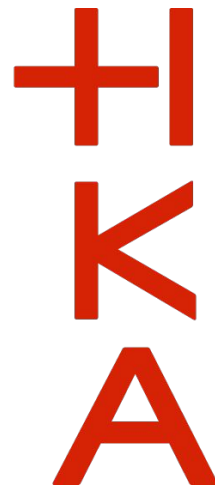




KI Labor - Sommersemester 2022

NLP Einführung



Robin Baumann, Sven Müller, **Maximilian Blanck**,  
**Pascal Fecht**, Tim Bossenmaier, Matthias Richter

Karlsruhe, 22. April 2022

# Schedule

Datum	Thema	Inhalt	Präsenz
18.03.22	Allg.	Organisation, Teamfindung, Vorstellung CV	Ja
25.03.22	CV	Q&A Sessions	Nein
01.04.22	CV	Sprintwechsel, Vorstellung Assignment	Ja
08.04.22	CV	Q&A Sessions	Nein
15.04.22	Ostern		
22.04.22	CV / NLP	Abgabe CV, Vorstellung NLP	Ja
29.04.22	NLP	Q&A Sessions	Nein
06.05.22	NLP	Sprintwechsel, Vorstellung Assignment	Ja
13.05.22	NLP	Q&A Sessions	Nein
20.05.22	NLP / RL	Abgabe NLP, Vorstellung RL	Ja
27.05.22	RL	Sprintwechsel, Vorstellung Assignment	Nein
03.06.22	Ausfall (Sommerplenum)		
10.06.22	Pfingsten (H-KA zu)		
17.06.22	RL	Q&A Sessions (Brückentag)	Nein
24.06.22	RL	Abgabe RL, Abschluss KI Labor	Ja
01.07.22		Puffer	

# Agenda

## › **Theorie**

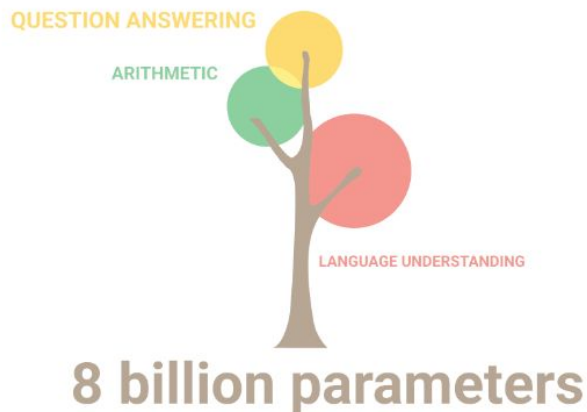
- Natural Language Processing
- Vokabular & Features
- Word Embeddings (word2vec)
- Text Klassifikation → Sentiment Analyse


## › **Übungsaufgaben**

- Word Embeddings Alice im Wunderland (Aufgabe 1)
- Sentiment Analyse für Twitter Posts (Aufgabe 2)

# Recent **News** and **Highlights** in NLP

## Pathways Language Model (PaLM): Scaling to 540 Billion Parameters for Breakthrough Performance





# DALL·E 2

DALL·E 2 is a new AI system that can create realistic images and art from a description in natural language.

# Natural **L**anguage **P**rocessing

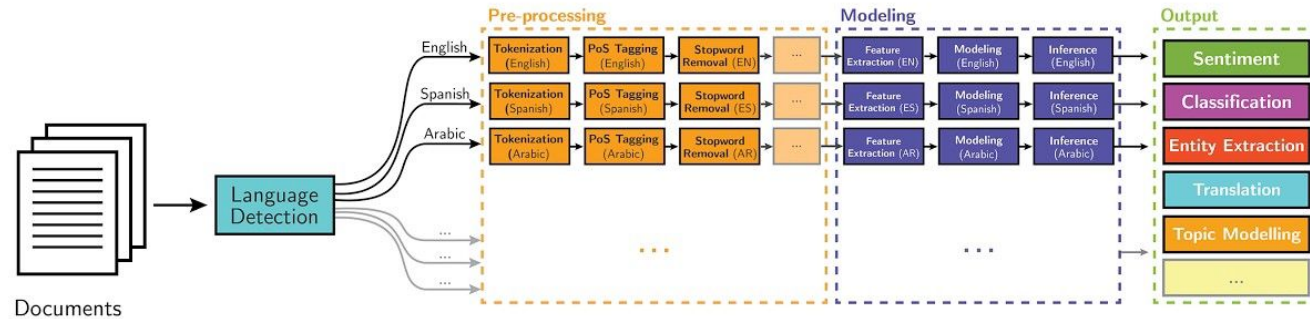
# Natural Language Processing...

- › **beinhaltet** Linguistik, Natürliche Sprache, Künstliche Intelligenz, Mathematik, ...
- › **befasst** sich mit der Interaktion zwischen Computern und Menschen
- › **beschäftigt** sich unter anderem mit:  
text classification, named entity recognition, machine translation, part of speech tagging, sentiment analysis, question answering, text summarization, text generation, speech recognition, speech to text, text to speech, chatbots, virtual assistants, ...

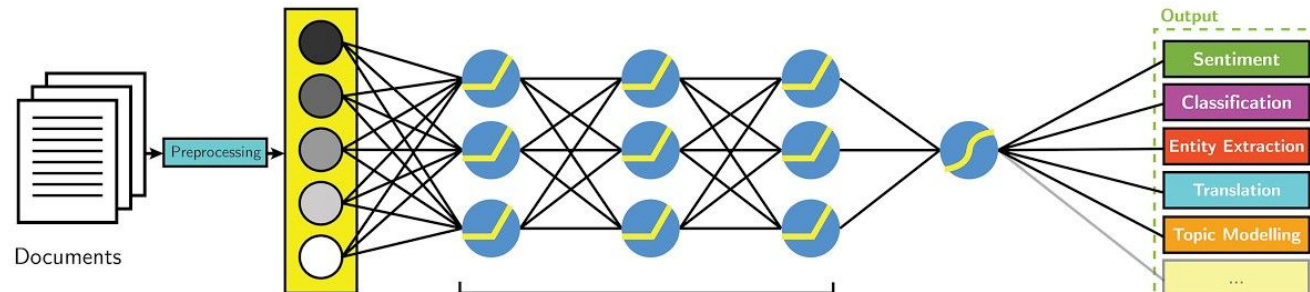
→ <https://paperswithcode.com/area/natural-language-processing>



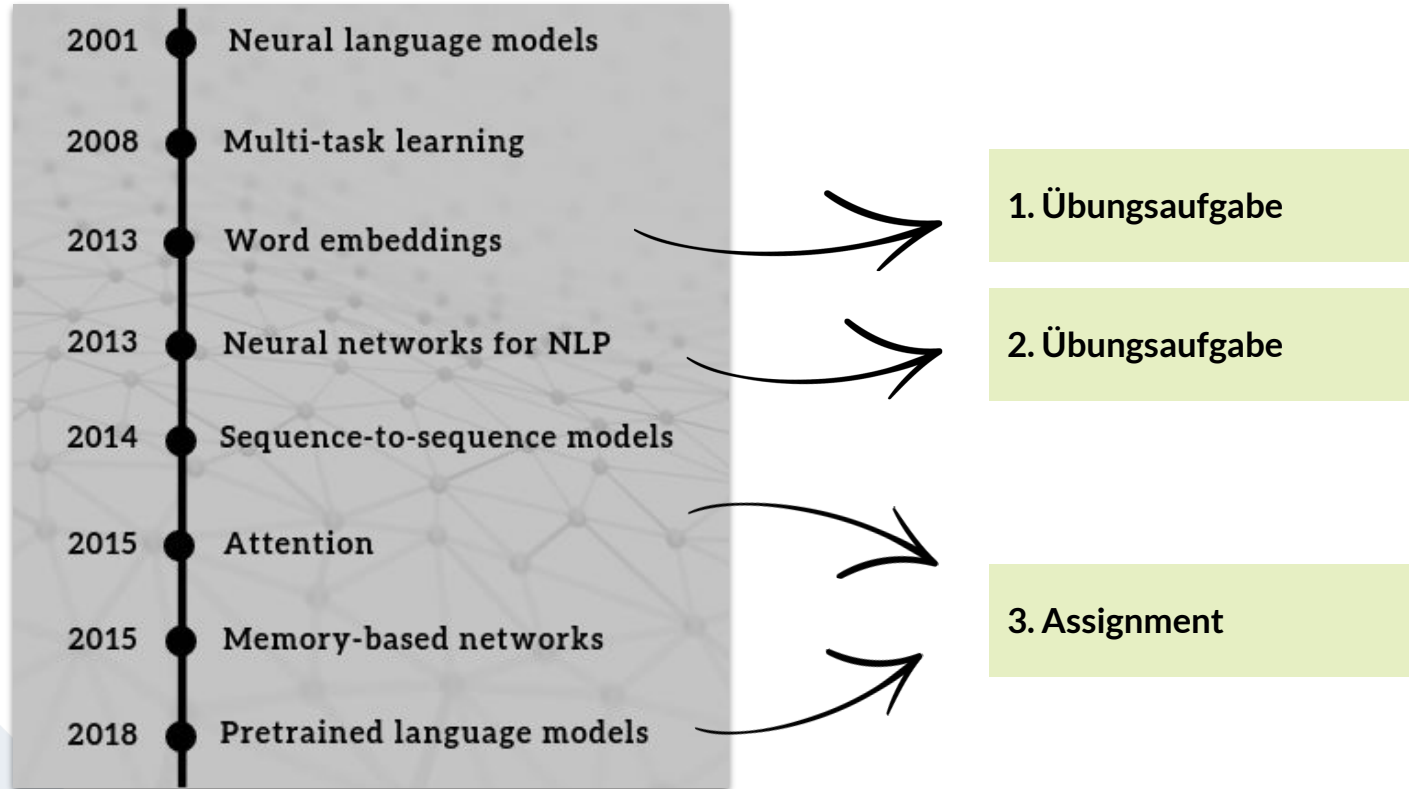
# Klassische vs Deep Learning NLP Pipeline



## Deep Learning-based NLP



# Die letzten 20+ Jahre in NLP



# Herausforderungen in NLP

1. Sprache muss irgendwie in den Computer kommen.
  - › Wort / Laut  $\Rightarrow$  Zahl(en) (Featurisierung)
2. Das Vokabular einer Sprache ist in der Regel sehr umfangreich
  - › Dimensionalitätsproblem (?)
3. Einige Wörter sind enger verbunden als andere
  - › Ähnlichkeit, Abstand  $\Rightarrow$  Maße?

# Verarbeitung von Textdaten

## **Vokabular**

# Vokabular

## Typische Herausforderungen

- › Wie viele unterschiedliche Wörter soll mein Modell als Eingabe erhalten?
- › Nicht alle Wörter können im Vokabular abgebildet werden.
- › Sehr seltene Wörter gehen unter.
- › Wörter, die häufig vorkommen aber wenig Inhalt enthalten (“der”, “ein”, ...), können starke Features werden.
- › Wie gehen wir mit Noise (falscher Rechtschreibung) um?
- › Was passiert, wenn wir zur Inferenz Wörter erhalten, die nicht Teil des Vokabulars sind?

# Vokabular

## Tokenisierung

“Wörter” → einfacher Ansatz: Trenne an “ ” (Leerzeichen):

```
re.compile(r'\W*\s+').split("Das Auto fährt") → ["Das", "Auto", "fährt"]
```

**Aber**, was ist mit:

- › “H-Milch”, “Auto-\nbahn”, “Dr. Musterdokter”
- › Sätze → Trenne an “.” Punkt?
- › Zahlen? Daten? Uhrzeiten? ...

# Vokabular/Features

## Tagging

**Beispiel:** Jede IBAN ist ein eigenes Token

⇒ Kein relevantes Feature für Modell

```
1 if re.match('^DE(?:\s*[0-9a-zA-Z]\s*){20}$', TOKEN):  
2     return "<IBAN>"
```

⇒ Modell erkennt alle (deutschen) IBANs als ein einziges Feature.

# Vokabular

- › Stoppwörter entfernen

```
1 from nltk.corpus import stopwords
2 stop_words = set(stopwords.words('english'))
3 tokens = [w for w in tokens if not w in stop_words]
```

- › Stemming / Lemmatisierung: Hauses, Häuser, Hauses, ... ⇒ Haus
- › Und viele weitere ...





# Vokabular

Sind Wörter die richtige Granularität für Tokens?

## **Sub**words

- › Nicht im Vokabular enthaltene Token werden in Sub-Wörter geteilt.

## **Char**acters

- › Jedes Char hat ein Encoding
- › Nachteil: Die Bedeutung von Wörtern geht verloren.

## **Byte-Pair** Encodings → [Erklärung](#)

- › Mix aus Subword und Character Level Encoding

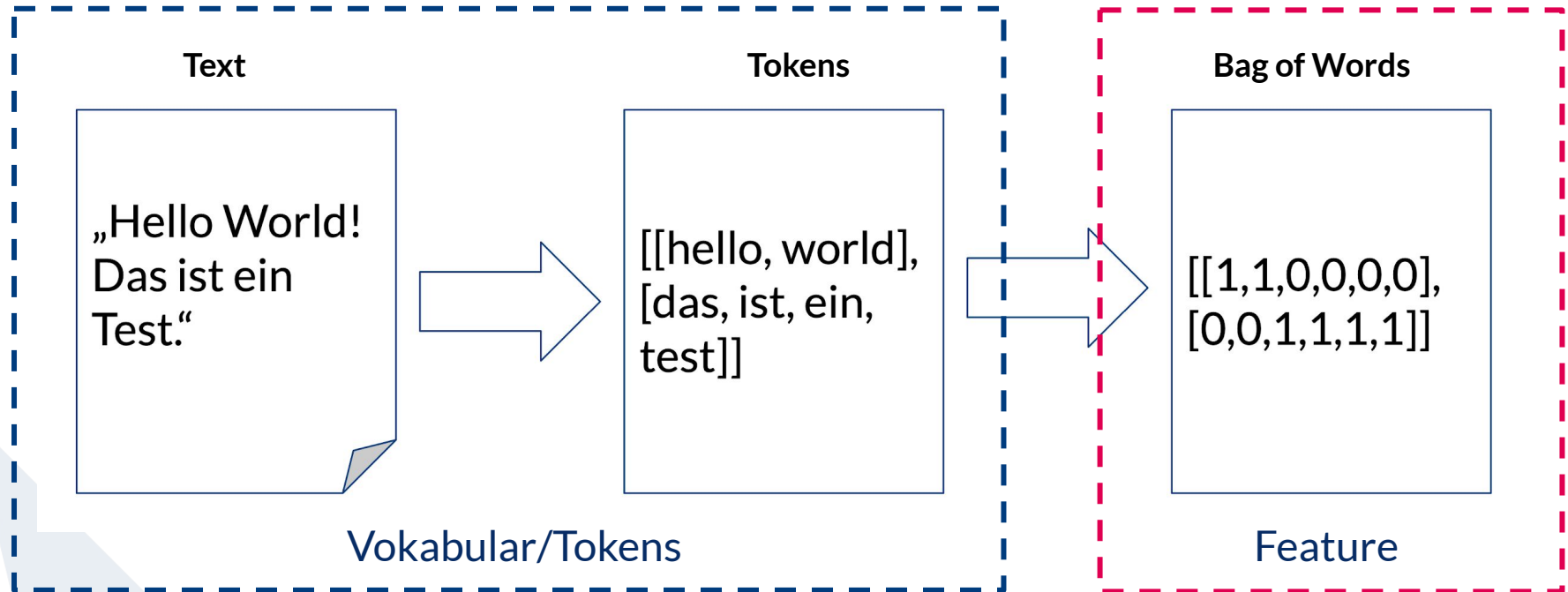
⇒ Wir fangen mit Wörtern, kommen aber darauf im Assignment zurück.

# Verarbeitung von Textdaten

## **Features**

# Wie kommt die Sprache in den Computer?

Klassischer Ansatz: **Text** → **Tokens** → **Bag of Words**



# Features

## Bag of Words

**Zähle alle unterschiedlichen Wörter in den Texten, die betrachtet werden:**

- › Baue einen Vektor mit Vokabularlänge
- › Jedes Wort hat einen Index im Vektor
- › Merke die Anzahl für jedes Wort und Speichere die Zahl im Vektor
- › Feature für einen Text:
  - $[0,0,2,1,0,0,\dots,0,0,1,3,0,0]$

**→ Problem: Der Vektor beinhaltet sehr viele “0”en**

# Klassischer Ansatz: BoW und hand-crafted Features

## Optimierung von BoW mit Tf-idf:

- › Reduzierung des Vokabulars mit Heuristik, die wichtige Tokens identifiziert (bspw. auf 2000 Input Features)
- › Wörter werden durch ihr “vorkommen” (im Corpus) gewichtet

## Hand-Crafted Features, wie:

- › Statistiken (Satzlänge, Anzahl unterschiedlicher Wörter)
- › Integration von externen Datenquellen (z.B. Wie viele Wörter kommen in einem Schimpfwort Lexikon vor?)
- › Named Entities / POS Tags
  - Wird im Text über Personen, Organisationen, ... gesprochen?
  - Wie viele Verben, Adjektive, usw. kommen vor?

# Word2Vec

- › “You shall know a word by the company it keeps” (Firth, J. R. 1957:11)
- › Word2Vec-Paper: <https://arxiv.org/abs/1301.3781>

## Idee:

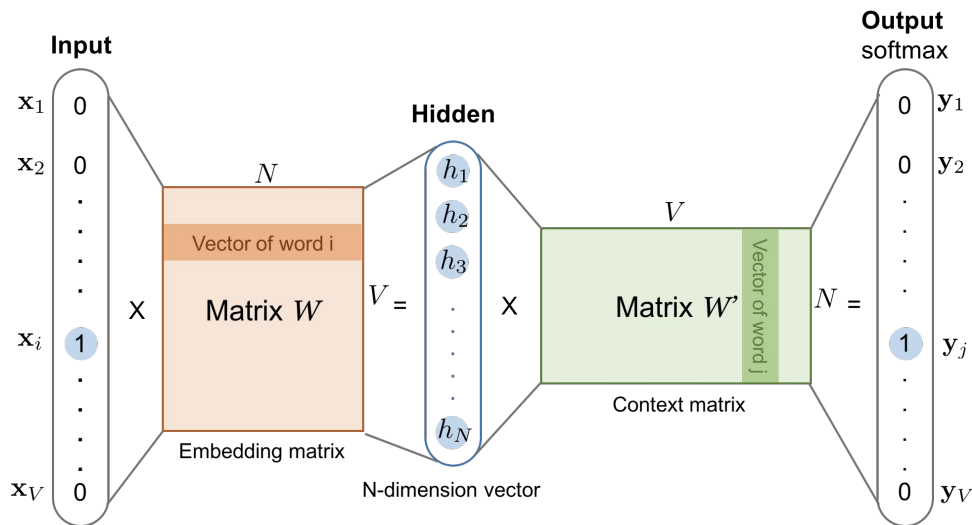
- › Die umliegenden Wörter eines Wortes (in einem beliebigen Text) haben Einfluss auf die Bedeutung dieses Wortes.
- › Beispielsatz: “*Die Ente **quakt** und tanzt.*”

# Word2Vec

- › Jedes Wort wird auf einen Vektor der Länge **n** abgebildet:
- › Bsp.  
“Auto” → [0.012, 0.981, -0.271,...]
- › **n** ist im Word2Vec-paper 300, andere Werte möglich
- › Wie werden diese Werte erzeugt?
  - 2 Ansätze:
    - Skip-Gram
    - Cbow

# Word2Vec - Skip-Gram (Kontext zum Wort)

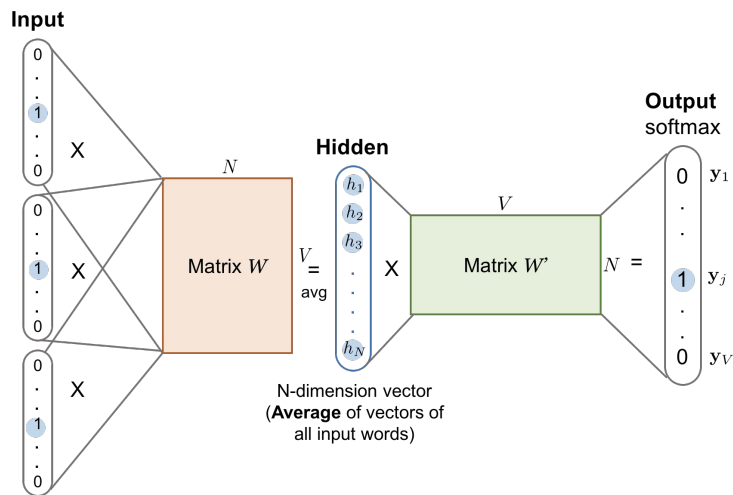
- › Gegeben das Wort “**quakt**”, wollen wir den Kontext vorhersagen
  - “Die Ente **quakt** und tanzt.” (Fenstergröße 5)
  - Kontext: [“Die”, “Ente”, “und”, “tanzt”]; Target: [“**quakt**”]
  - Trainingsdaten:  $X_1 = [\text{“quakt”}]$ ,  $Y_1 = [\text{“Die”}]$ ;  $X_2 = [\text{“quakt”}]$ ,  $Y_2 = [\text{“Ente”}]$ ; ...





# Word2Vec - Cbow (Wort zum Kontext)

- › Gegeben Kontext “Die”, “Ente”, “und”, “tanzt”, wollen wir “**quakt**” vorhersagen.
  - “Die Ente **quakt** und tanzt.” (Fenstergröße 5)
  - Kontext: [“Die”, “Ente”, “und”, “tanzt”]; Target: [“**quakt**”]
  - Trainingsdaten:  $X = [\text{“Die”, “Ente”, “und”, “tanzt”}]$ ,  $Y = [\text{“quakt”}]$



# Word Embeddings, Abstände, Gensim

- › Mit Word Embeddings lassen sich nun Abstände berechnen
  - Bsp.: “Ente” und “Gans” haben einen geringeren Abstand als “Ente” und “Auto”, da sie in *Texten* öfters mit den gleichen Kontext Wörtern vorkommen
- › Gensim\* ist ein Python package, das das Handling von Wortvektoren vereinfacht.
- › Beispiel:

```
from gensim.models import Word2Vec
model = Word2Vec.load("GoogleNews-vectors-negative300.bin")
model.similarity('germany', 'france')
```

# Klassifikation

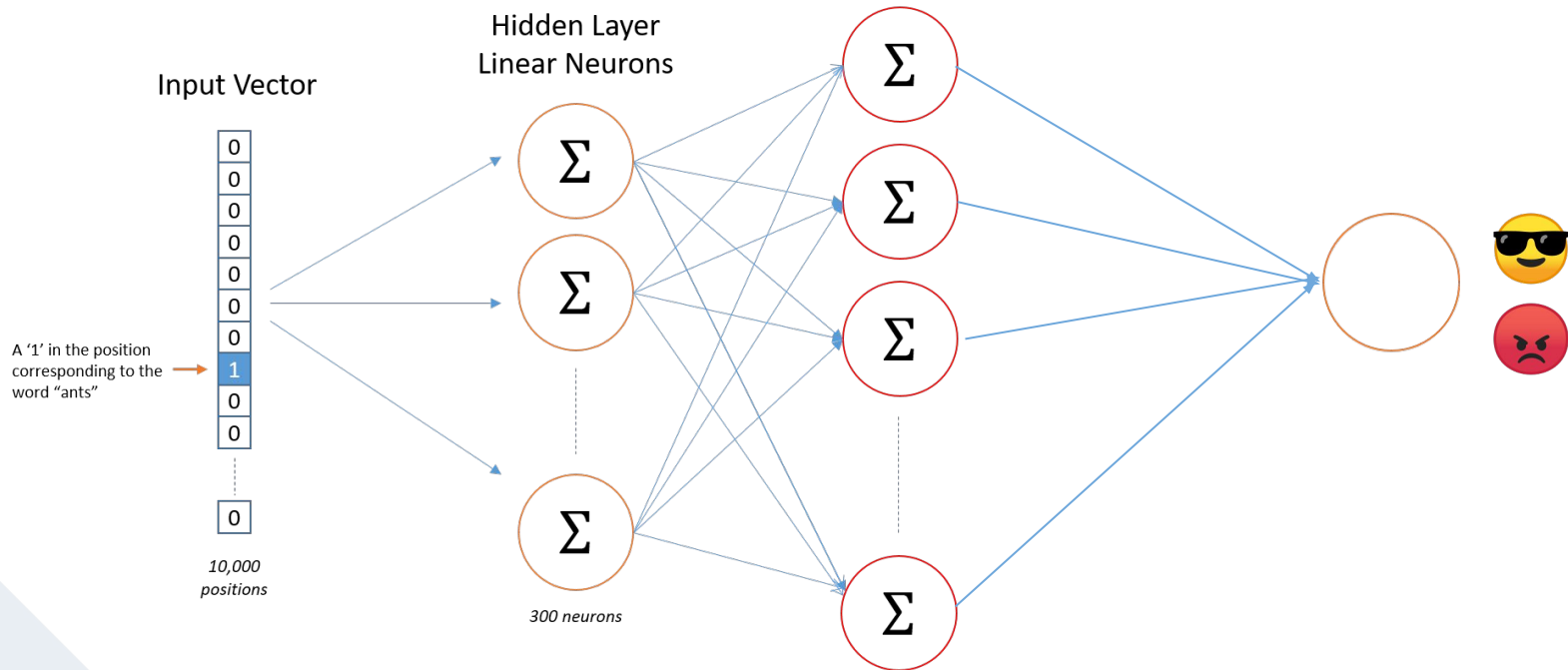
# **Sentiment Analyse**

# Sentiment Analyse

- › Sentiment Analyse = Stimmung eines Textes erkennen
- › Herausforderungen, wie:
  - “XY ist *blöd*” vs. “XY wäre *blöd* gewesen, diese Chance nicht zu nutzen.”
- › Häufige verwendet für Reviews (Produktreviews, Restaurantbesuche, ...)
- › In Übungsaufgabe 2:
  - Datensatz: Sentiment140 (16 Mio Tweets)
  - Target Label klassifiziert durch enthaltene Emojis
  - **Ziel:** Sentiment hinsichtlich Produkte, Marken etc. aus den Tweets herausfinden.

# Sentiment Analysis

## Binäre Klassifikation mit BoW



# colab

# Vielen Dank

inovex GmbH  
Ludwig-Erhard-Allee 6  
76131 Karlsruhe



# Verarbeitung eines Datensatzes

- › **Vokabular** (Liste unterschiedlicher Wörter im Datensatz):
  - › Einschränkung des Vokabulars
  - › Erkennung von Satz- / Wortgrenzen
  - › Normalisierung (z. B. Lowercasing)
  - › Stoppwörter entfernen (z.B. der, die, das, und, oder, an, in, um, ...)
- › Umwandlung von Text in **Features**:
  - › Part of Speech Tagging
  - › Bag Of Words, **Embeddings**, **Context Embeddings**
  - › Andere Features, z. B.
    - Rechtschreibung (Qualität der Sprache)
    - numerische Werte (Satzlänge, Anzahl unterschiedlicher Wörter)



# Google Colab

- › In GCP gehosteter Jupyter Notebook Service
- › Kollaborative Arbeit (ähnlich GDoc) möglich
- › Frei nutzbar (inklusive GPU/TPU!)
  - › → Ressourcen sind limitiert
  - › → GPU/TPU Runtime sparsam nutzen!
- › Notebooks und Daten liegen im Google Drive



#TFDevSummit

## Making the most of Colab



# Feedback



<https://forms.gle/zDjawTGbbs1Z8ryXA>