Textanalyse und semantische Suche

Mag. Dr. Gottfried Luef, Executive IT Architect, IBM Österreich

Was sie lernen werden

- Worin besteht die Analyse von natürlichsprachlichen Dokumenten
- Was ist zu tun, um effektiv darin zu suchen
- Was ist semantische Suche und welche Techniken werden benötigt
- Was kann Machine Learning dazu beitragen
- Was sind wichtige Python Libraries für diese Aufgaben, und wie geht man mit ihnen um

Inhalt

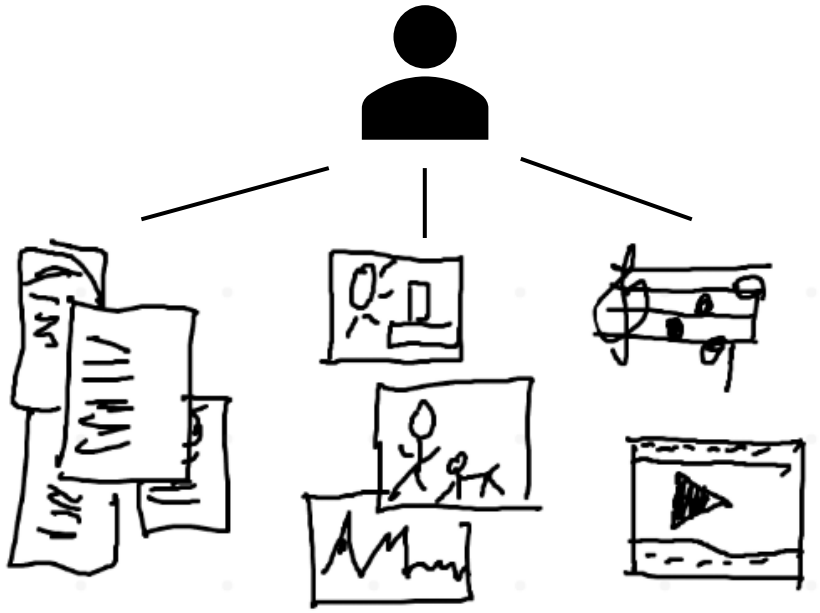
1. Information Retrieval
2. Suchmaschinen
3. Semantische Suche
4. NLP – Natural Language Processing
5. Einbettungen
6. Validierung
7. Anwendungen
8. Ausblick

1. Information Retrieval

1.1 Information Retrieval

1.2 Information Retrieval - Anwendungen

- **Information Retrieval**: Suche in unstrukturierten Daten



Web Suchmaschinen

Digitale Assistenten

Digitale Bibliotheken

Online Help

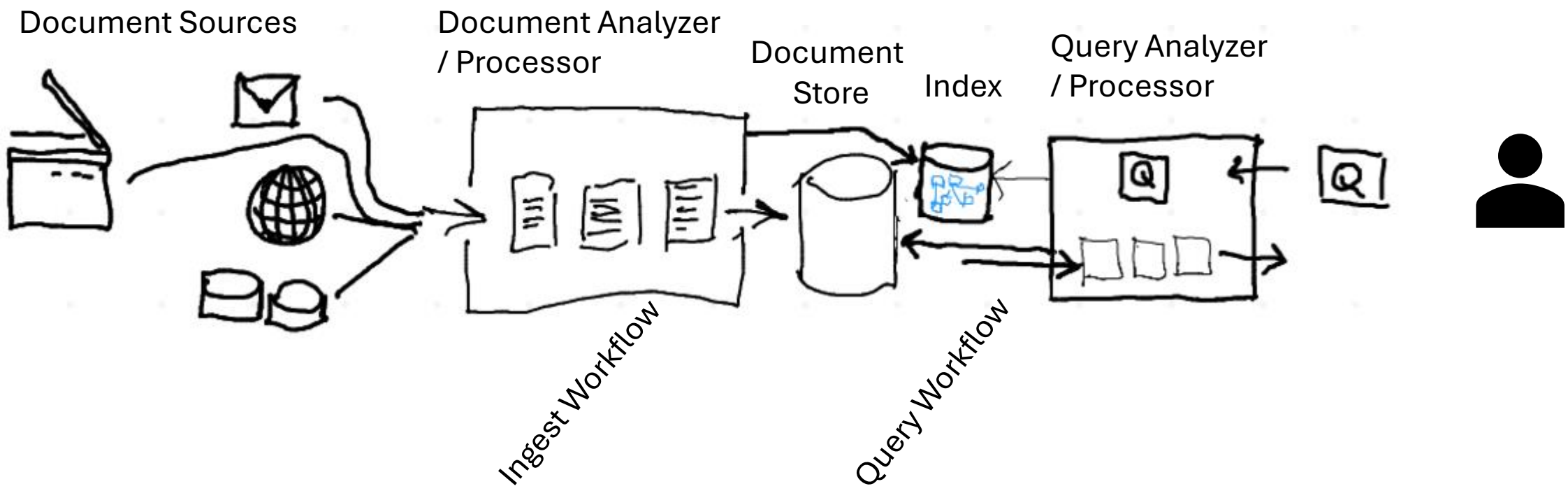
Enterprise Search

1.3 Information Retrieval Anwendungen: Begriffe

- *Information Retrieval*: Suche nach unstrukturierten Inhalten in verschiedenen Darstellungsformen
- *Web Suchmaschinen*: Suche im gesamten Internet, öffentlich
- *Digitale Assistenten*: Geschlossener Inhalt auf ein Thema bezogen, Frage-Antwort-Stil
- *Digitale Bibliotheken*: Geschlossener Inhalt als konsumierbare Bücher, Artikel, Musikstücke, …
- *Online Help*: Geschlossener Inhalt auf ein Thema bezogen, Volltextsuche, Suche nach Topics, Kontextbezogene Suche, …
- *Enterprise Search*: Inhalt sind Unternehmensdokumente, Suche nicht öffentlich zugänglich, Security wichtig, meist Volltextsuche und FAQ

2. Suchmaschinen

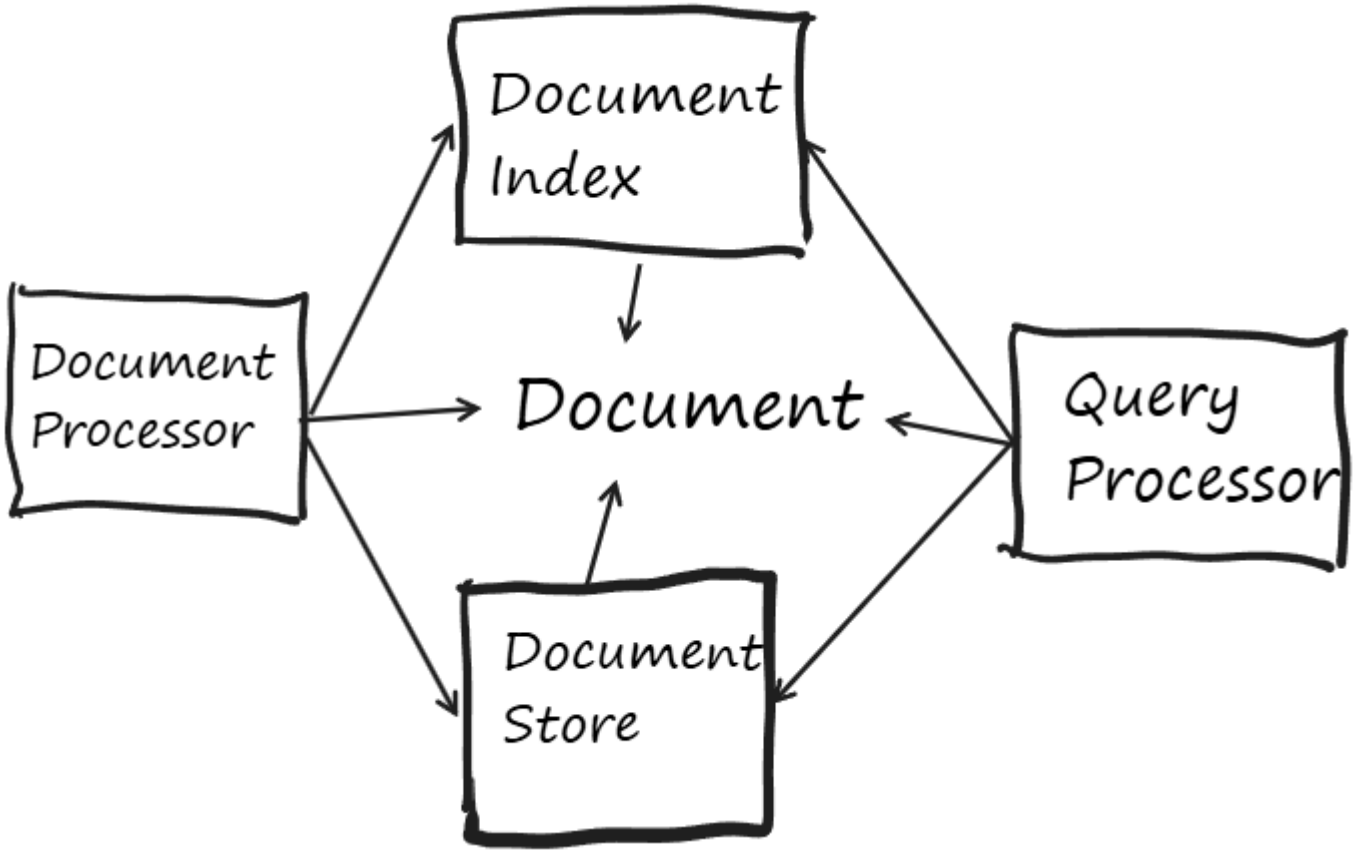
2.1 Architektur von Suchmaschinen



2.2 Architektur von Suchmaschinen: Begriffe

- *Document*: Unstrukturierter Text, maschinenlesbar
- *Document Source*: E-Mail, Internet, Laufwerke, Scan, …
- *Document Analyzer/Processor*: Vorbehandlung von Dokumenten
  - NLP (Natural Language Processing)
  - Feature Extraction
  - Indexierung
- *Index*: Datenstruktur, die die Auffindbarkeit von Dokumenten aufgrund von Abfragen (Queries) ermöglicht
- *Query Analyzer / Processor*: Vorbehandlung der Query
  - NLP, Feature Extraction
- Der *Ingest-Workflow* befördert die Dokumente in den Document Store über den Document Analyzer
- Der *Query-Worklow* befördert die Dokumente zum Benuzter über den Query Analyzer

2.3 Python Klassen



Document Processor
- liest Dokumente ein
- extrahiert Features
- fügt Features in den Index ein
- fügt Dokumente in den Doc Store ein

Document Store
- speichert Dokumente
- gewährt Zugriff per Doc-Id

Document Index
- bietet Suche von Dokumenten durch Angabe von Features

Query Processor
- nimmt Abfrage entgegen
- extrahiert Fetures aus Abfrage
- sucht im Index mit Features
- liest die erhaltenen Dokumente

3. Semantische Suche

3.1 „Semantische Suche“

Semantische Suche

Schlüsselwortsuche

- Extraktion von relevanten Begriffen
- Suche nach Worthäufigkeiten, statistischen Übereinstimmungen
- Invertierter Text-Index

Metadaten-Suche

- Entity Extraction
- Relationship Extraction
- Knowledge Graphen

Ähnlichkeits-Suche

- Extraktion und Suche von „Bedeutungen“, d.h. Nähe zu anderen Texten
- Kontextbezogen
- (Hybride) Vektorsuche
- Vektor-Index
- Chunking
- Embedding

Tokenisierung

Lemmatisierung

Stopword Removal

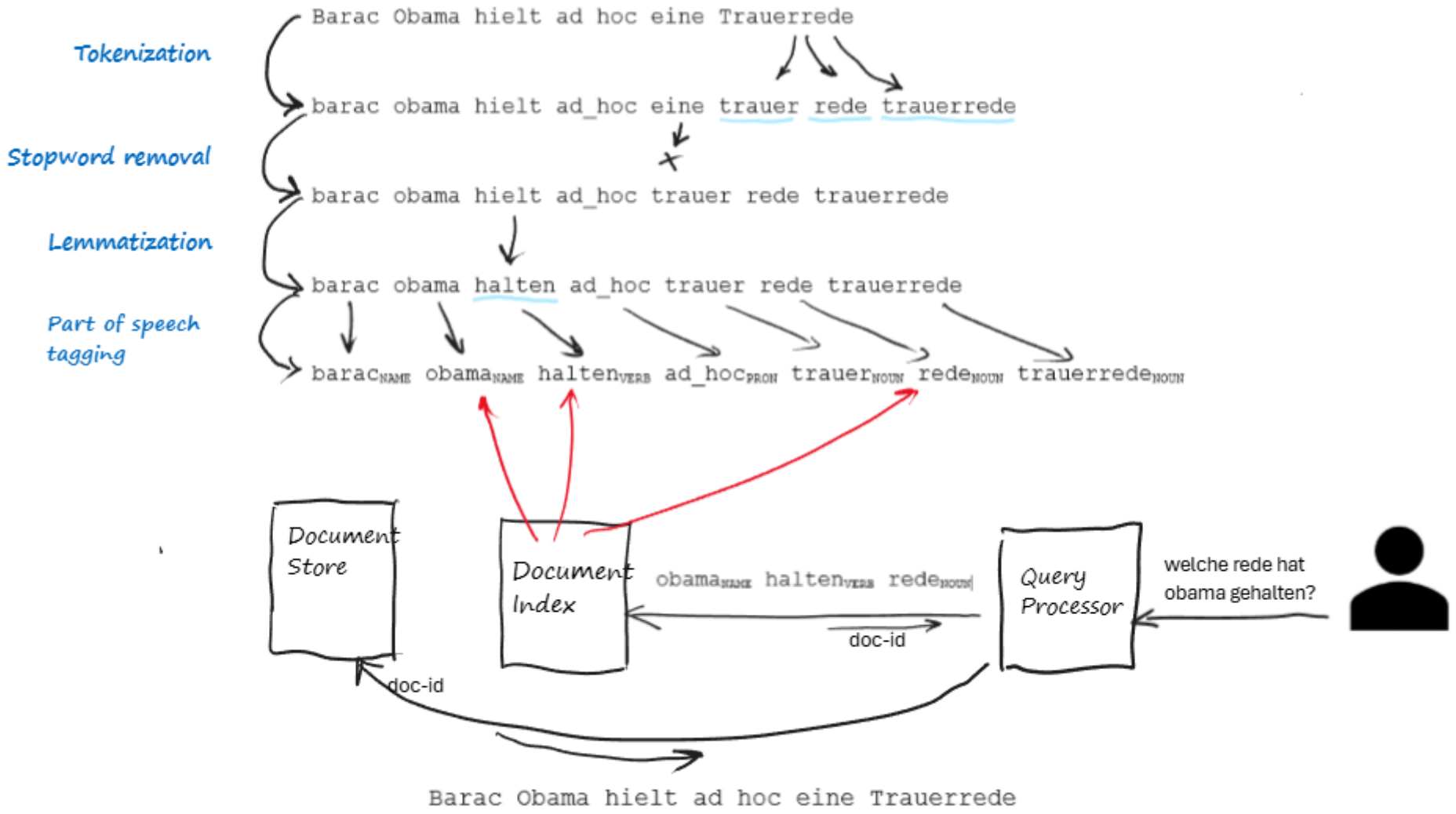
POS Tagging

3.2 Semantische Suche - Begriffe

- Die *Schlüsselwortsuche* ist das herkömmliche Suchverfahren. Es basiert auf versierte Techniken, Meta-Informationen aus den Dokumenten abzuleiten, sowie Wörter und Phrasen in den Dokumenten und Meta-Informationen zu finden.
- *Metadaten* sind wesentlicher Bestandteil im Information Retrieval. Sie werden aus den Dokumenten und ihrem Kontext gewonnen. (z.B. Erstellungsdatum, Entities (z.b. berühmte Personen))
- *Ähnlichkeitssuche* versucht, unabhängig vom Schlüsselwort-Matching möglichst solche Dokumente zu finden, die in einem größeren Zusammenhang mit der Abfrage übereinstimmen.
- *Semantische Suche* verwendet ‚herkömmliche' Techniken wie Entity Extraction, aber vor allem Ähnlichkeitssuche.

4. NLP (Natural Languare Processing)

4.1 NLP – Basic Pipeline

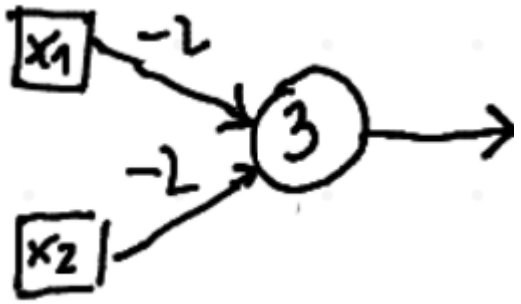


4.2 NLP: Begriffe

- *Tokenization*: Umwandlung eines Textes in eine Liste von indizierbaren Termen (aka Token). Dabei werden Wörter des Textes getrennt, ergänzt oder zusammengefasst.
- *Stopword Removal*: Wörter, nach denen nicht gesucht werden soll, werden weggelassen. Z.B. Artikel, Partikel, Personalpronomen, …
- *Lemmatization*: Zurückführung eines Wortes auf eine Stammform, z.b. bei Verben Nennform, bei Substantiva Nominativ im Singular.
- *Part-Of-Speech-Tagging*: Markierung (aka Tagging) eines Wortes als Verb, Substantiv, Adjektiv, ….

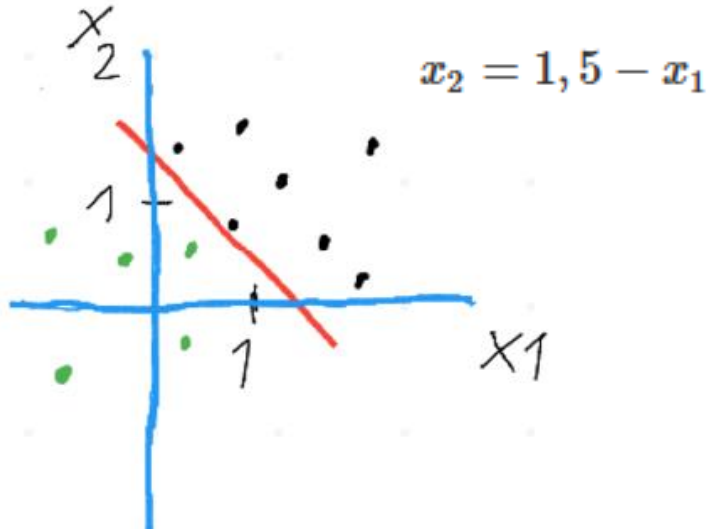
5. Einbettungen

5.1 Motivation Neuronale Netze: Das Perzeptron



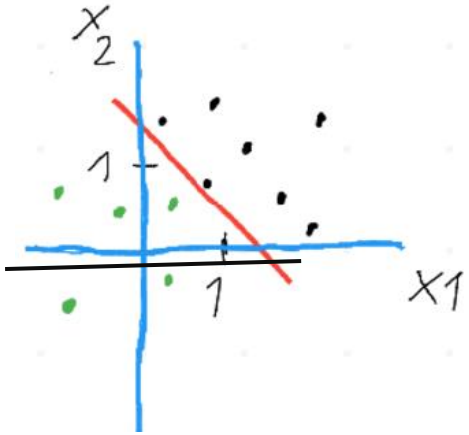
$$A = f(z) = \begin{cases} 0 & if \quad z \leq 0 \\ 1 & if \quad z > 0 \end{cases}$$

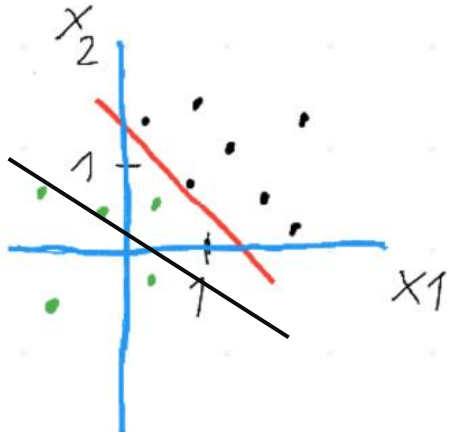
$$z = -2x_1 - 2x_2 + 3$$

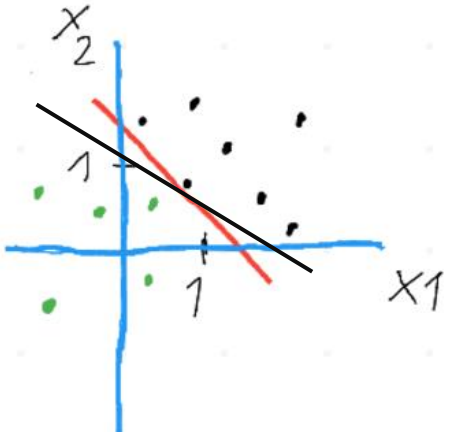


$x_2 = 1,5 - x_1$

Training:



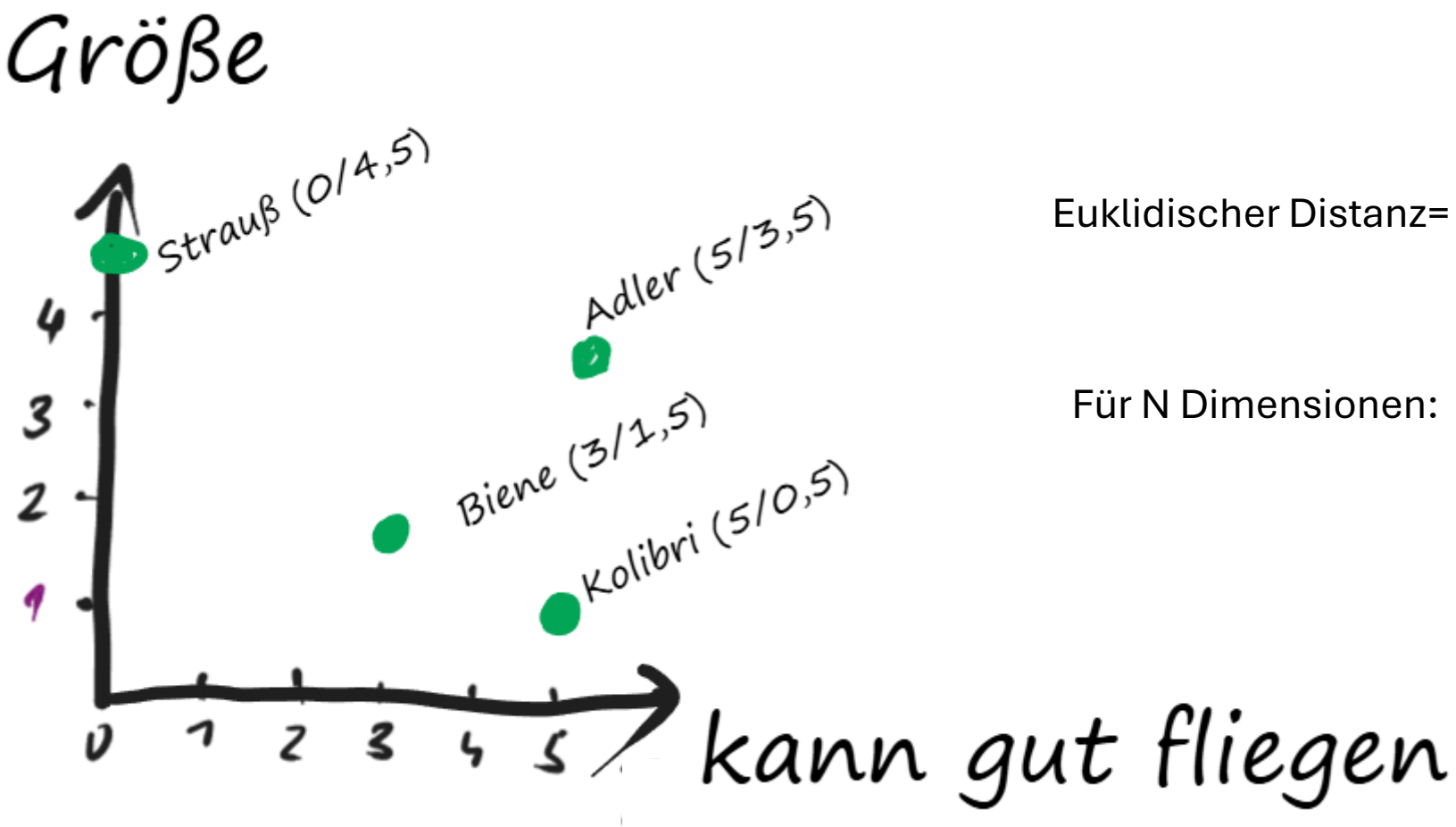




5.2 Motivation Neuronale Netze: Das Perzeptron - Begriffe

- Die *Inputs* eines Perzeptrons, x_i, sind die beobachteten Daten, die zum Training verwendet werden.
- Die *Gewichte*, w_i sind bestimmend für die *Activity Function z,* die jeden Input mit seinem Gewicht multipliziert und diese Zahlen aufsummiert.
- Der *Output* wird durch *Activation Function* bestimmt. Er ist 0 oder 1, je nachdem, ob das Ergebnis der Activity Function den Threshold übersteigt oder nicht.
- Das Prinzip des *Trainings* besteht in der Anpassung der Gewichte „in die richtige Richtung“ aufgrund der vorliegenen Inputs und Outputs (Trainingsdaten).
- Jeder Trainingsschritt (*Epoch*) passt die Gewichte „ein wenig“ an, sodass der *Verlust* kleiner wird. Wenn keine solche Anpassung mehr möglich ist, wird das Training beendet.
- Die Anpassung der Gewichte erfolgt durch eine *Training Rule*. Diese ist so gestaltet, dass die neuen Gewichte besser zu den Trainingsdaten passen als die alten – Abweichungen von den Trainingsdaten werden als *Verlust* bezeichnet.

5.3 Worteinbettung - Was ist das?



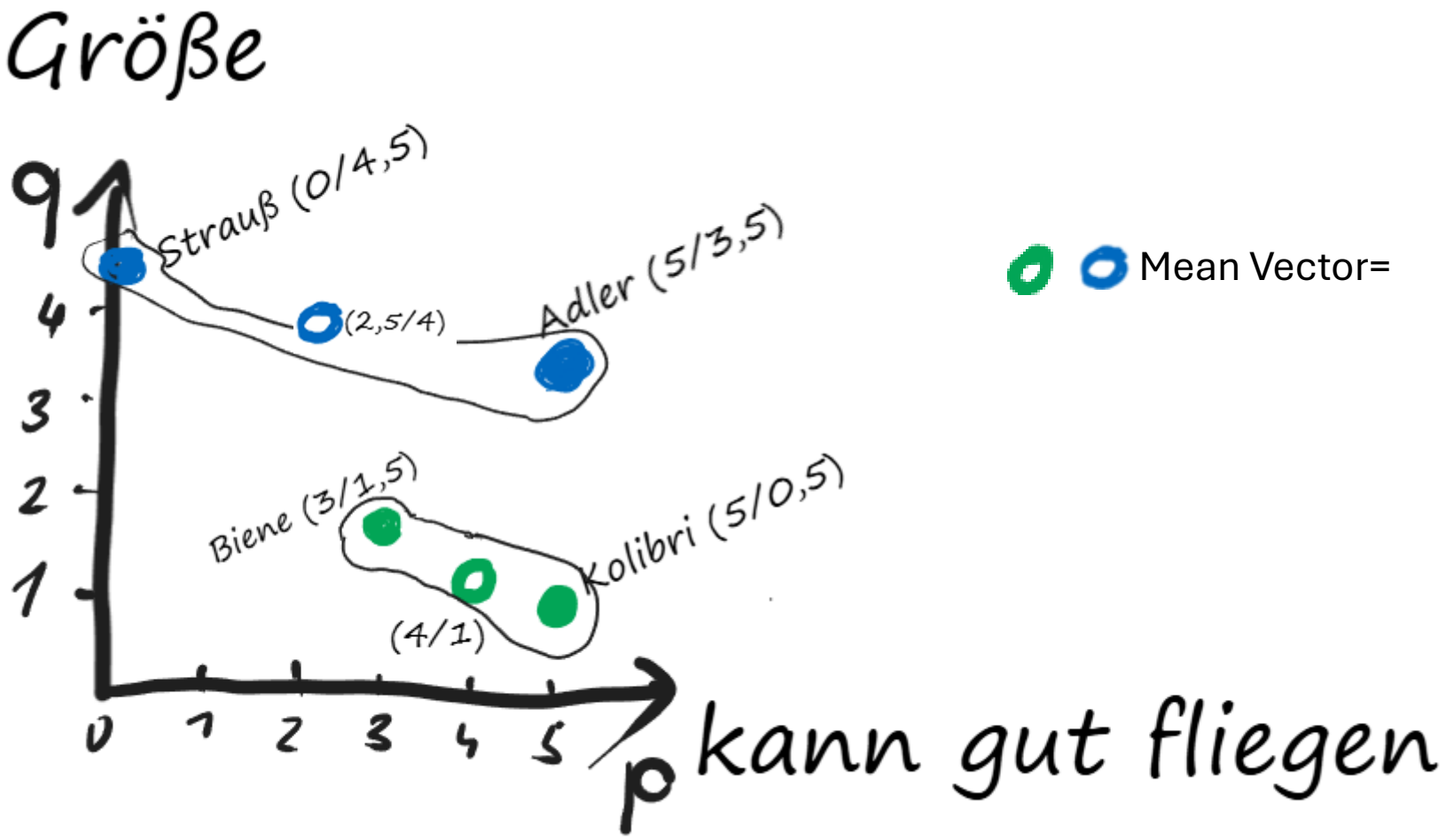
Euklidischer Distanz= $\sqrt{(q_1 - p_1)_2 + (q_2 - p_2)_2}$

Für N Dimensionen: $\sqrt{\sum_{i=1}^{N} (q_i - p_i)^2}$

5.4 Worteinbettung - Begriffe

- Die Wörter (im vorliegenden Vokabular) werden anhand einer Anzahl von *Kriterien* betrachtet *(Beispiel: Kriterium „Größe“ und „kann gut fliegen“)*.
- Jedes Wort erhält pro Kriterium eine Zahl für die *Ausprägung*, die das bezeichnete Objekt bei dem Kriterium hat. (Beispiel: „Strauß“ hat bei „Größe“ die Ausprägung 5, bei „kann gut fliegen“ die Ausprägung 0.).
- Es entsteht für jedes Wort ein *Einbettungsvektor*, dessen Betrag die Anzahl der Kriterien ist.
- Dadurch erhält man eine *Abbildung der Wörter in einen Vektorraum*, in dem „gerechnet“ werden kann. Zum Beispiel kann die Distanz zweier Wörter als die euklidische Distanz der Einbettungen berechnet werden.

5.5 Satzeinbettung



Mean Vector= $\begin{bmatrix} (p_1 + p_2)/2 \\ (q_1 + q_2)/2 \end{bmatrix}$

5.6 Satzeinbettung - Begriffe

- Eine Menge von Wörtern wird als Satz zusammengefasst
- Der Satz kann durch den *Mittelwertsvektor* dargestellt werden
- Diese Darstellung berücksichtigt NICHT die Position der Wörter
- Diese Darstellung verleiht jedem Wort das selbe Gewicht
- Es gibt elaborierte *Alternativen* zur Mittelwertsdarstellung (https://proceedings.mlr.press/v37/kusnerb15.html, https://aclanthology.org/D19-1410.pdf)

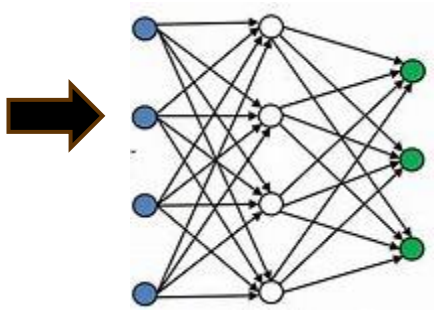
5.7 Worteinbettung durch unsupervised Learning

Das Klima in Dschibuti ist heiß

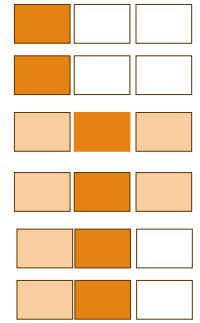
Das Wetter im Sudan ist heiß

Das Klima in Ägypten ist heiß

Das Wetter im Marokko ist heiß



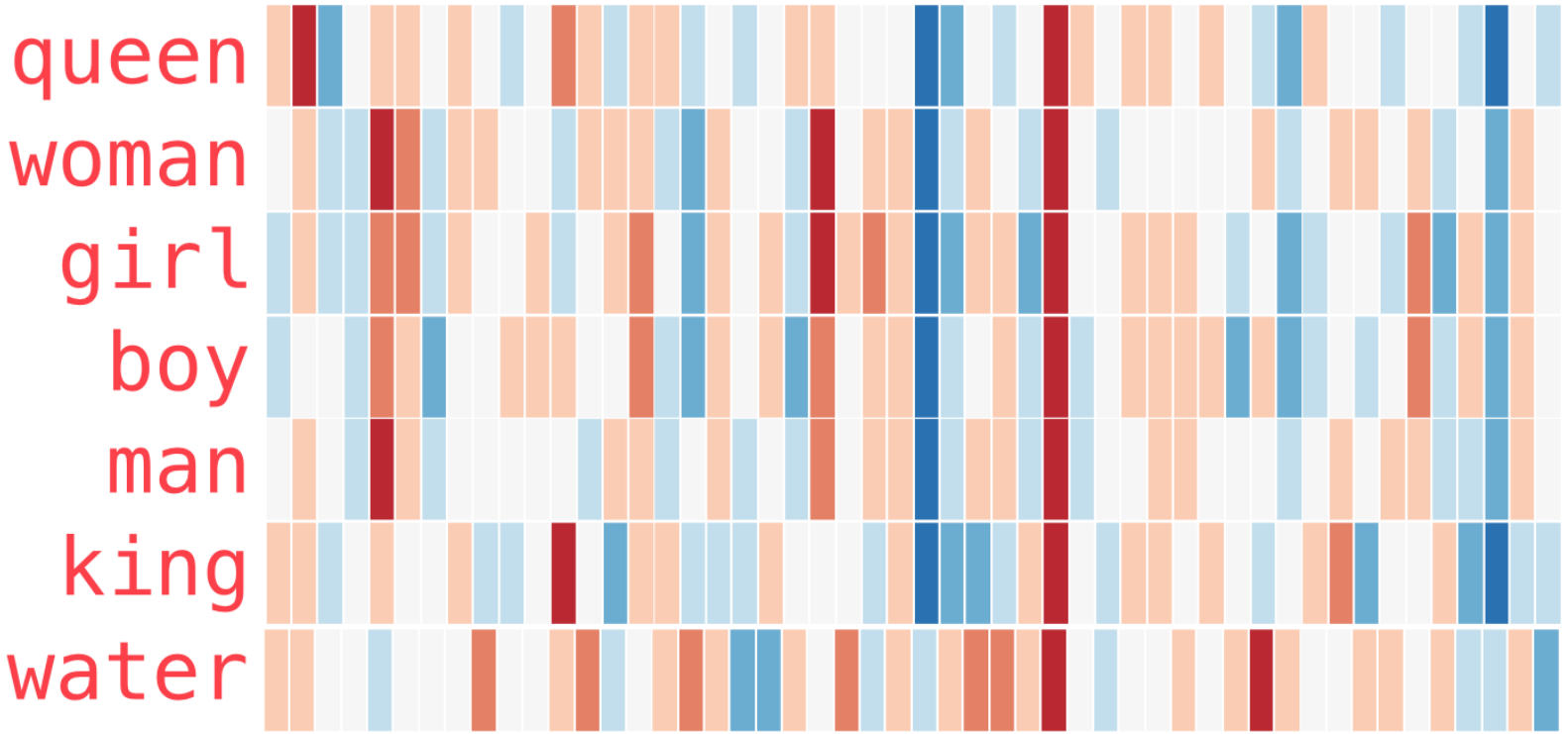
Klima
Wetter
Dschibuti
Sudan
Ägypten
Marokko



5.8 Worteinbettung durch unsupervised Learning: Begriffe

- Beim *unsupervised Learning* werden solche Wörter als ähnlich gewertet, die von den gleichen (oder ähnlichen) Wörtern umgeben sind
- *Ziel*: Ähnliche Wörter sollen ähnliche Wortvektoren erhalten
- *Verlustfunktion*: Der Gesamtverlust, der entsteht, wenn die resultierenden Vektoren die gefundenen Ähnlichkeiten nicht exakt abbilden, soll minimiert werden
- *Ergebnis*: Wortvektoren mit einer einstellbaren Dimension (=Anzahl der Features), wobei die Bedeutung der Features nicht explizit ist

5.9 Worteinbettungen (glove, wikipedia)



Quelle: https://jalammar.github.io/illustrated-word2vec/

5.10 Kontextuelle Einbettung: BERT

A) Masked Language Modeling

The quick brown fox jumped over the lazy dog

The [MASK] brown fox jumped over the [MASK] dog

B) Attention

the woman who lived on the hill saw a shooting star last night

C) Kontextualität Die Vektor-Einbettung eines Wortes hängt vom Text ab, in dem es steht

Der Gewinn der Deutschen Bank ist gestiegen → v1

Die Bank am Waldrand lädt zum Sitzen ein → v2

v1 ≠ v2

5.11 Kontextuelle Einbettung – BERT- Begriffe

- *BERT*: **B**i-Directional **E**ncoder **R**epresentations from **T**ransformers
- *Attention*:
  - für alle Wörter eines Satzes wird die Relevanz gelernt, die sie für diesen Satz haben. (z.B. ‚over')
  - auch Wortbeziehungen, die weiter entfernt im Satz stehen, werden erkannt
- *Kontextualität*: Wörter, die Verschiedenes bedeuten können, werden Homonyme genannt. BERT unterscheidet die Homonyme je nach dem Kontext, in dem sie vorkommen
- Output eines (gewichteten) BERT-Modelles
  - Worteinbettungen (Hidden)
  - Grad der Ähnlichkeit zweier Textpassagen
  - Klassifizierung eines Textes

5.12 Kontextuelle Einbettung: Varianten

- BERT (2017) 

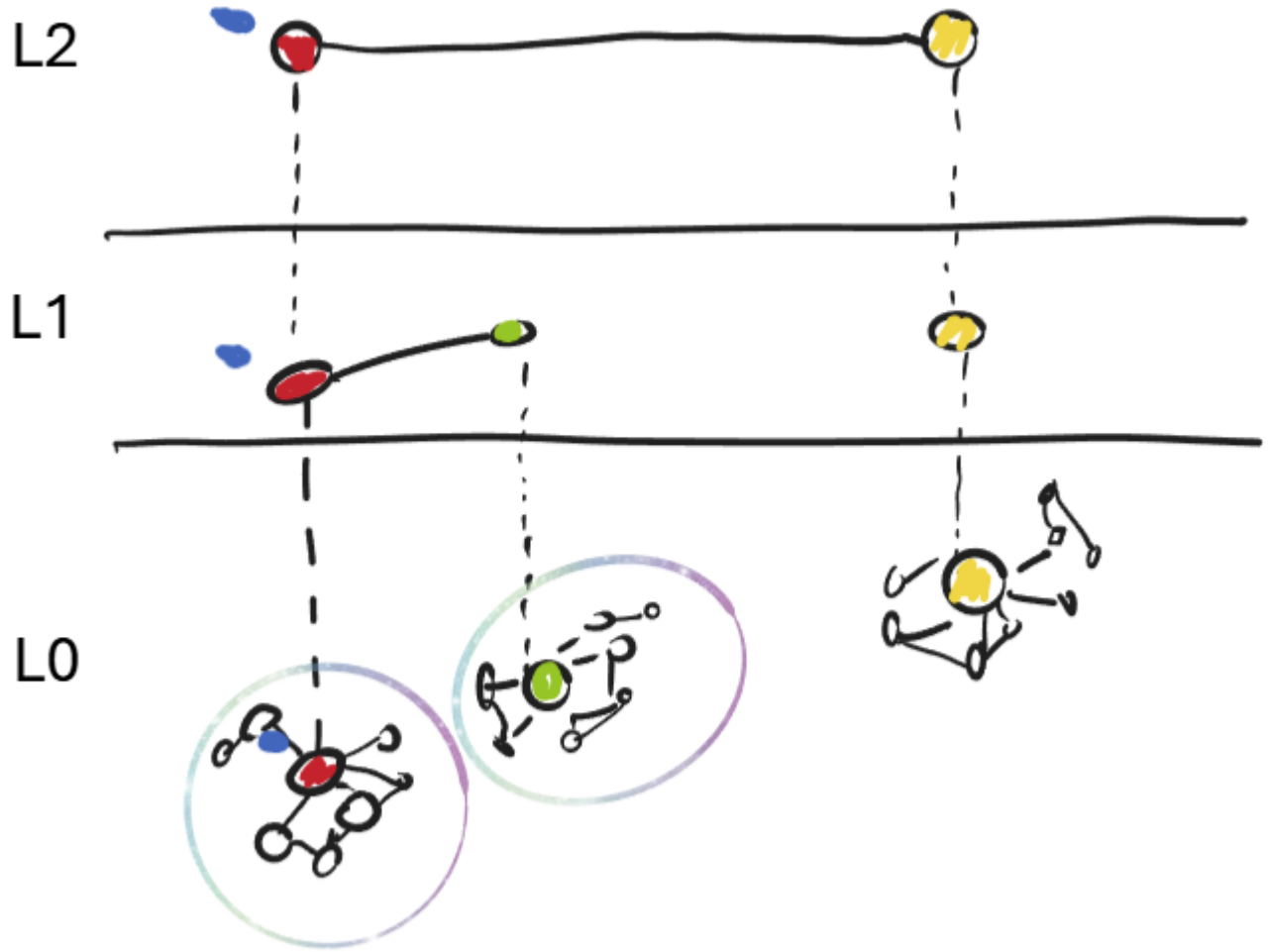
  - Input sind 1 oder 2 Textsequenzen
  - Einbettungen werden nur für Wörter erzeugt
  - Usage: Textklassifikation, STS, Q-A
- sBERT (sentence BERT) (2019) 

  - Input ist 1 Textsequenz
  - Einbettung von Wörtern im Kontext
  - Einbettung der gesamten Textsequenz (nicht-naiv)
- In Contrast: GPT (2017) 

  - Weiterentwicklung der BERT-Architektur
  - Usage: Textgenerierung

5. 13 Vektor-Indizes: Hierarchical Navigable Small Word – Technik (HNSW)

- Die Basisebene, L0, enthält alle Vektoren
- Auf jeder Ebene sind die Vektoren mit den nächsten Nachbarn verbunden
- Je höher die Ebene, desto weniger Vektoren
- Die Suche beginnt von oben
- Effekt: es müssen nur „wenige“ Distanzen von Vektoren gemessen werden
- Algorithmus: ANN = Approximate Nearest Neighbourhood



6. Validierung

6.1 Metriken - Benchmarks

- *Pearson/Spearman Correlation:* Messen die Stärke des linearen Zusammenhanges zwischen zwei Messreihen
- *F1 Score*: Harmonisches Mittel zwischen Precision und Recall. Liegt zwischen 0 und 1.
  - Precision: wie viele der Ergebnisse sind korrekt (Anteil)
  - Recall: wie viele korrekte Ergebniss wurden überhaupt gefunden (Anteil)
- *nDCG@K* (siehe Folgefolien)
- *MRR@K* (Mean Reciprocal Rank at K) – Wert zwischen 0 und 1

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}$$

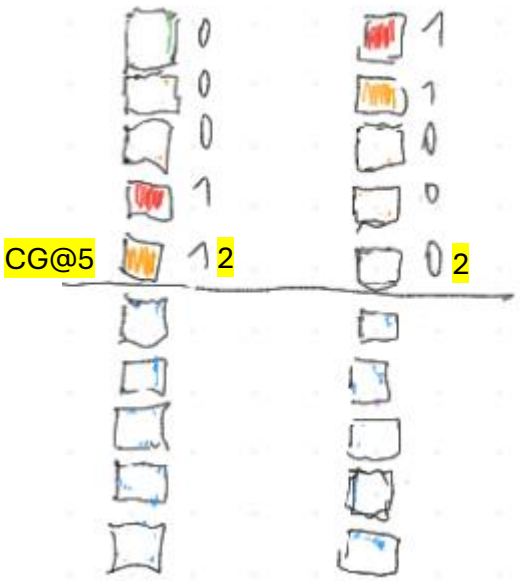
|Q|: Anzahl der Queries.

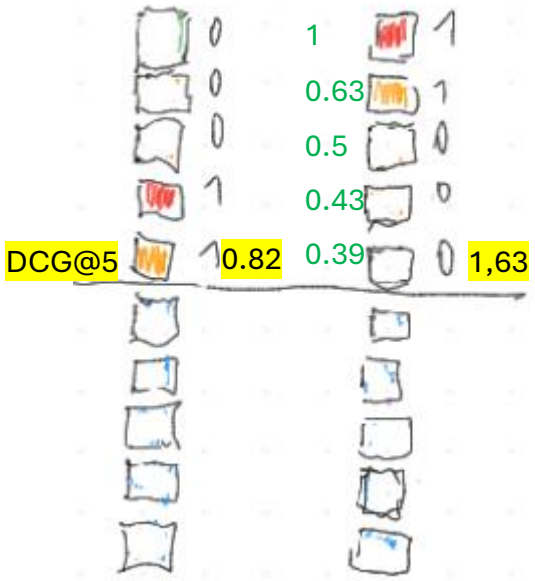
$rank_i$:Rang des ersten relevanten Dokumentes in Query i

- *MAP@K (Mean Average Precision@K) betrachtet die ersten K Antworten,* „belohnt“ Treffer, die weiter vorne im Ergebnis platziert werden und mittelt über mehrere Abfragen (liegt zwischen 0 und 1)
- *MTEB* Massive Text Embedding Benchmark (siehe Folgefolien)

6.2 normalized Discounted Cumulative Gain (nDCG@k)

- Misst, ob die relevanten Dokumente in der Ergebnisliste vorne stehen
- Wertebereich: [0, 1]





nDCG@5 = 0.82/1,63=0,50

$$CG@K = \sum_{1}^{K} G_k$$

$$DCG@K = \sum_{1}^{K} G_k \frac{1}{log_2(i+1)}$$

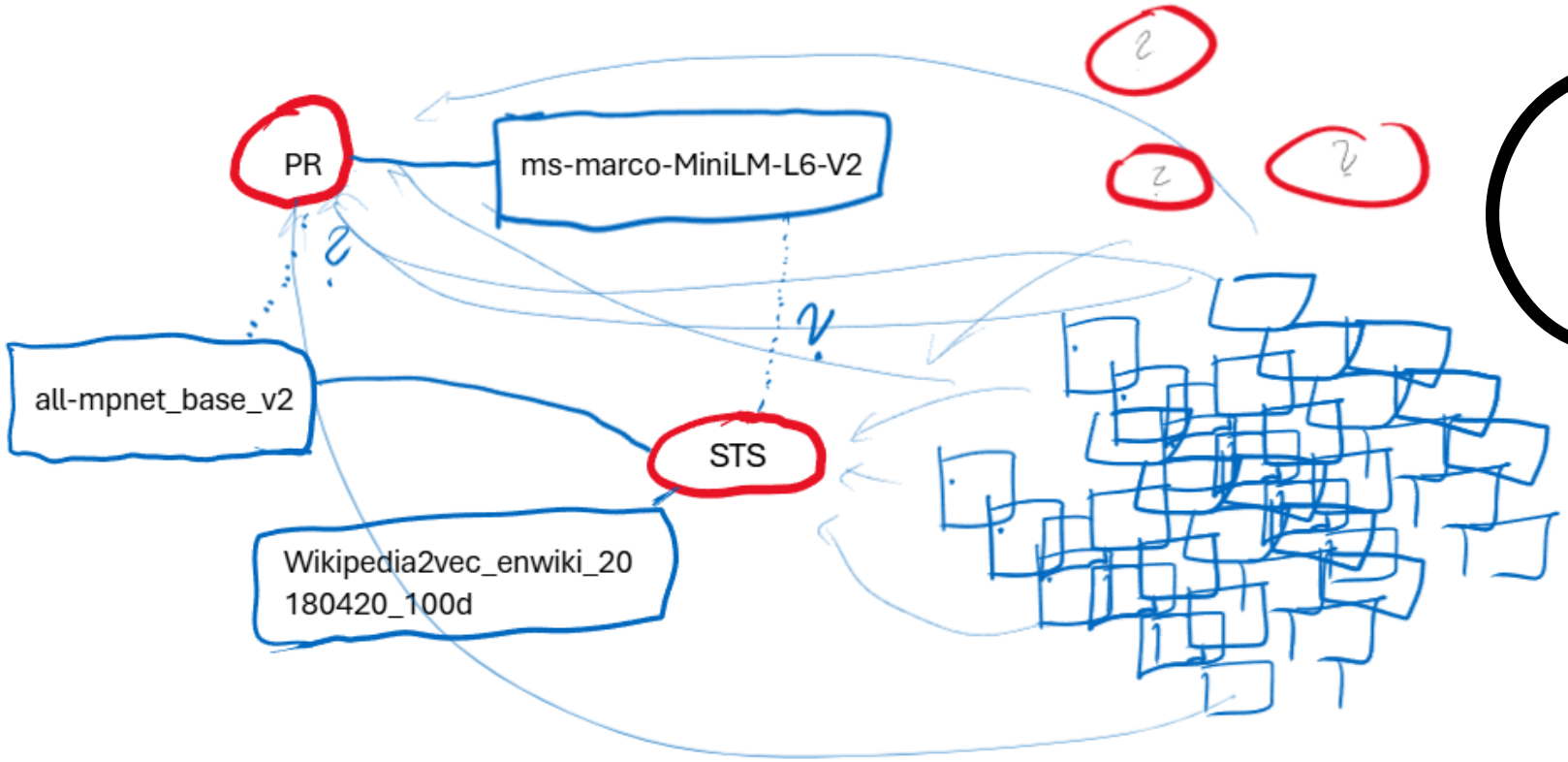
$$nDCG@K = \frac{DCG@K}{IDCG@K}$$

6.3 normalized Discounted Cumulative Gain (nDCG@k)

- *CG@K* – Cumulative Gain: Diese Maßzahl gibt an, wie viele relevante Dokumente in den ersten K Ergebnisdokumenten vorkommen.
- *DCG@K* – Discounted Cumulative Gain: Wie CG@K, nur dass das relevante Ergebnis umso stärker abgewertet wird, je weiter hinten es in der Liste der ersten K Ergebnisdokumente steht
- *IDCG@K* – Ideal Discounted Cumulative Gain: Wie DCG@K, aber für das „richtige“ Ergebnis
- *nDCG@K* – Wie „gut“ das Ergebnis abschneidet, wenn man es mit dem „richtigen“ Ergebnis vergleicht (Zahl zwischen 0 und 1)

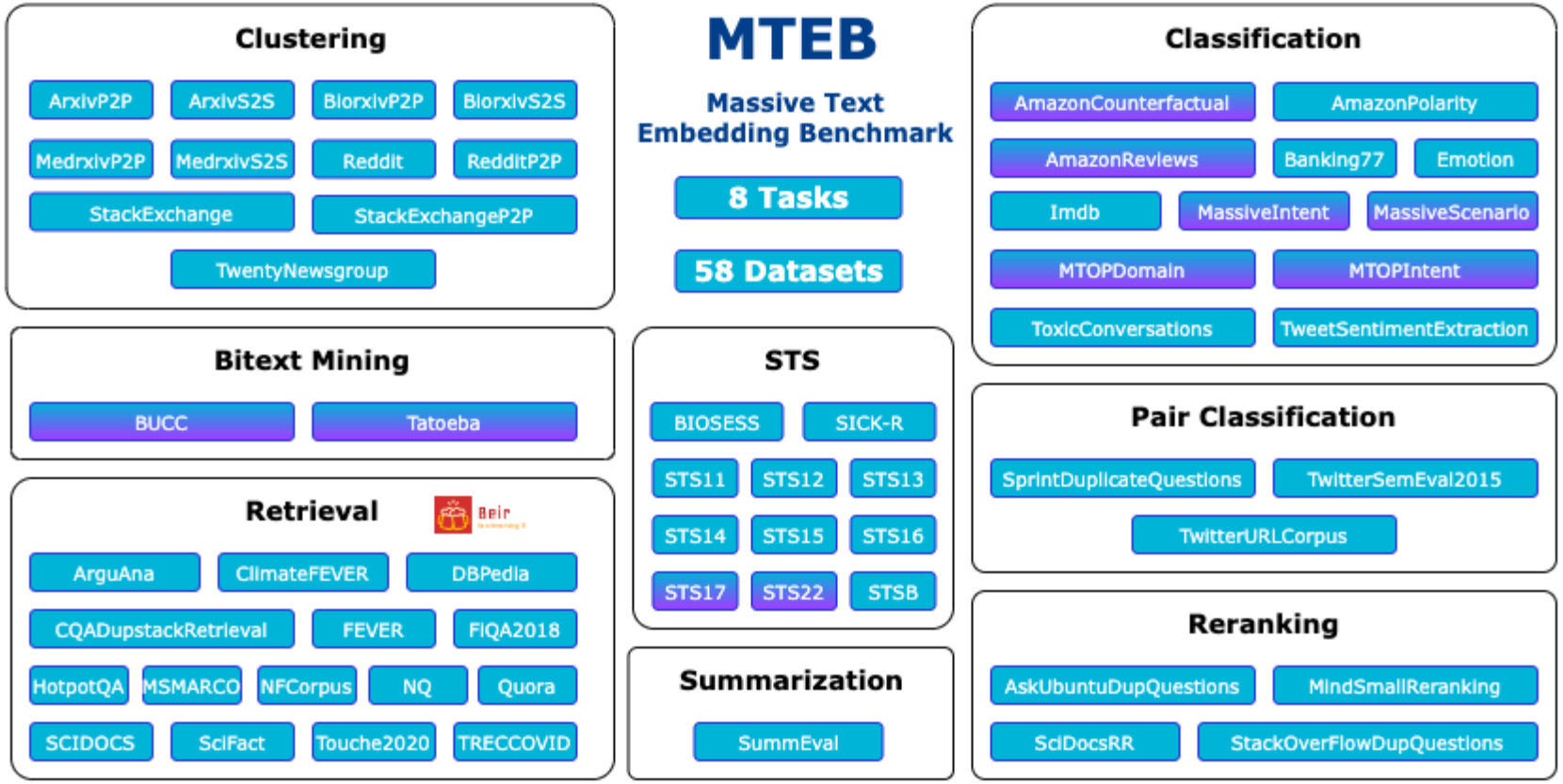
6.4 MTEB - Problemstellung

- Viele Modelle
- Kleine Trainings-Datesätze
- Für bestimmte Tasks



? Welche anderen Tasks
? Skalierung für große Datensätze
? Idente Datensätze
? Idente Metriken

6.5 MTEB – Massive Text Embedding Benchmark



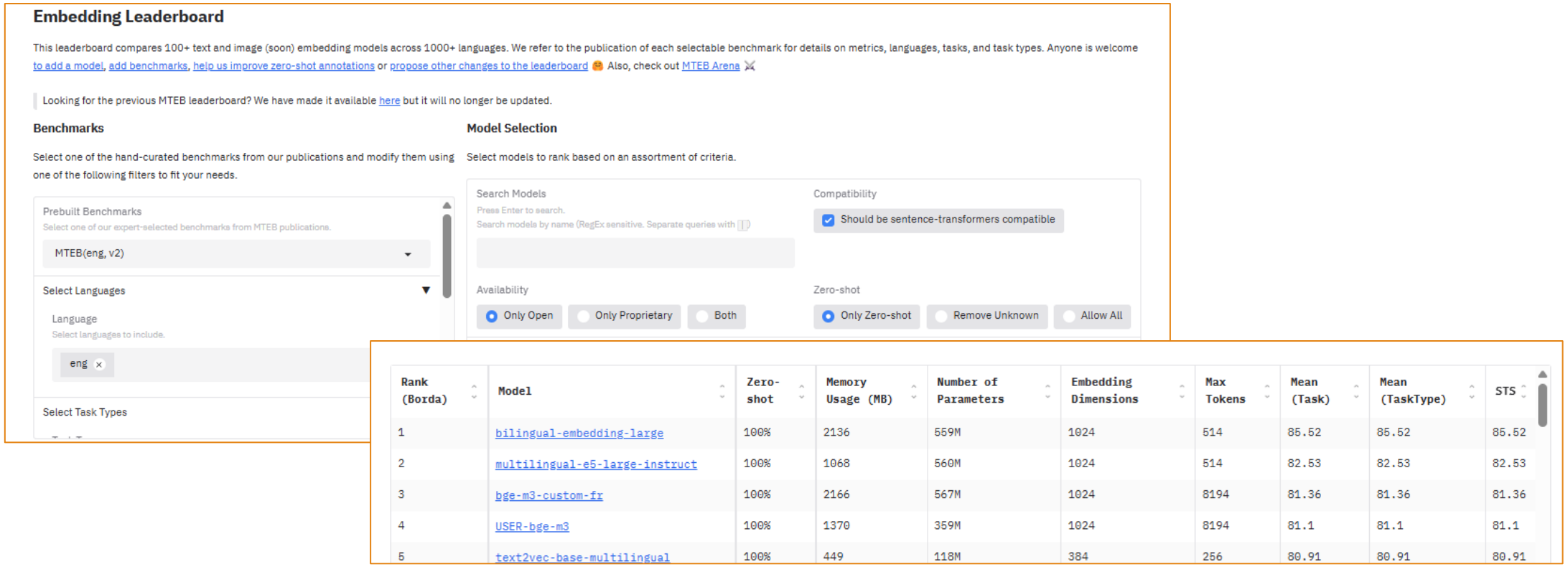
Quelle: https://arxiv.org/pdf/2210.07316

6.6 MTEB: Task Types für semantische Suche

- *Retrieval*: Gegeben ist eine Query, gesucht werden die relevantesten Dokumente. Metrik: nDCC@10
- *Semantic Text Similarity*: Gegeben zwei Sätze: wie ähnlich sind sich diese Sätze? Metrik: Spearman Corr.
- *Reranking*: Gegeben eine Query und eine Menge von Ergebnisdokumenten einer Abfrage: Was sind die korrekten Ranks der Dokumente? Metrik: MAP

6.7 Demo: Hugging Face MTEB Leaderboard

https://huggingface.co/spaces/mteb/leaderboard



Rank (Borda)	Model	Zero-shot	Memory Usage (MB)	Number of Parameters	Embedding Dimensions	Max Tokens	Mean (Task)	Mean (TaskType)	STS
1	bilingual-embedding-large	100%	2136	559M	1024	514	85.52	85.52	85.52
2	multilingual-e5-large-instruct	100%	1068	560M	1024	514	82.53	82.53	82.53
3	bge-m3-custom-fr	100%	2166	567M	1024	8194	81.36	81.36	81.36
4	USER-bge-m3	100%	1370	359M	1024	8194	81.1	81.1	81.1
5	text2vec-base-multilingual	100%	449	118M	384	256	80.91	80.91	80.91

6.8 MTEB Leader Board - Begriffe

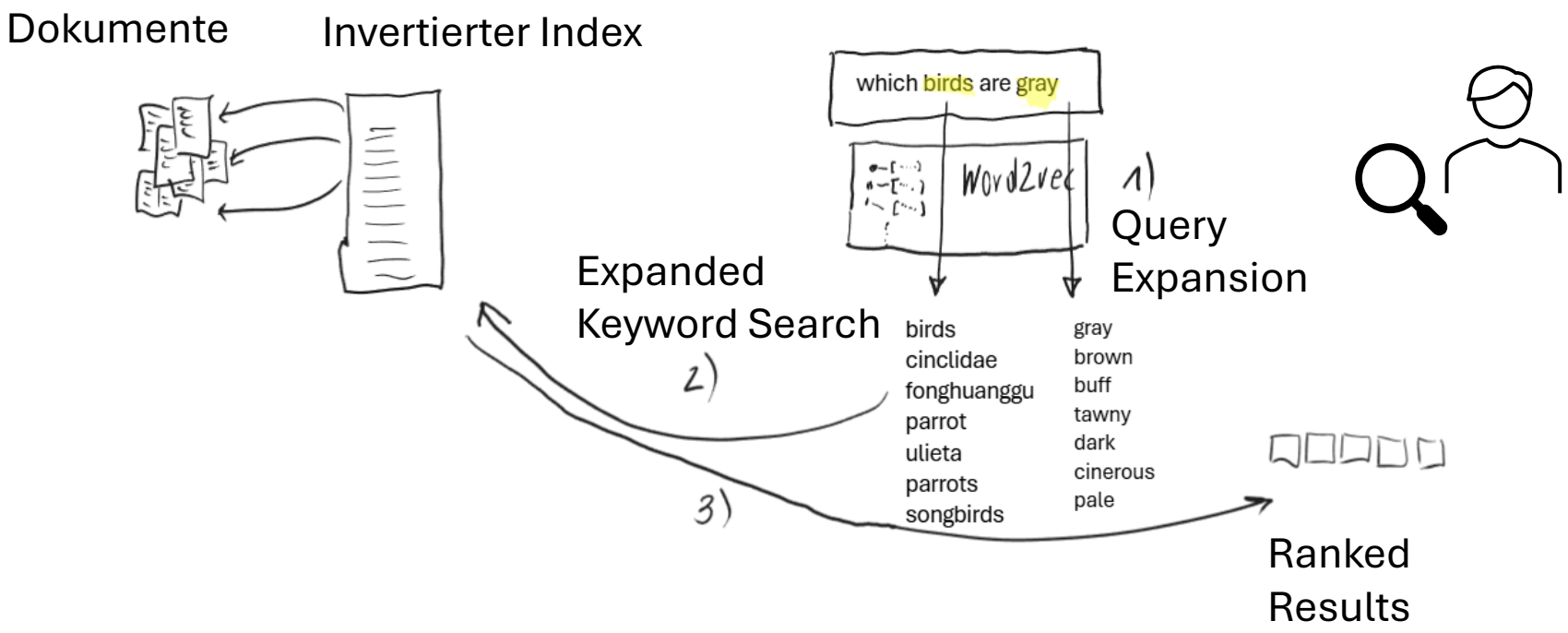
- Suchfilter
  - *Benchmarks*: BEIR oder MTEB
  - ***Task Type*: Retrieval, STS oder Reranking**
  - *Domain*: aus welchem Themenbereich kommen die Task-Daten
  - *Task*: Test-Datensatz – hängt mit Domain stark zusammen
  - *Only Zero Shot*: Nur solche Modelle finden, die ohne Fine Tuning Ergebnisse geliefert haben
- MTEB Summary Page
  - *Embedding Dimensions*: Länge der Vektoren
  - *Max Tokens*: maximale Länge eines „Dokumentes“ in Zeichen
  - *Mean (Task Type):* Zahl zwischen 0 und 100, keine absolute Aussage

6.9 Hugging Face

Thema	Inhalte
Model Card	• Welche IR-Use Cases werden abgedeckt
Trainings-Dataset	• Sprache, Umfang, Materie der Trainingsdaten • Paper-Verweis
Performance	MTEB-Leaderboard: https://huggingface.co/spaces/mteb/leaderboard • Search Model • Tab: Summary
Dokumentationen	Library-Dokumentationen
Community Blog	Artikel über Model Updates, Hands-On-Artikel, Übersichten
Learning	„Kurse“ / Tutorials über LLM, …

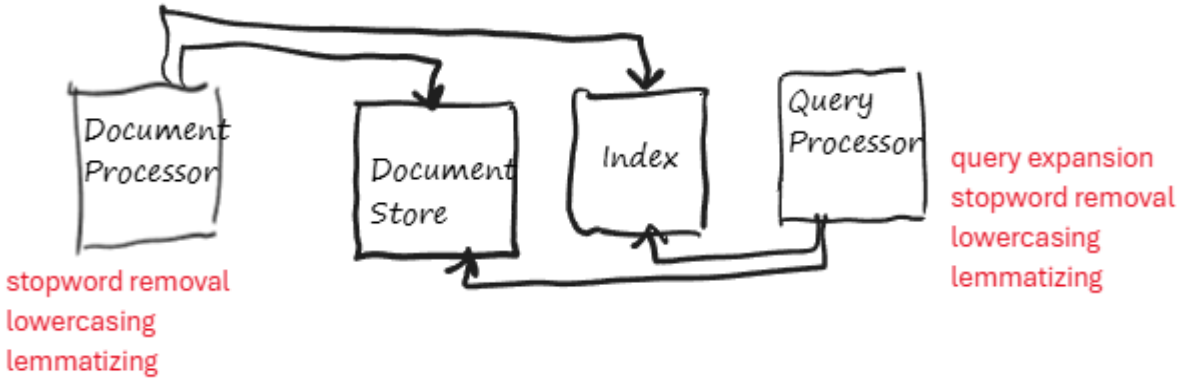
7. Anwendungen

7.1 Anwendung: Query Expansion

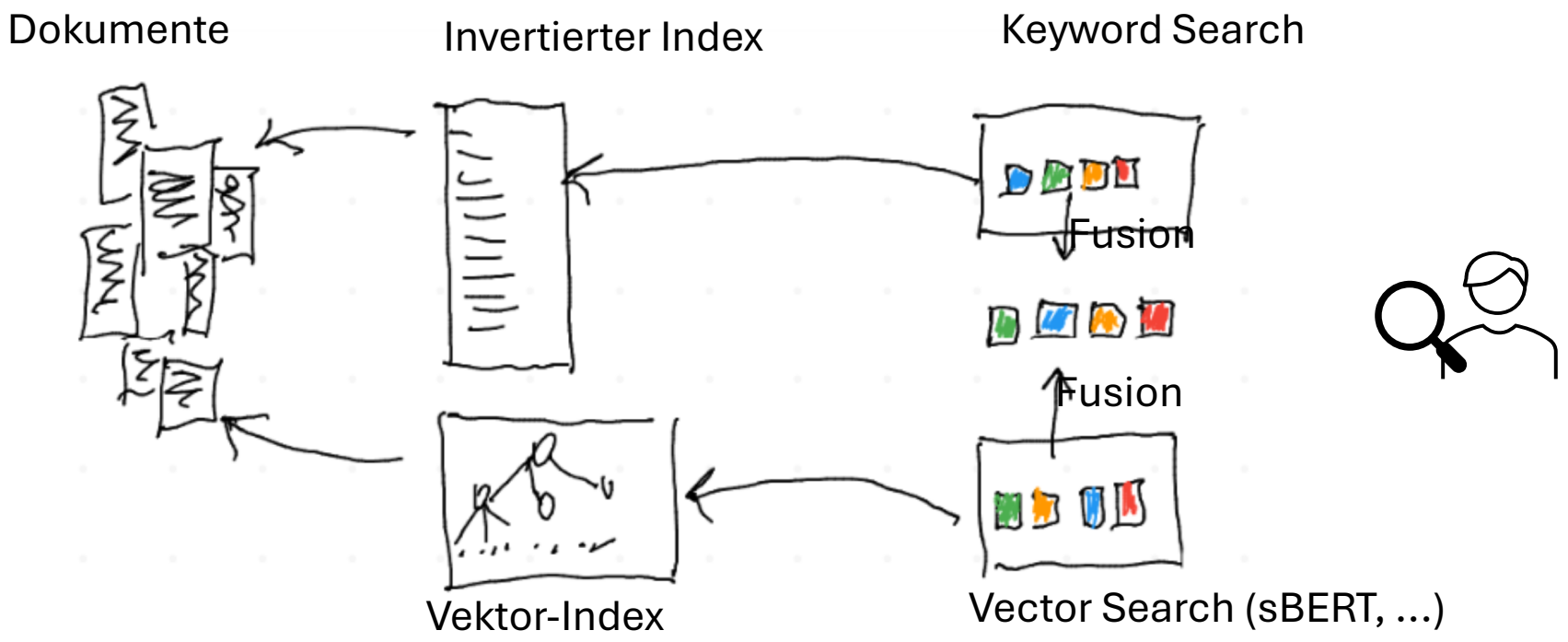


7.2 Query Expansion: Details

- *Query expansion:* Die Wörter der Abfrage (Substantiva, Verben, Adjektiva) werden durch ähnliche Wörter ergänzt
- *Stopword removal:* Die Stoppwörter werden aus der Query entfernt und auch nicht indiziert
- *Lemmatizing*: Ersetzung der Wörter durch Stammformen beim Indizieren und beim Abfragen
- Mit der veränderten Query wird im Index gesucht



7.3 Anwendung: Hybride Suche

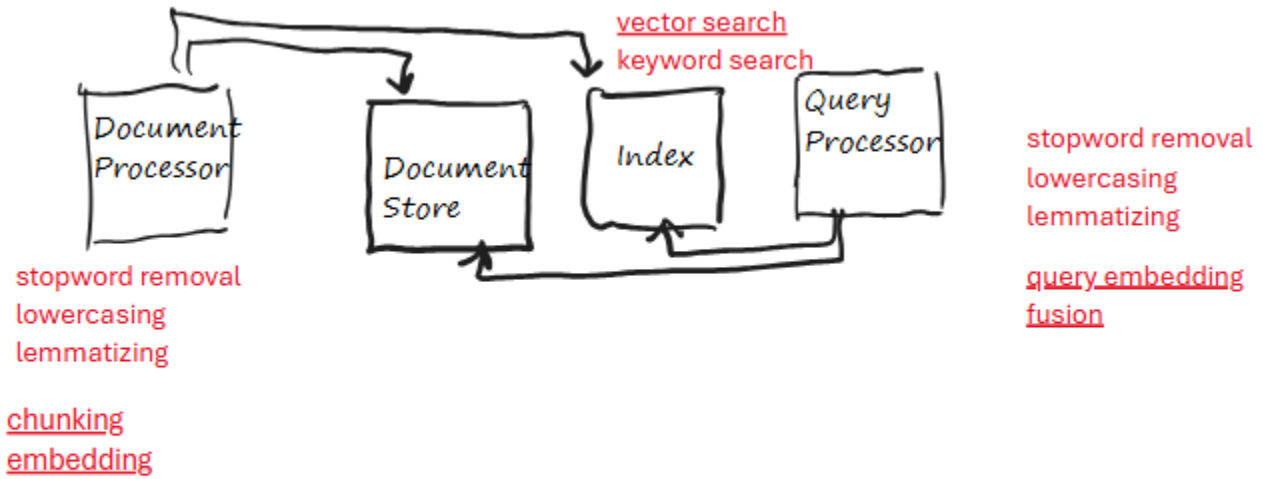


Fusion Algorithm:
z.B. Reciprocal Rank Fusion

$$rank(d \in D) \sum_{i=1}^{|methods|} \frac{1}{rank_i(d) + 60}$$

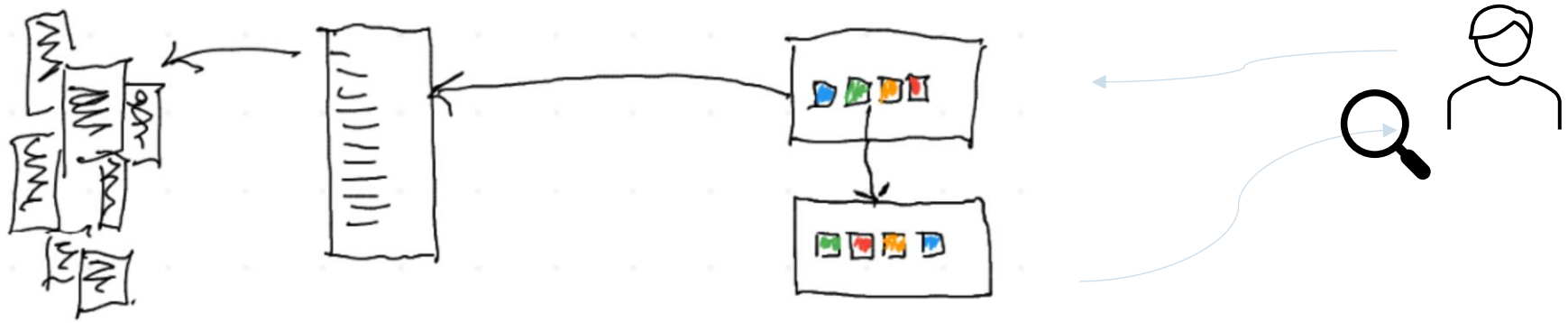
7.4 Hybride Suche: Details

- *Hybrider Idex*: Die Dokumente werden sowohl durch einen Inverted Keyword Index als auch durch einen Vektor-Index indiziert.
- *Hybrides Query Processing*: Die Query wird 2x prozessiert. Für den Vektor-Index wird die Query embedded und der Query Vektor wird im Vektor-Index gesucht
- *Hybrides Document Processing*: Die Dokumente werden für die Vektor-Indizierung gechunkt und die Chunks eingebettet



7.5 Anwendung: Re-Ranking

Dokumente Invertierter Index Search Engine (Lucene)



Re-Ranker (BERT, ...)

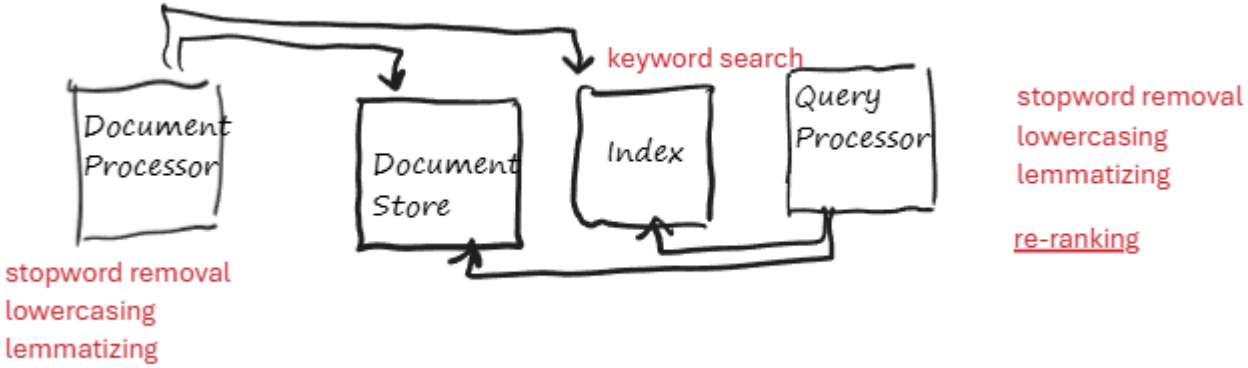
MRR@10-Benchmark (0-100%): Wie oft kommt das erste relevante Ergebnis von 10 an die erste Stelle?

Method	MS MARCO MRR@10 Dev	MS MARCO MRR@10 Eval	TREC-CAR MAP Test
BM25 (Lucene, no tuning)	16.7	16.5	12.3
BM25 (Anserini, tuned)	-	-	15.3
Co-PACRR* (MacAvaney et al., 2017)	-	-	14.8
KNRM (Xiong et al., 2017)	21.8	19.8	-
Conv-KNRM (Dai et al., 2018)	29.0	27.1	-
IRNet†	27.8	28.1	-
BERT Base	34.7	-	31.0
BERT Large	36.5	35.8	33.5

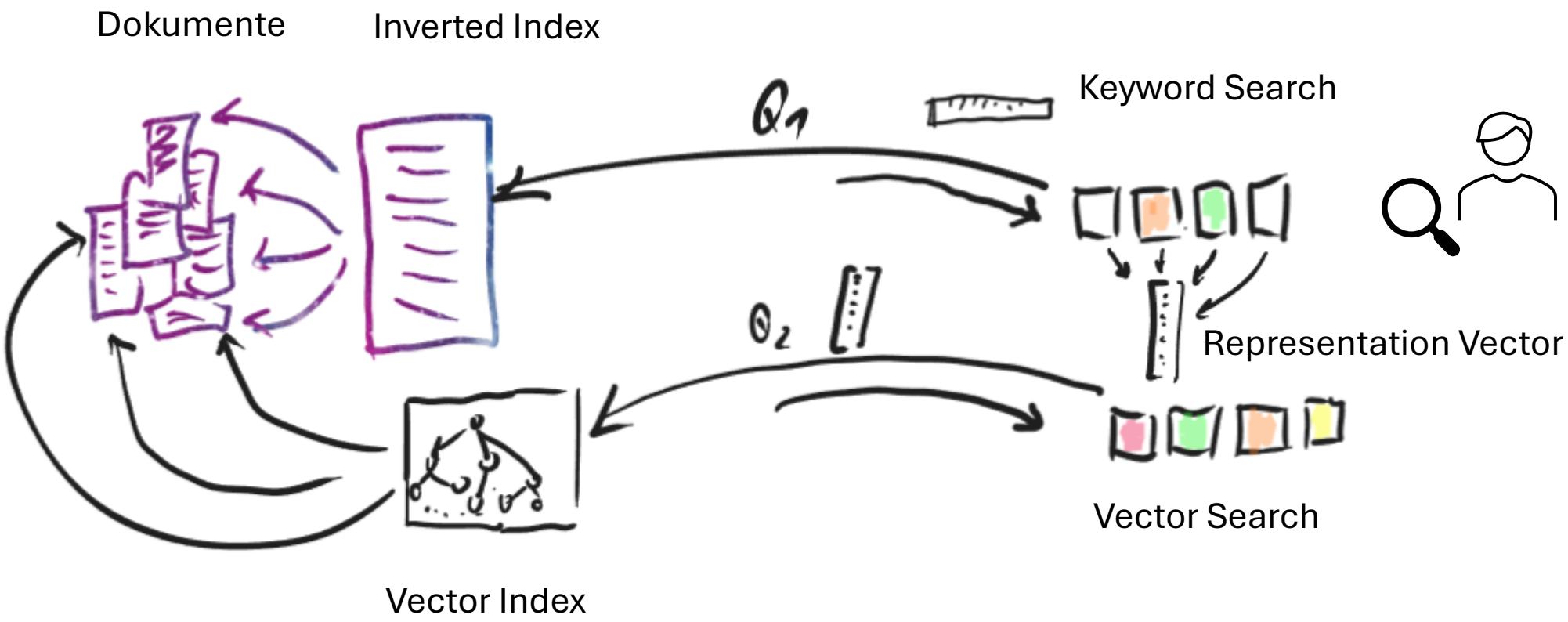
Quelle: https://training.continuumlabs.ai/disruption/search/bert-as-a-reranking-engine

7.6 Re-Ranking: Details

- Die Abfrage wird mit einem „üblichen“ Verfahren ausgeführt
- Die Ergebnisse werden mit Hilfe eines Re-Ranking-Modelles umgeordnet
- Dabei wird jedes Ergebnisdokument paarweise mit der Abfrage durch das Ranking-Modell prozessiert

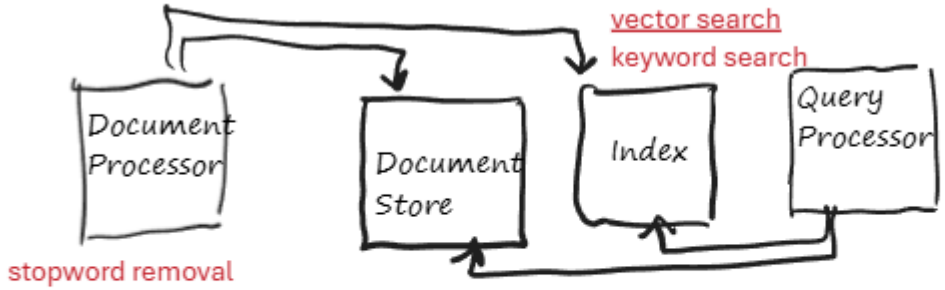


7.7 Anwendung: Pseudo Relevance Feedback



7.8 Pseudo-Relevance Feedback: Details

- Eine erste, konventionelle Abfrage liefert erste Antwortdokumente
- Die ersten Antworten werden eingebettet und als zweite Abfrage in einem Represetnation Vector zusammengefasst
- Der Representaction Vector ist z.B. der Mittelwertsvektor über die Vektoren der ersten Antworten
- Die zweite Abfrage, über einen Vektor-Index, liefert die finalen Dokumente



vector search
keyword search

stopword removal
lowercasing
lemmatizing

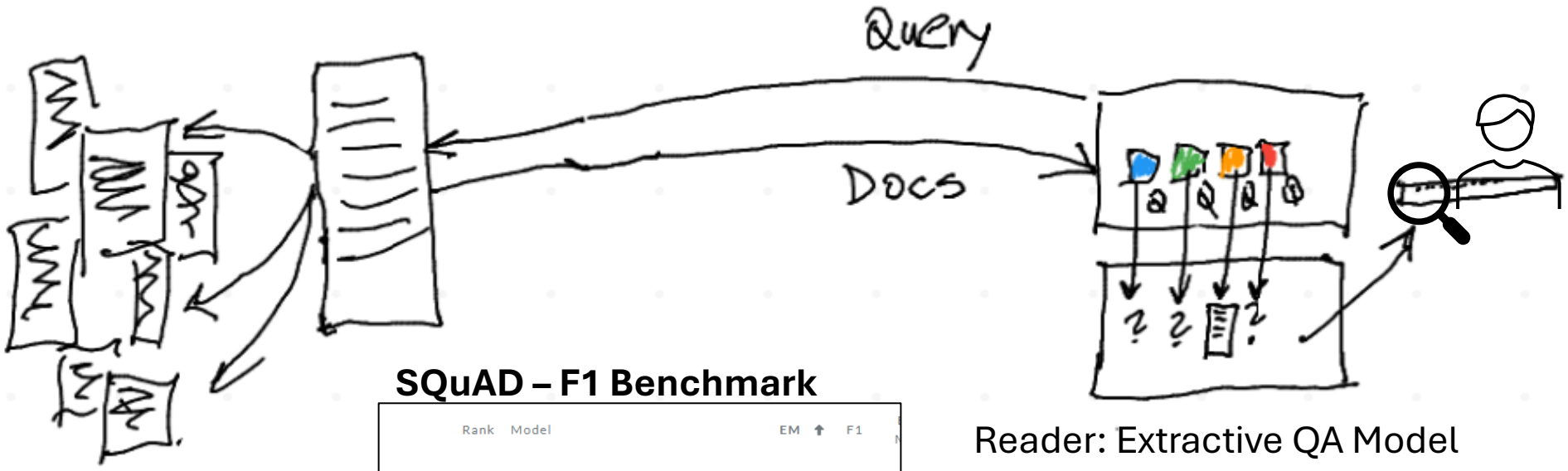
text embedding
vector averaging

stopword removal
lowercasing
lemmatizing

chunking
embedding

7.9 Anwendung: Question Answering

Dokumente Inverted Index Retriever: Keyword Search



Reader: Extractive QA Model

SQuAD – F1 Benchmark

Rank	Model	EM	F1
1	**LUKE**	90.2	
2	**XLNet** (single model)	89.898	95.080
3	**SpanBERT** (single model)	88.8	94.6

Quelle: https://paperswithcode.com/sota/question-answering-on-squad11?tag_filter=4

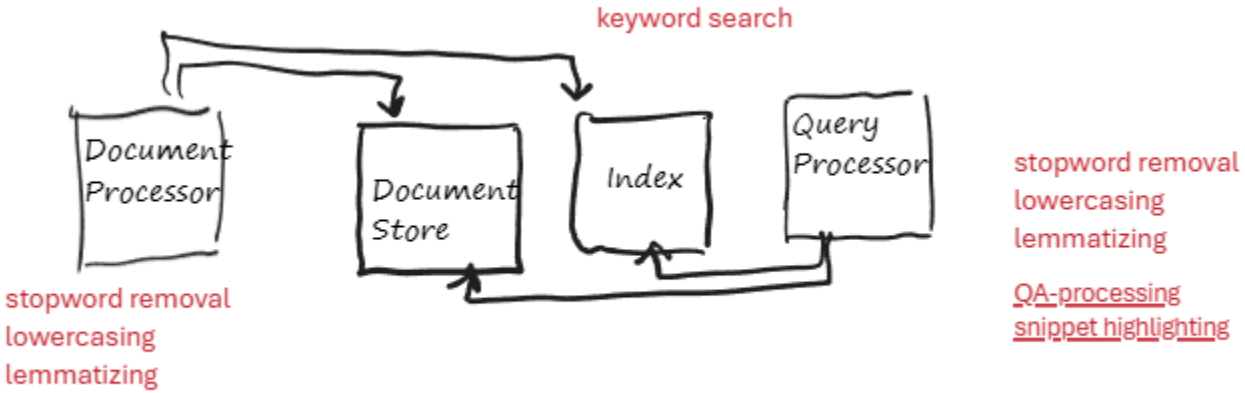
7.10 Question Answering: Details

Q: how can i turn on this device?

Doc: After I upacked the Laptop, I found a **power button at the rear side.** There is an interesting …

Answer: power button at the rear side

- Für QA gibt es spezielle Modelle - Extractive Model (Transformer based)
- Die Antwort-Dokumente eine Abfrage werden durch das QA-Modell prozessier
- Das Dokument mit der „besten“ Prediction ist das „beste“ Antwortdokument
- Die Antwort-Passagen in den Antwort-Dokumenten werden visuell markiert



8 Ausblick

8.1 Literatur

Information Retrieval (einführend): https://moodle.hochschule-burgenland.at/mod/resource/view.php?id=769812

Information Retrieval (Buch): https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf

Word Embeddings: https://www.ibm.com/think/topics/word-embeddings

Transformer-Modelle (BERT, ...) : https://huggingface.co/learn/llm-course/chapter1/4

Vector Indexing (HNSW): https://arxiv.org/pdf/1603.09320

Hybride Suche: https://weaviate.io/blog/hybrid-search-explained

Re-Ranking: https://www.pinecone.io/learn/series/rag/rerankers/

Pseudo-Relevance Feedback: https://ielab.io/publications/pdfs/li2022tois.pdf

Extractive Question Answering: https://arxiv.org/pdf/2311.02961

8.2 Libraries und Modelle

- Libraries
  - `gensim` – Worteinbettung
  - `sentence-transformers` – Satzeinbettung mit BERT
  - `huggingface_hub` – Model & Pipeline von Hugging Face
  - `annoy` – Vektor-Index
- Modelle (Hugging Face)
  - `Word2vec/wikipedia2vec_enwiki_20180420_100d` (Worteinbettung)
  - `sentence-transformers/all-mpnet-base-v2` (Texteinbettung)
  - `sentence-transformers/msmarco-bert-base-dot-v5` (Passage Retrieval)
  - `deepset/minilm-uncased-squad2` (Question Answering)
  - `cross-encoder/ms-marco-MiniLM-L6-v2` (Ranking)

8.3 Vorbereitungen für Workshop-2

- VS Code Extensions:
  - Python
  - Pylance Language Server
  - Python Debugger
  - Conda Environment Switcher
- Tools:
  - Python
  - Miniconda
  - Curl
  - Git command line cli

- Repository klonen: https://github.com/hochschule-burgenland/hbg-sems.git
- `readme.md` lesen und die angegebenen Schritte durchführen
- In Visual Studio Code: Python: select interpreter aufrufen und das Conda-Environment „sems“ auswählen