorithmus auswählen (`trainer =`)

- DTTrainer: Entscheidungsbaum
- NBTrainer: Naive Bayes

Schritt 3: Aufruf anpassen

Beispiel:

```
python3 test.py ../data/Parzival_train.tsv pa TeamA
```

Schritt 1: Datei `test.py` öffnen

Schritt 2: Algorithmus auswählen (`trainer =`)

- DTTrainer: Entscheidungsbaum
- NBTrainer: Naive Bayes

Schritt 3: Aufruf anpassen

Beispiel:

```
python3 test.py ../data/Parzival_train.tsv pa TeamA
```

Schritt 1: Datei `test.py` öffnen

Schritt 2: Algorithmus auswählen (`trainer =`)

- DTTrainer: Entscheidungsbaum
- NBTrainer: Naive Bayes

Schritt 3: Aufruf anpassen

Beispiel:

```
python3 test.py ../data/Parzival_train.tsv pa TeamA
```

Werther:	`we`
Parzival:	`pa`
Bundestag:	`bu`

Schritt 1: Datei `test.py` öffnen

Schritt 2: Algorithmus auswählen (`trainer =`)

- DTTrainer: Entscheidungsbaum
- NBTrainer: Naive Bayes

Schritt 3: Aufruf anpassen

Beispiel:

```
python3 test.py ../data/Parzival_train.tsv pa TeamA
```

Werther:	`we`
Parzival:	`pa`
Bundestag:	`bu`

Evaluation

Accuracy („Richtigkeit“)

Welcher Anteil aller Wörter wurde korrekt klassifiziert?

$$A = \frac{\textbf{korrekt klassifizierte}}{\textbf{alle Wörter}}$$

Accuracy („Richtigkeit“)

Welcher Anteil aller Wörter wurde korrekt klassifiziert?

$$A = \frac{\textbf{korrekt klassifizierte}}{\textbf{alle Wörter}}$$

- Wert zwischen 0 und 1,
  Angabe in %
- als Grobmaß
- aber: accuracy paradox

Accuracy („Richtigkeit“)

Welcher Anteil aller Wörter wurde korrekt klassifiziert?

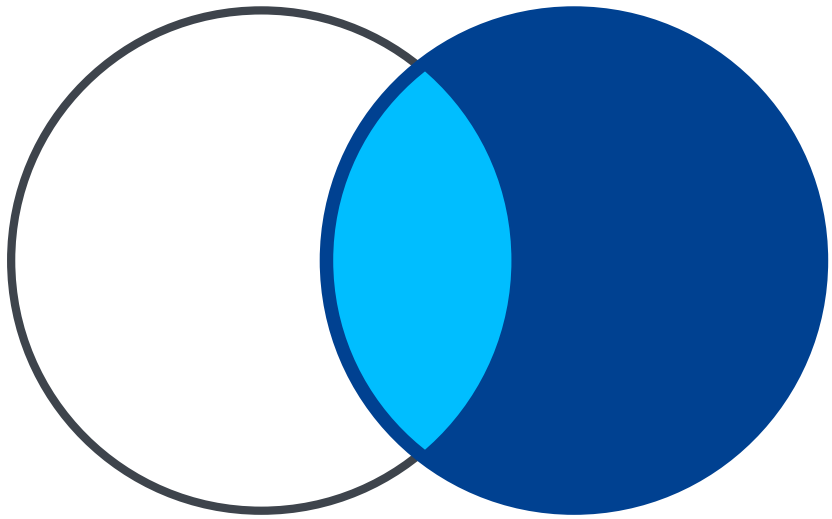
$$A = \frac{\textbf{korrekt klassifizierte}}{\textbf{alle Wörter}}$$

		System-Tipp			
		KAT1	KAT2	…	OTHER
Gold-Annotation	KAT1	**30**	5	…	670
	KAT2	45	**18**	…	…
	⋮	⋮	⋮	⋮	⋮
	OTHER	15	…	…	**9171**

Precision („Genauigkeit“)

Welcher Anteil der als X klassifizierten Entitäten korrekt?

$$P = \frac{\textbf{korrekt} \quad \textbf{als X klassifiziert}}{\textbf{alle Wörter, die} \quad \textbf{als X klassifiziert}}$$



Gold-Annotation

System-Tipp

Precision („Genauigkeit“)

Welcher Anteil der als X klassifizierten Entitäten korrekt?

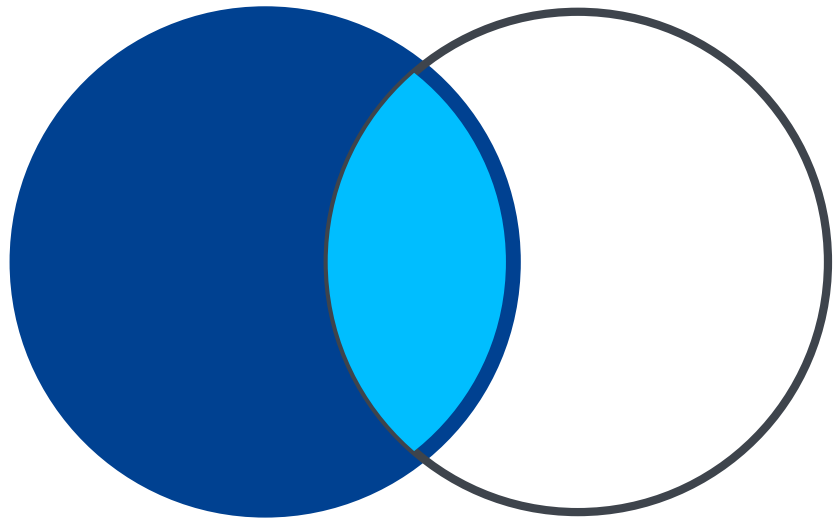
$$P = \frac{\textbf{korrekt}}{\textbf{alle Wörter, die}} \quad \frac{\textbf{als X klassifiziert}}{\textbf{als X klassifiziert}}$$

		System-Tipp		
		KAT1	KAT2	…
Gold-Annotation	KAT1	30	5	…
	KAT2	45	18	…
	⋮	⋮	⋮	⋮

Recall („Vollständigkeit“)

Welcher Anteil an Entitäten der Klasse X erkannt?

$$R = \frac{\textbf{korrekt als X klassifiziert}}{\textbf{Entitäten der Klasse X}}$$



Gold-Annotation

System-Tipp

Recall („Vollständigkeit“)

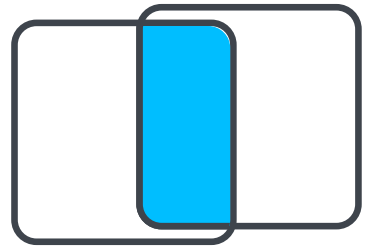
Welcher Anteil an Entitäten der Klasse X erkannt?

$$R = \frac{\textbf{korrekt als X klassifiziert}}{\textbf{Entitäten der Klasse X}}$$

		System-Tipp		
		KAT1	KAT2	…
Gold-Annotation	KAT1	30	5	…
	KAT2	45	18	…
	⋮	⋮	⋮	⋮

Precision ⇆ Recall

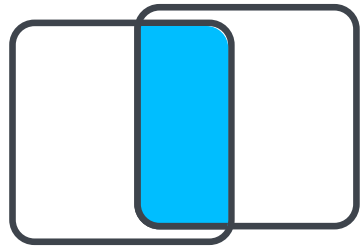
Gold-
Annotation



System-
Tipp

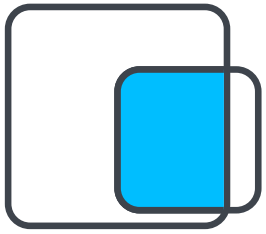
Precision ⇆ Recall

Gold-
Annotation



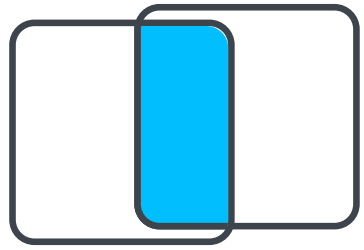
System-
Tipp

Precision erhöht:



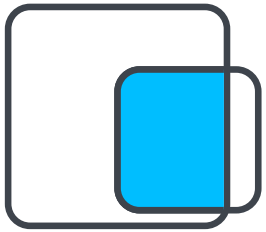
Precision ⇆ Recall

Gold-
Annotation

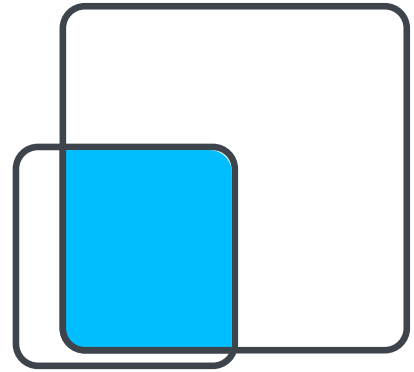


System-
Tipp

Precision erhöht:



Recall erhöht:




Ergebnisse!