Computer-Linguistische Anwendungen

CLA | B.Sc. | LMU

Maschinelles Lernen

Computerlinguistische Anwendungen

40”

The Big Picture

Computerlinguistische Anwendungen

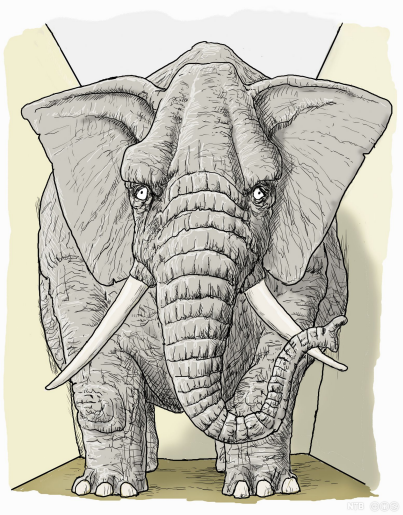
PW

Present in bullet points the big picture of why computational linguistics (CL) contributes to understanding of language. Start with early developments of CL, how they lead to current trends in language modelling and how CL applications are important not only for NLP, but AI and society in general

Early Developments of Computational Linguistics:

- The early developments of Computational Linguistics (CL) date back to the 1940s and 1950s, when researchers started to use computers to process natural language.
- The first significant breakthrough was in the 1950s with the development of the first machine translation systems, which used rules-based approaches to translate text from one language to another.
- Later on, the development of statistical language models in the 1990s helped to improve the accuracy of language processing systems.

Elephant in the Room



Creator: Andrew Pinder | Credit: Ikon Images

The Big Picture

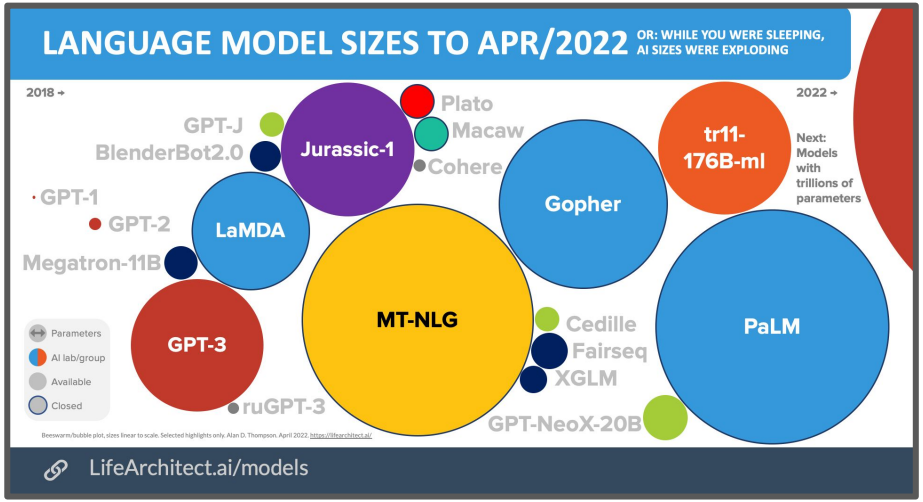
Computerlinguistische Anwendungen

Aktuelle Trends in der Sprachmodellierung:

- Aktuelle Trends in der CL konzentrieren sich auf den Aufbau von **Machine-Learning-Modellen**, die natürliche Sprache verstehen und erzeugen können

- Eine der bedeutendsten Entwicklungen in den letzten Jahren war der Aufstieg von Deep-Learning-Techniken wie rekurrenten neuronalen Netzwerken (RNNs) und Transformers, die sich als effektiv bei der Verarbeitung natürlicher Sprache erwiesen haben

- Diese Techniken werden in einer Vielzahl von Anwendungen eingesetzt, einschließlich Spracherkennung, Textklassifikation, maschineller Übersetzung und Frage-Antwort-Systemen



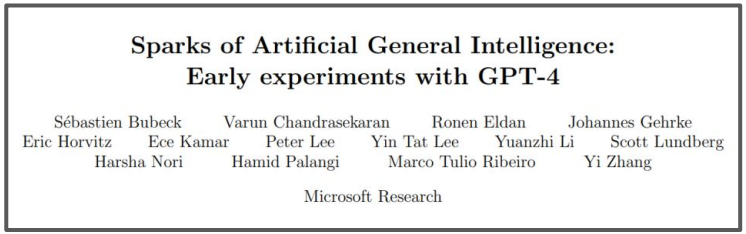
CC: Dr Alan D. Thompson, LifeArchitect.ai (Jan/2023).

The Big Picture

Computerlinguistische Anwendungen

Bedeutung von CL-Anwendungen:

- CL-Anwendungen sind nicht nur für die Verarbeitung natürlicher Sprache (NLP), sondern auch für künstliche Intelligenz (KI) und die Gesellschaft im Allgemeinen von Bedeutung.
- Im Bereich der KI sind Sprachmodelle kritische Komponenten vieler Anwendungen, einschließlich Chatbots, persönlicher Assistenten und Empfehlungssystemen.
- In der Gesellschaft haben CL-Anwendungen das Potenzial, die Art und Weise zu revolutionieren, wie wir miteinander und mit Maschinen kommunizieren, indem sie es einfacher machen, für Menschen auf Informationen zuzugreifen und über sprachliche und kulturelle Barrieren hinweg zu kommunizieren.


Sparks of Artificial General Intelligence:
Early experiments with GPT-4

Sébastien Bubeck Varun Chandrasekaran Ronen Eldan Johannes Gehrke
Eric Horvitz Ece Kamar Peter Lee Yin Tat Lee Yuanzhi Li Scott Lundberg
Harsha Nori Hamid Palangi Marco Tulio Ribeiro Yi Zhang

Microsoft Research

https://arxiv.org/abs/2303.12712 (2023)

The Big Picture

Computerlinguistische Anwendungen

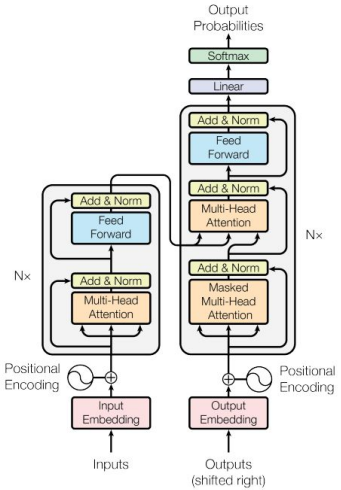
Wie kommen wir von:

N-gram Modellen

```
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
```

Zu:

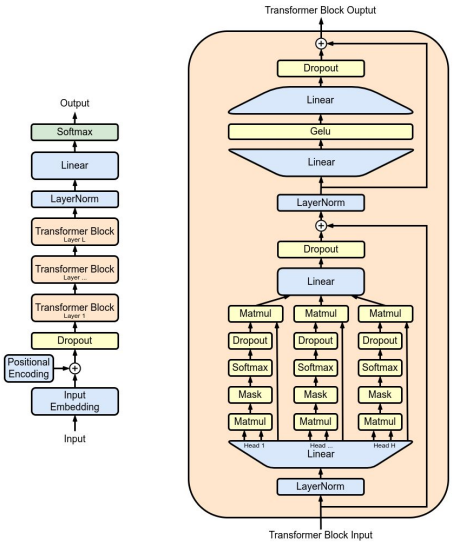
Neuronalen Netz Modellen



Attention is All you Need | Advances in NEURIPS, 2017) Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

Zu:

Transformer LLMs

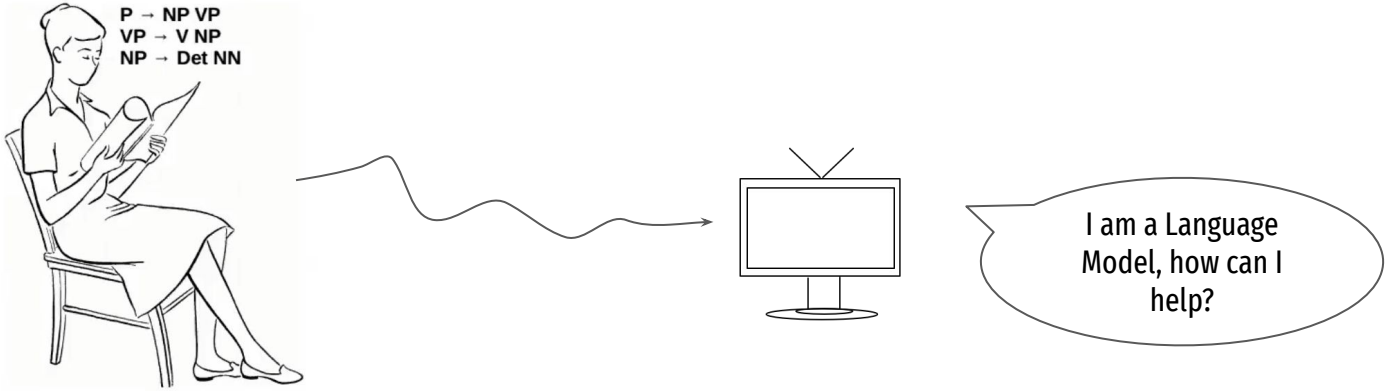


https://en.wikipedia.org/wiki/Generative_pre-trained_transformer

Maschinelles Lernen

Computerlinguistische Anwendungen

- Warum?
- Vor- und Nachteile gegenüber Alternativen?
- Genauigkeit; Abdeckung, erforderliche Resourcen (Daten, Expertise, Arbeitsaufwand)



Maschinelles Lernen

Alternativen; Vor- und Nachteile

Maschinelles Lernen (Neuronale Netze)	Regelbasierter Ansatz (Symbolische AI)
• Daten • Abdeckung • Annotationen • Statistische Kenntnisse	• Regeln • Genauigkeit • Linguistisches Wissen

... aber was genau ist ML?

Maschinelles Lernen

Eine Definition

"A computer program is said to learn from **experience** *E* with respect to some class of **tasks** *T* and **performance** measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*." (Mitchell, 1997)

Maschinelles Lernen

Eine Definition

"A computer program is said to learn from **experience** *E* with respect to some class of **tasks** *T* and **performance** measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*." (Mitchell, 1997)

- Lernen: Die Fähigkeit erlangen, eine Aufgabe auszuführen
- Aufgaben werden als Menge von **Instanzen** (Beispielen) repräsentiert ("experience")
- Instanzen werden durch **Merkmale** (Features) repräsentiert: Mengen numerischer Eigenschaften, die als Vektoren $\mathbf{x} \in \mathbf{R}^n$ dargestellt werden können.

Daten

“A computer program is said to learn from **experience** *E* [...], if its performance [...] improves with **experience** *E*.”

- Datensatz: Sammlung von Instanzen
- Design-Matrix

$$\mathbf{X} \in \mathbf{R}^{n \times m}$$

  - ➢ n: Anzahl von Instanzen
  - ➢ m: Anzahl von Merkmalen
  - ➢ Beispiel: $X_{i,j}$ Wert des Merkmals j (z.B. Häufigkeit eines Wortes) in Dokument *i*

Bei überwachten (supervised) Lernverfahren, z.B. Klassifizierung, gibt es für jede Instanz noch ein **Label**

- ➢ Label: vorauszusagender Wert/Kategorie
- ➢ Trainingsdaten: Labelvektor $\mathbf{y} \in \mathbf{R}^{n}$ zusätzlich zu **X**

Daten

“A computer program is said to learn from **experience** *E* [...], if its performance [...] improves with **experience** *E*.”

- Datensatz: Sammlung von Instanzen
- Design-Matrix

$$\mathbf{X} \in \mathbf{R}^{n \times m}$$

- ➢ n: Anzahl von Instanzen
- ➢ m: Anzahl von Merkmalen
- ➢ Beispiel: $X_{i,j}$ Wert des Merkmals j (z.B. Häufigkeit eines Wortes) in Dokument *i*

Design-Matrix

	0	1	2	...	**m**
0					
1					
2					
...					
n					

Daten

“A computer program is said to learn from **experience** *E* [...], if its performance [...] improves with **experience** *E*.”

- Datensatz: Sammlung von Instanzen
- Design-Matrix

$$\mathbf{X} \in \mathbf{R}^{n \times m}$$

- ➢ n: Anzahl von Instanzen
- ➢ m: Anzahl von Merkmalen
- ➢ Beispiel: $X_{i,j}$ Wert des Merkmals j (z.B. Häufigkeit eines Wortes) in Dokument *i*

Design-Matrix

	0 *the*	1 *quick*	2 *brown*	... *...*	m
0	12	2	3		
1	45	1	0		
2	30	2	8		
...					
n					

Dokument Nummer i bis n

Anzahl des Wortes j bis m

Datensatz

Beispiel: Spam-Filter

Email Betreff	Label (No spam?)
our meeting tomorrow	True
congratulations you won 100.000.000 €	False
which flight did you prefer?	True
cheap pills	False
registration confirmation	True
investment opportun1ty	False
hi i am single	False
10% off this month X	True Y

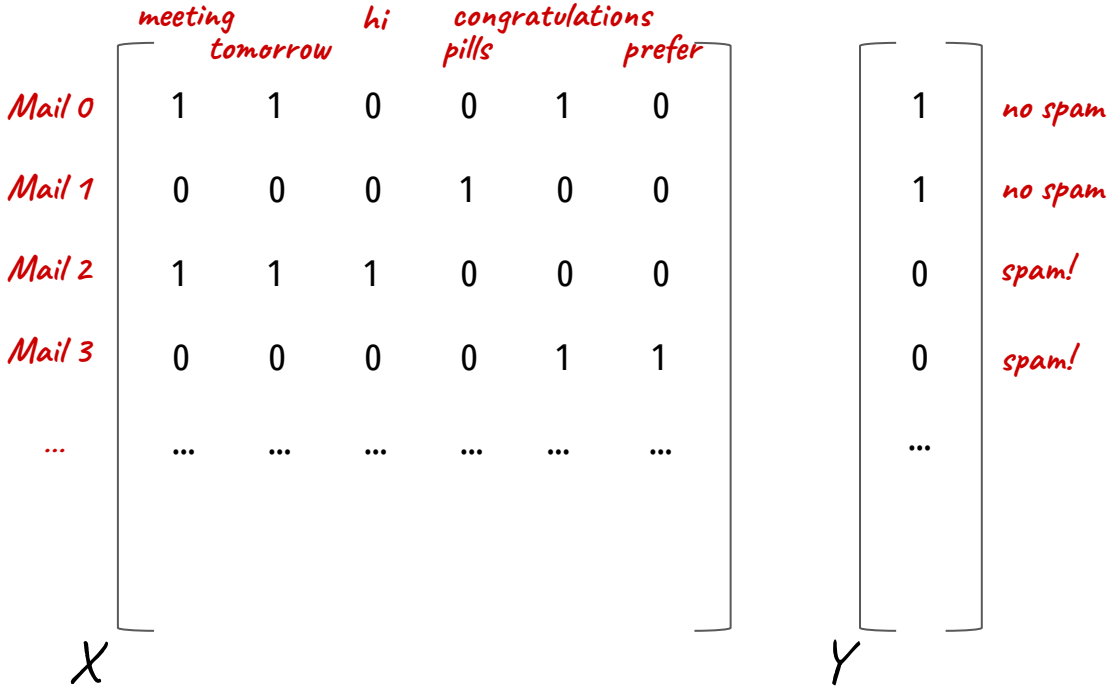
Datensatz

Beispiel: Spam-Filter

- Durch welche Merkmale könnte jede Instanz (Email) dargestellt werden?
  - Unigramme (einzelne Worte), Bigramme (Wort-Paare), Absender-Domain, ...
- Eine Instanz besteht aus
  - Merkmalen mit Merkmalswerten
  - Label
- Ein Datensatz besteht aus einer Menge von Instanzen
- Äquivalent: Ein Datensatz besteht aus Design-Matrix und Label-Vektor

Datensatz

Design-Matrix, Label-Vektor



Datensatz

Design-Matrix, Label-Vektor

- Ein Label-Vektor **y** enthält für jede Instanz im Datensatz einen Integer-Wert, der das Label codiert
  - Bei einer *binären Klassifikation* 1 oder 0 (je nachdem, ob das Label True oder False ist)
  - Bei *Multiklassen-Klassifikation,* enthält **y** bei *k* Klassen Werte aus dem Bereich 1 … k

  Hinweis: in der mathematischen Beschreibung eines Modells ist der Bereich 1 … k, in der praktischen Implementierung 0 … k - 1

- Dieselbe Information kann auch in einer Label-Matrix $\mathbf{Y} \in \mathbf{R}^{n \times k}$ codiert werden.
  - Für die i-te Instanz mit dazugehörigem Label $\mathbf{y}_i = j$ hat die Label-Matrix den Eintrag $\mathbf{Y}_{ij} = 1$, für alle $I \neq j$ gilt: $\mathbf{Y}_{il} = 0$
  - Diese Darstellung nennt man auch *1-hot* Encoding

Datensatz

Beispiel: Spam-Filter

Email Betreff	Kategorie
our meeting tomorrow	Primary
congratulations you won 100.000.000 €	Spam
which flight did you prefer?	Primary
cheap pills	Spam
registration confirmation	Social
investment opportun1ty	Spam
hi i am single	Spam
10% off this month X	Promotions Y

Datensatz

Beispiel: Spam-Filter

Email Betreff	Kategorie	0	1	2	3
our meeting tomorrow	Primary 0	1	0	0	0
congratulations you won 100.000.000 €	Spam 1	0	1	0	0
which flight did you prefer?	Primary	1	0	0	0
cheap pills	Spam	0	1	0	0
registration confirmation	Social 2	0	0	1	0
investment opportun1ty	Spam	0	1	0	0
hi i am single	Spam	0	1	0	0
10% off this month X	Promotions 3	0	0	0	1 Y

Aufgaben/Problemklassen

“A computer program is said to learn [...] with respect to some class of **tasks *T*** [...] if its performance at **tasks in *T***, [...] improves [...] ”

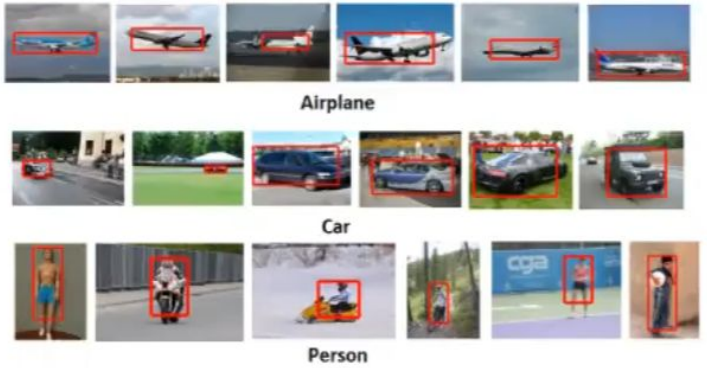
- **Klassifizierung**
- Regression
- Clustering
- ...

Aufgabe: Klassifizierung

- Zu welcher von *k* Kategorien gehört eine Instanz?

$$f: \mathbf{R}^{m} \rightarrow \{1 \dots k\}$$

- Beispiel: Kategorisierung von Bildausschnitten
  - Merkmalsvektor: Farbteile für jedes Pixel; Davon abgeleitete Merkmale
  - Vordefinierte Menge von Ausgabenkategorien



- Beispiel: Erkennung von Spam Emails
  - Merkmalsvektor: Hochdimensionaler Vektor mit wenigen Einträgen != 0 (sparse). Jede Dimension zeigt das Vorkommen eines bestimmten Wortes an.
  - Ausgabenkategorien: "*Good Email*" vs. "*Spam Email*"

Klassifizierung: Wichtige Konzepte

- **Lineares Modell:**

  Jeder Merkmalswert wird mit einem Gewicht multipliziert, die Summe der Produkte ergibt die Vorhersage für eine Klasse.
- **Fehlerfunktion:**

  Misst, wie gut die echte Labels vorhergesagt werden.
- **Test- und Trainingsdaten**:

  Auf den Großteil 70-80% der Daten wird trainiert, dabei werden 20-30% der Daten zum Test zurückgehalten. Beide Daten sollten die gleiche Wahrscheinlichkeitsverteilung aufweisen.
- **Overfitting:**

  Wenn auf den Trainingsdaten ein kleiner Fehler, aber auf den Test-Daten ein großer Fehler gemessen wird.
- **Regularisierung:**

  Verhindern von Overfitting während des Trainings, durch Einschränkung des Modells. Zum Beispiel, indem einzelne Merkmale nicht beliebig große Modell-Gewichte erhalten können.

Performanz-Maße (Fehlerfunktionen)

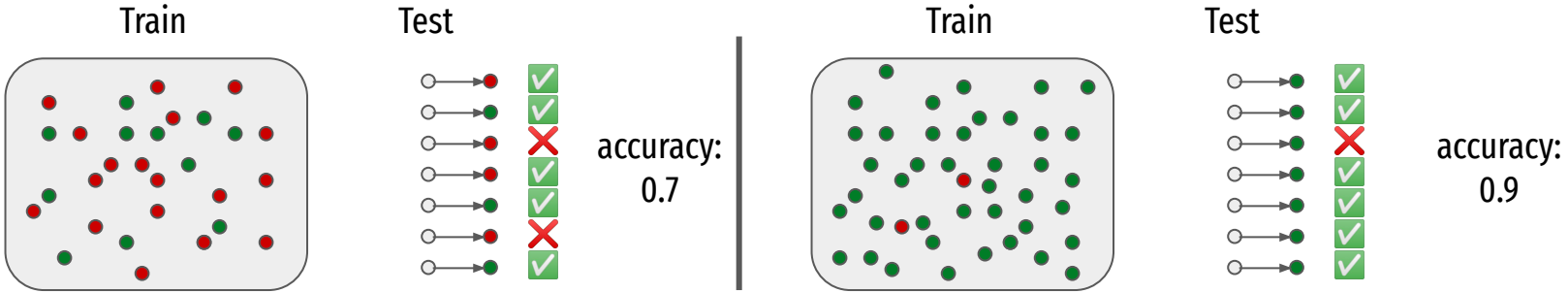
"A computer program is said to learn [...] with respect to some [...] **performance measure** P, if its performance [...] **as measured by** P, improves [...]"

- Ein Performanz-Maß ermöglicht die Vorhersagequalität eines Algorithmus quantitativ festzustellen.
- Welches Maß verwendet werden kann, hängt von der Art der Aufgabe ab:
  - **Klassifizierung:** Accuracy, F1-Score
  - **Ranking:** Mean Average Precision, Spearman's Rank Correlation
  - **Regression:** Mean Squared Error
  - **Textüberlappung (maschinelle Übersetzung):** BLEU, ...
  - **Wahrscheinlichkeitsmodell:** Wahrscheinlichkeit der Daten (Likelihood) kann als Maß verwendet werden

Fehlerfunktionen für Klassifikation

Accuracy

- Anteil der Instanzen, für die der Klassifikator die korrekte Kategorie vorhersagt
- error rate = 1 - accuracy (1: 100%)
- Wenn ein großes Ungleichgewicht in der Verteilung der Klassen besteht, ist Accuracy kein gutes Maß. Beispiel:
  - In einem Datensatz sind 99% der E-mails GUT (kein Spam), und 1% SPAM.

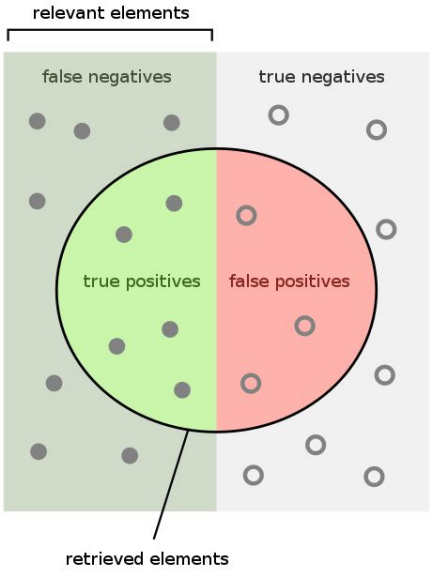


wird nahezu immer “kein Spam” sagen und hat hohe accuracy

F-measure (F-score)

Wird berechnet aus **Precision** und **Recall**

F1-score	=	2 * Precision * Recall	/	Precision + Recall
Precision	=	|relevant ∩ retrieved|	/	|retrieved|
Recall	=	|relevant ∩ retrieved|	/	|relevant|



Der **F1-Score** ist ein Maß für die Genauigkeit eines Modells, das die Präzision und den Recall kombiniert. Ein perfekter F1-Score ist 1 und ein schlechter F1-Score liegt nahe bei 0.

Die **Präzision** misst, wie viele der vom Modell als positiv identifizierten Instanzen tatsächlich positiv sind. Es gibt an, wie genau das Modell in der Identifikation von positiven Instanzen ist.

Der **Recall** gibt hingegen an, wie viele der tatsächlich positiven Instanzen vom Modell korrekt identifiziert wurden. Er gibt an, wie vollständig das Modell positive Instanzen erfasst hat.

https://en.wikipedia.org/wiki/Precision_and_recall

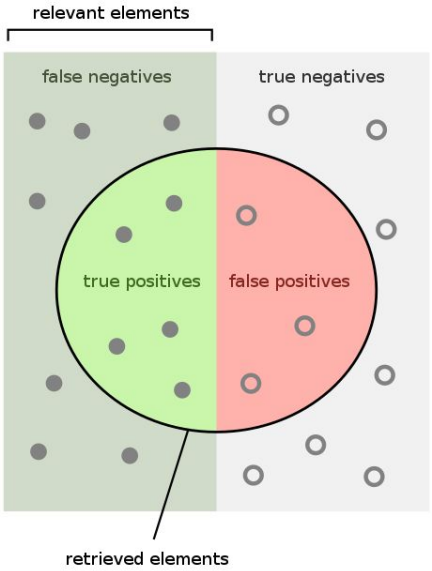
F-measure (F-score)

Wird berechnet aus **Precision** und **Recall**

F1-score	=	2 * Precision * Recall	/	Precision + Recall
Precision	=	|relevant ∩ retrieved|	/	|retrieved|
Recall	=	|relevant ∩ retrieved|	/	|relevant|

- “relevant” : Menge der Instanzen, die das relevante Label haben
- “retrieved”: Menge der Instanzen für die das relevante Label vorhergesagt wurde

→ Für das F-Measure muss immer ausgewählt werden, welche die relevante Kategorie ist.



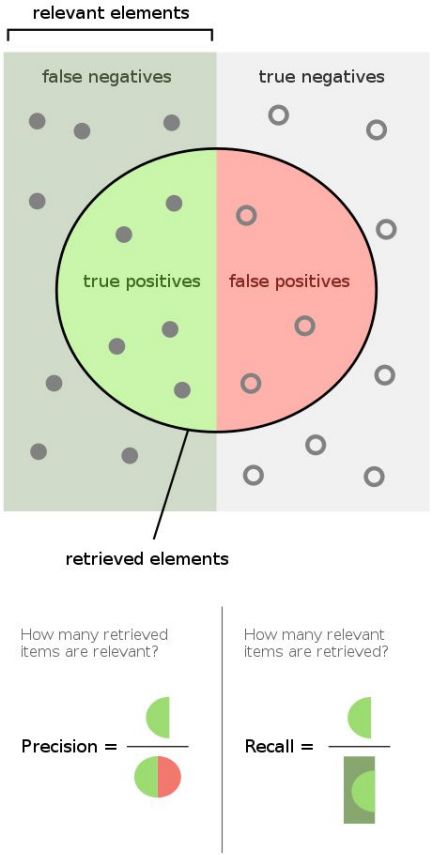
https://en.wikipedia.org/wiki/Precision_and_recall

F-measure (F-score)

Beispiel:

Wir benutzen eine Suchmaschine, welche uns 30 Seiten zurückliefert, von denen nur 20 relevant sind, während wir 40 weitere relevante Seiten nicht erhalten.

Berechnen wir die Precision, Recall und den F-Score!



https://en.wikipedia.org/wiki/Precision_and_recall

F-measure (F-score)

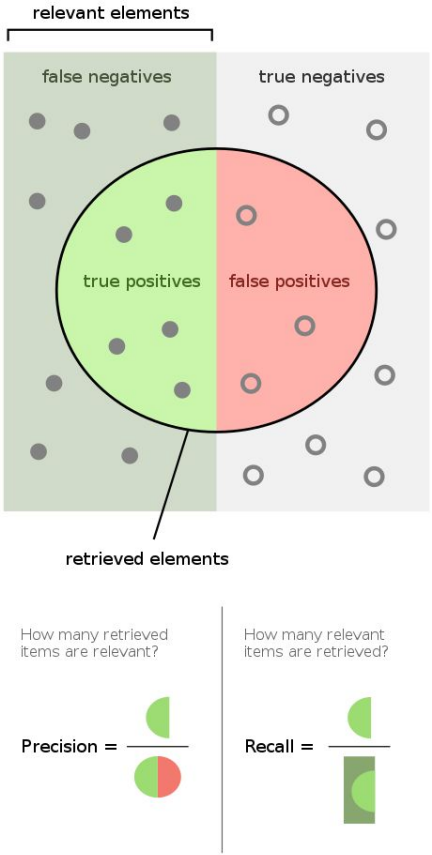
Beispiel:

Die **Präzision** des Modells wäre in diesem Fall 20/30 oder 0,67. Dies bedeutet, dass das Modell 67% der als positiv identifizierten Instanzen korrekt klassifiziert hat.

Der **Recall** des Modells wäre 20/60 oder 0,33. Dies bedeutet, dass das Modell 33% der tatsächlich positiven Instanzen korrekt erfasst hat.

Der **F1-Score** des Modells wäre also: F1 = 2 * (precision * recall) / (precision + recall)

F1 = 2 * (2/3 * 1/3) / (2/3 + 1/3)
= 2 * (2/9) / (1)
= 4/9
≈ 0.444



https://en.wikipedia.org/wiki/Precision_and_recall

Auswahl der Daten

Daten in Trainings-, Test- und Entwicklungsdaten (train, test, dev) aufteilen. Meist in etwa: 60% : 20% : 20%

Email Betreff	Label	
our meeting tomorrow	0	train set
congratulations you won 100.000.000 €	1	
which flight did you prefer?	0	
cheap pills	1	
registration confirmation	0	
investment opportun1ty	1	
hi i am single	1	dev set
10% off this month	1	
best offer I can make	0	test set
meeting in the meeting room	0	

Trainings- / Development- / Test-Fehler

- Merkmalsgewichte werden auf Trainingsdaten automatisch optimiert
- Das Modell (Architektur, Datenrepräsentation, Hyperparameter) wird anhand der Development-Daten ausgewählt
- Die zu erwartende Performanz des Models wird anhand der Testdaten ermittelt
- Ergebnisse auf Trainings- oder Entwicklungsdaten können nicht als Bewertung des Algorithmus aufgefasst werden!