KI-basierte Wissensabfrage auf deutsch - PCnewsGPT

Noch nie zuvor hatten wir eine solche Menge an verfügbarem, nachschlagbarem Wissen. Egal ob für uns als Individuen, oder als Teil von Organisationen bzw. Firmen. Das immer größer werdende Problem dabei ist, das passende Wissen auch zu finden. Waren früher Verzeichnisse und Karteien bzw. später Suchmaschinen unsere Helfer, so kann uns die Künstliche Intelligenz (KI) hier völlig neue Perspektiven eröffnen, da die KI eben nicht nur so wie die Suchmaschinen Schlagwörter kennt, sondern die Bedeutung unserer Fragen verstehen kann. Durchaus ähnlich einem menschlichen Bibliothekar, bloß mit unendlicher Geduld und auch zumeist mehr Fachwissen.

In diesem Beitrag versuche ich eine Einführung zum den aktuellen Stand der KI zu diesem Thema zu geben. Speziell auch dahingehend, wie das technisch, am eigenen Computer, und für deutschsprachige Wissensinhalte funktioniert. Englisch ist ja die typische Internet- und KI-Sprache – die meiste KI wurde mit weniger als 0,1% Deutschsprachigen Inhalten angelernt.

Als praktisches Beispiel verwende ich einen ausprogrammierten Prototyp für die KI-Abfrage von deutschsprachigen Wissensinhalten am eigenen Computer – PCnewsGPT. Den Namen habe ich übrigens gewählt, da ich zum Testen der Lösung die Inhalte der Clubzeitung PC-News als Wissensbasis verwendet habe. Der Prototyp funktioniert aber natürlich auch für beliebige PDF-Inhalte mit eingebettetem Text.

## KI-Grundlagen, aber etwas anders …

Bevor wir in die Details einsteigen, hier ein kurzer KI-Überblick, um die Technik hinter PCnewsGPT einfacher zu verstehen.

Was ist eigentlich Intelligenz? Frei nach Mustafa Suleyman, dem ex-Chef von DeepMind/Google bedeutet dies:

1. Die aktuelle Situation zu erkennen: „Synthesize Information“/Verstehen
2. Darauf basierend Vorhersagen treffen zu können: „Make Predictions“/Vorhersage
3. Und ausgehend davon Aktionen zu setzen: „Take Actions“/Aktionen

Punkt 3/Aktionen klammern wir hier weitgehend aus. Unsere KI ist bloß ein Kopilot, der uns als Kapitän unterstützt. Die KI handelt hier nicht selbständig, sondern hilft uns nur mit Wissensant­wor­ten auf Fragen.

Die Punkte 1/Verstehen der Frage bzw. des Wissenskontexts, und 2/Vorhersage der zur Frage und dem Kontext passenden Antwort, sind das Wesentliche an KI-Lösungen wie z.B. chatGPT. Diese KI-Lösungen basieren auf KI-Modellen, sogenannten **Large Language Models** (LLMs), welche unsere Sprache sehr gut verstehen. Sie sind dabei Programme, welche stark vereinfacht unserem Gehirn nachempfunden sind – als **Künstliche Neuronale Netzwerke**. Die Neuronen darin sind eigentlich mathematische Formeln und Berechnungen, und sehr viele dieser Neuronen sind nebeneinander in vielen Schichten und auch hintereinander dann zu einem Netzwerk verbunden.

Das Wissen der KI wird dabei vorab von ihr aufwändig und selbständig von Lernmaterialien angelernt. Als Lernmaterial dient dabei ein ausgewählter, großer Teil des Internets. Beim Lernen speichert die KI dabei nicht die Eingaben, sondern sie ändert im Lernprozess die Feinstruktur des Neuronalen Netzwerkes – wie intensiv bestimmte Neuronen zusammenhängen, und wie sie reagieren sollen. Und zwar sowohl im Teil 1/Verstehen als auch für Teil 2/Vorhersage. Diese angelernten Einstellungen werden als **Parameter** des KI-Modells bzw. LLMs bezeichnet. Ein gutes, aktuelles LLM hat viele Milliarden bis zu einer Billion Parameter.

Die KI bildet beim Lernen selbständig eine Art mentales Modell für Verständnis und Vorhersage im LLM. Sie lernt und verdichtet dabei die Zusammenhänge des Lernmaterials und speichert nicht bloß dessen Texte, wie normale Programme das tun. Dieses Lernen ist aber extrem berechnungsintensiv, die aktuellen KI-Anwendungen lernen daher im aktiven Betrieb nichts mehr dazu.

## Transformer – der KI-Durchbruch

Die Elemente des Neuronalen Netzwerkes sind zumeist in sogenannten „**Transformer-**“ Strukturen angeordnet. Die Anwendung der Transformer-Idee verhalf der KI erst zum Durchbruch – chatGPT steht übrigens für „chat“ mit **G**enerative (erzeugt Antworten) **P**retrained (d.h. angelernter) **T**ransformer.

Ein Transformer teilt das neuronale Netz in die bereits bekannten, aber getrennten Neuronalen Netzwerkteile für 1/Verstehen und 2/Vorhersage. Er analysiert die Bedeutung der Eingabe und liefert eine Vorhersage für eine passende Antwort – und zwar Wortteil für Wortteil zusammengesetzt.

Transformer LLMs haben folgende wesentliche Qualitätskennkriterien:

1. Die Anzahl der Modellparameter – mehr heißt meist besser, benötigt aber auch weit mehr Rechenleistung – sowohl zum Anlernen als auch zum Ausführen.  
   Typische aktuelle Transformer LLMs haben einige Milliarden (engl. Billion, z.B. Meta’s LLaMA-2 7B) bis zu einer Billion (engl. Trillion, geschätzt für GPT-4/chatGPT) Parameter.
2. Qualität des Lernmaterials – ist es bloß pures, unsortiertes Internet oder ausgewählt gutes Material; und wie sieht es mit internationalen Inhalten aus, oder ist es nur Englisches.  
   Einer der Qualitätsaspekte von chatGPT ist, dass es zusätzlich noch mit menschlich ausgewähltem Material angelernt ist (Reinforced Learning with Human Feedback).
3. Maximale Eingabekontextlänge als Eingabelimit - der Teil 1/Verstehen des Transformers ist in der Eingabetextlänge limitiert. Ab dieser maximalen Länge „vergisst“ der Transformer alles Ältere, und ohne jegliche Warnung bzw. Fehlermeldung. Das Neuronale Netz verhält sich hier durchaus menschenähnlich, es hat eine Aufmerksamkeitsspanne, und Eingabetexte sind damit in der Länge limitiert. Erschwerend dabei ist, dass während des Antworterzeugens und als Gedächtnis im Dialog hier alles mitgezählt wird – der Transformer lernt ja eigentlich nicht dazu, alles ist im aktuellen Netzwerkteil 1/Verstehen inkludiert. Längerer maximaler Eingabekontext vergrößert typischerweise auch die Rechenleistungsanforderungen dramatisch.

## Mehr KI-Fachchinesisch: Transformer, Embeddings, Vektordatenbanken

Ein Transformer besteht ja im Wesentlichen aus den Bereichen 1/Verstehen und 2/Vorhersage. Der Übergang zwischen diesen beiden Teilen ist ein einzelner Zahlenvektor, bezeichnet als „**Embeddings**“. Dieser Vektor besteht aus vielen hunderten bis tausenden Gleitkommazahlen, abhängig vom KI-Modell. Dieser Embedding-Vektor ist damit eigentlich eine Darstellung der verstandenen inhaltlichen Bedeutung der Texteingabe des Transformers. Embeddings sind daher eine Codierung der inhaltlichen Bedeutung eines Textes durch einen Vektor. Damit sind Transformer nicht nur insgesamt fürs Erzeugen von Antworten interessant, sondern deren erster Teil (1/Verstehen) mit dem Ergebnis-Embedding-Vektor hilft auch bei der Erfassung der Bedeutung von Texten.

Zwischen verschiedenen Embedding-Vektoren kann nämlich eine „Nähe“ berechnet werden, d.h. wie ähnlich sich Texte von der Bedeutung her sind. Z.B. „Franz sucht Hans“ verwendet die gleichen Worte und mag für die schlagwortorientierten Suchmaschinen ähnlich zu „Hans sucht Franz“ klingen, deren Bedeutung und Embedding-Vektoren sind aber deutlich verschieden. Und umgekehrt sind eine andere Wortwahl bzw. Synonyme einander in Embeddings sehr nahe. Die Werte für die Embeddings ist ja über das angelernte Mentale Modell der KI gegeben. Oft geht die auch über Sprachgrenzen hinweg – wenn das KI-Modell mit mehrsprachigen Inhalten angelernt wurde (z.B. in dem es mit Wikipedia angelernt wurde), erkennt es meist auch Zusammenhänge außerhalb des eigentlichen Lernmaterials über Sprachgrenzen hinweg, d.h. versteht sehr viele Inhalte unabhängig von der Sprache.

Technisch gibt es einige verschiedene Methoden für die Berechnung der Nähe von Vektoren – es ist ja nicht ganz einfach, tausende Zahlen zu vergleichen – „Cosine-Similarity“ (d.h. mathematische Cosinus-Ähnlichkeit) ist dabei die am häufigsten verwendete.

Embedding-Vektoren ermöglichten eine neue Art von Datenbanken, nämlich Datenbanken, die über die Inhaltsbedeutung und nicht wie bisher nach Kategorien und Schlagwörtern abgefragt werden können. Diese Datenbanken werden als **Vektordatenbanken** bezeichnet. Vektordatenbanken liefern bei einer Abfrage eine gewisse Maximalanzahl passender Datenbankinhalte zurück, welche am besten zur Bedeutung der Abfrage passen. Dies erfolgt dadurch, dass sie beim Speichern von Inhalten deren Embeddingvektoren berechnen und diese mitspeichern. Bei einer Abfrage wird der Embeddingvektor berechnet, dann werden die Inhalte mit den nähesten Embeddings gesucht und zurückgegeben. Die auf die Abfrage zurückgegebenen Inhalte liefern auch gleich ihre Distanz – d.h. wie gut sie inhaltlich zur Frage passen – mit.

## KI und Wissensabfrage

Wir verwenden die aktuelle KI wie z.B. chatGPT eigentlich falsch – nämlich als ein Orakel für Wissen. Ihre eigentliche Stärke ist ja das Verständnis unserer Sprache sowie sprachlich gute Text-Antworten. Wissensspeicherung ist für die KI eigentlich problematisch.

Obwohl die KI große Teile des öffentlichen Internets gelernt hat, ist ihr spezifisches Wissen sehr beschränkt. Sie ist im Wissen zumeist sehr auf den angloamerikanischen Sprachraum fokussiert. Und das Wissen ist mit jenem Zeitpunkt eingefroren, zu dem sie fertig angelernt war – ein aktuelles Wissen kennt sie nicht. Und noch schlimmer, die Art wie sie programmiert wurde zwingt sie, auf Fragen immer mit etwas zu antworten – damit etwas zu erfinden bzw. mit etwas komplett Falschem zu halluzinieren. Diese Halluzinationen klingen dabei sprachlich korrekt und oft durchaus glaubhaft.

Ein besserer, zuverlässiger Ansatz wäre, einer allgemeinen, sprachgewandten KI/LLM eine faktenbasierte Wissensbasis zur Seite zu stellen. Und bei Fragen über dieses Wissen der KI die passenden Inhalte dieser Wissensbasis als Ausgangsmaterialen zu liefern, eben über eine Vektordatenbank. Die KI sollte somit nur zum Analysieren, Verdichten des Wissens aus der Vektordatenbank und zum Formulieren der Antwort verwendet werden inklusive der Antwort, dass die KI nichts zur Frage Passendes in der Wissensbasis gefunden hat, anstatt hier zu halluzinieren.

Diesen Technikansatz nennt man **Retrieval Augmented Generation** (RAG). PCnewsGPT ist ein Prototyp für eine deutschsprachige RAG-Implementierung.

## Problembereiche dieser Wissensabfrage

Dieser Retrieval Augmented Generation (RAG) Ansatz hat mehrere inhärente Schwachstellen, die für eine gute Lösung adressiert werden müssen. Einige davon sind:

1. Maximale Eingabekontextlänge des Transformer LLMs – damit muss das Wissen in kleine, unabhängige Teile zerteilt werden. Aus diesen werden dann eine gewisse Maximalzahl zur Frage passender Teile ausgewählt und der KI als Fragekontext mitgegeben. Dies alles erfolgt so, dass mit diesem Kontext, der Frage und der Antwort zusammen das LLM-Eingabekontextlimit nicht überschritten wird.  
   Das Zerteilen der Wissensbasis in die Teile bzw. Wissensfragmente muss vollautomatisch über Texterkennung so erfolgen, dass die Wissensfragmente in sich konsistent und gut sind. Und damit aus den einzelnen, unabhängigen Fragmenten ein guter Kontext erstellt werden kann.
2. Garbage-in-garbage-out: Die Qualität der Antworten ist sehr von der Qualität des Wissensbasismaterials abhängig. Eine gute Qualitätssicherung des Basismaterials ist wesentlich. Der RAG-Ansatz zwingt die KI, im Zweifelsfall den Inhalt der Wissensbasis zu verwenden, selbst wenn er eventuell im Widerspruch zum Basiswissen der KI ist.  
   Ein Beispiel: Wenn Dokumente automatisch importiert werden, sollten Verzeichnisse (Inhalt, Stichworte) ignoriert werden. Oder Wissensbereiche, die über mehrere Seiten gehen, sollten nicht durch die Seitengrenzen in willkürlich getrennte Bereiche zerstückelt werden.
3. Das LLM und auch die Embeddings müssen die Sprache der Wissensbasis und der Abfrage gut unterstützen. Die häufig verwendeten Modelle sind oft nur auf Englisch optimiert.
4. Wissen hat oft einen Zeitbezug, Vektordatenbanken betrachten aber nur Inhalte, ohne Zeitbezug. Es ist daher sinnvoll, in den Metadaten der Vektordatenbankinhalte auch Informationen über das jeweilige Quelldatum des Inhalts mitzuspeichern.  
   Die Wissensabfrage sollte dann neuere Quellen gegenüber inhaltsähnlichen älteren bevorzugen, wenn sie sich widersprechen. Dabei nehmen wir an, dass spätere Wissensteile in sich korrekt sind, d.h. etwas Neueres nicht bloß zusätzlich zum Älteren gilt.  
   Ein Beispiel: Informationen zu Lebensläufen von Personen – diese Person hatte in einem Textmaterial zum Zeitpunkt a die Berufsposition x, in einem anderen Material zu einem späteren Zeitpunkt b die Position y. Meist ist damit die Position x nicht mehr aktuell und sollte im Wissen zumindest als frühere Position erkannt und als solche behandelt werden.

## Machine Learning – Fine-Tuning

Als Ergänzung ist es eventuell noch interessant, eine allgemeine KI mittels Feinabstimmung weiter anzulernen, damit sie in ihrer Sprache, bzw. den Formulierungen so klingt, wie es für die spezifische Organisation oder Firma typisch wäre. Dies ist aber KEIN eigentliches Wissensanlernen der KI. Der PCnewsGPT Prototyp hat diesen Teil noch nicht implementiert.

## PCnewsGPT ein deutschsprachiger, lokaler RAG-Prototyp

Das Projekt PCnewsGPT versucht diese Idee in einer speziell auf deutschsprachigen Inhalt abgestimmten Lösung umzusetzen und diese Lösung rein auf einem lokalen Computer laufen zu lassen. Eine Lösung, bei der das Wissen komplett lokal und vertraulich bzw. sicher verarbeitet wird – d.h. ohne öffentlich neues, zukünftiges KI-Lernmaterial mit eventuell vertraulichen, urheberrechtsgeschützten Inhalten zu liefern. PCnewsGPT wird als Open-Source Python-Programme unter Apache-Lizenz realisiert und ist öffentlich verfügbar (https://github.com/AndreasKunar/pcnewsgpt) bzw. jederzeit anpassbar. *PCnewsGPT ist aber nur ein laufend weiterentwickelter Prototyp, jegliche Anwendung erfolgt auf eigene Gefahr.*

Als Testwissensbasis wird die ClubComputer Zeitschrift PC-News verwendet – mit ihren bisher über 175 Ausgaben. Dies erfolgt ohne jegliche menschliche Inhaltsaufbereitung wie Kategorisierung, Beschlagwortung, etc. – rein über das Importieren der öffentlich (https://clubcomputer.at/pcnews/) verfügbaren PDF-Dateien.

# Die Technik hinter PCnewsGPT im Detail

Das hier Folgende ist eine konzeptionelle, technische Einführung, teilweise mit Wiederholung und Vertiefung des bisher Vorgebrachten. Mehr und auch Aktuelles gibt es online unter https://github.com/AndreasKunar/pcnewsgpt/tree/main#readme

## Auswahl der KI bzw. des LLMs

Als Programm für unsere KI wird llama.cpp (<https://github.com/ggerganov/llama.cpp>) verwendet. Es ist als Open-Source Lösung frei verfügbar und ermöglicht ein schnelles, lokales Ausführen verschiedener KI-Modelle (LLMs). Viele dieser Modelle/LLMs sind unter <https://huggingface.co/TheBloke> verfügbar (Tipp: als Dateityp/Modelltyp benötigen wir Quantisiertes GGUF-Format). Für PCnewsGPT brauchen wir Modelle, die auch gut deutsch sprechen, das sind z.B. für Internationale Anwendungen feinabgestimmte LLaMa-2 basierte „chat“ Modelle.

Llama.cpp arbeitet auf modernen „normalen“ Rechnern, ohne einen KI-Supercomputer zu benötigen – z.B. bei mir auf einem MacBook Air M2 Laptop mit mindestens 16GB RAM. Die KI-Modellgröße bestimmt den Speicherbedarf des Rechners und die Ausführungszeit, d.h. ein 7B oder 13B Modell (eines mit 7 oder 13 Milliarden Parametern) ist ein guter Kompromiss für moderne Rechner. Wichtig ist auch, dass unser verwendetes Modell zum Speichersparen komprimiert ist, eine „4-Bit Quantisierung“ (genauer vom Typ: Q4\_K\_S) ist ein guter Kompromiss zwischen Komprimierung und Qualität.

Ich verwende Python als Programmiersprache – llama.cpp ist für Python-Programme über die Bibliothek llama-cpp-python verfügbar.

## Eine KI mit Spezialwissen anzulernen, macht wenig Sinn!

Wenn man an Spezialwissen und KI denkt, ist natürlich der erste Impuls dieses Wissen einfach laufend einer bereits allgemein vorangelernten KI wie llama-2 anzulernen. Das ist aber extrem aufwändig – es braucht dafür sehr gute Rechner um viele tausende Euros, und wochenlanges Lernen. Außerdem ist das Anlernen einer KI ein sehr wissensintensives, komplexes Thema.

Das Schlimmste an dieser Lösung ist aber, dass falls die KI eigentlich nichts zur Frage weiß, sie meist durchaus glaubwürdig klingende „fake news“ halluziniert. Halluzinationen mögen fürs Geschichten schreiben kein Problem sein, sie machen aber Wissensabfragen völlig unbrauchbar.

Damit wird die KI der PCnewsGPT nicht mit Wissen angelernt!

## Die Lösung – „Retrieval Augmented Generation“ (RAG) mittels KI

Was ist die bessere Alternative zum fachspezifischen Anlernen einer KI? Einer allgemeinen KI einfach zu einer spezifischen Fragestellung die inhaltlich zur Frage am besten passenden Materialien als Kontext mitzugeben. Damit liest/analysiert die KI dann diese Materialien, und verwendet sie zur Fragebeantwortung – eben Retrieval Augmented Generation (RAG). Aber wie geht das im Detail?

## Ein Maß für die Bedeutung von Texten – Embeddings

Die aktuellen KI Trends haben eine wesentliche Innovation als Seiteneffekt mitgebracht, sogenannte „Embeddings“. Diese Embeddings sind eine Umwandlung von Texten auf große Gleitkommazahlenvektoren (z.B. für openAI’s text-embedding-ada-002 sind es 1‘536 dimensionale Vektoren). Diese Vektoren repräsentieren die Bedeutung des Inhaltes des Textes.

Ein Bild, das Text, Reihe, Screenshot, Diagramm enthält.

Automatisch generierte Beschreibung

Abbildung 1: Embeddings (Quelle: openAI)

Die Vektoren für inhaltlich ähnliche Texte sind sich dabei mathematisch näher als die von Bedeutungsfremden. Technisch erfolgt diese Nähe-Berechnung der Vektoren meistens über mathematische „cosine simmilarity“.

Ein Bild, das Diagramm, Reihe, Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 2: Ähnlichkeit und Embedings (Quelle: YouTube “What's wrong with LLMs and what we should be building instead” - Tom Dietterich)

## Vektordatenbanken – über Embedding-Vektoren indizierte Datenbanken

Vektordatenbanken eignen sich perfekt für die Wissensspeicherung und -abfrage. Diese bieten dabei eine Wissensspeicherung ohne die bisher nötige und übliche manuelle Klassifizierung der Inhalte. Für jedes Wissenselement als Datenbanksatz wird automatisch ein Embeddingvektor erzeugt. Die Wissensdatenbank liefert dann bei Abfrage eine gewisse Maximalanzahl von zur Abfrage passenden, „nahen“ Inhaltsdatensätzen. Dies erfolgt technisch über Distanzberechnung zwischen den Embeddingvektoren der Frage zu den Embeddingvektoren der Inhaltsdatensätze, und Auswahl der nähesten.

Damit wird’s einfach, passende Materialien zu einer KI-Frage zu finden und ihr als Wissenskontext mitzuliefern. Vorausgesetzt, es gibt eine passende Vektordatenbank mit Wissen.

Die Embeddingmodelle für die Vektordatenbank sind unabhängig vom KI-Modell/LLM. Die Embeddings müssen gut Bedeutungen analysieren können, das LLM muss gute Antworten formulieren. D.h. das LLM bekommt Klartext und keine Embeddingvektoren als Fragestellung und wandelt diesen dann bei der Verarbeitung in seine eigenen Embedding um.

## Wissenszerstückelung (chunks)

Leider gibt’s dabei eben eine weitere, fundamentale Einschränkung - KI-Fragen und Antworten sind insgesamt in der Länge limitiert. Damit können in KI-Fragen keine großen, kompletten Zeitschriftenartikel, oder lange Dokumente als Kontext mitgegeben werden. Diese Artikel oder Dokumente müssen inhaltlich in unabhängige Fragmente (chunks) zerstückelt werden.

Ein wenig technischer Detailhintergrund: Wenn die sog. „Maximum Context Length“ einer KI für die Frage-, Kontext- und Antwortkette überschritten wird, liefert die KI keine Fehlermeldung, sondern sie „vergisst“ dann bloß alles Ältere und berücksichtigt dieses nicht mehr für die Antwort. Die Antwort wird damit seltsam und wahrscheinlich falsch. Erschwerend ist dabei, dass diese Maximum Context Length keine exakt definitive Textlänge ist, sondern sich nur auf die Anzahl von sog. Tokens bezieht. Intern versteht die KI nämlich nur Zahlen, eben diese Tokens. Tokens sind damit verschieden lange Textelemente, welche die KI intern sofort in einzelne Token-Zahlen umwandelt – diese werden für das Ergebnis dann auch wieder von Token-Zahlen zurück auf lesbaren Text gewandelt. Die Maximum Context Length einer KI ist von der spezifischen KI abhängig – typischerweise ist es für uns max. 4096 Tokens. Ein guter Richtwert für eine durchschnittliche Token Textlänge ist 4 Zeichen, damit sollte die gesamte Frage+Kontext+Antwortkette nie länger als 16‘000 Zeichen sein.

Das Zerstückeln muss auch vorsichtig angewendet werden – wenn Sätze abgeschnitten werden, verändert sich die Bedeutung. Es gibt vorgefertigte „Text-Splitter“, welche Texte nach verschiedenen Kriterien zerstückeln – z.B. spaCy’s Sentence Boundary Detection (<https://spacy.io>) ist ein recht guter, multilingualer Text-Splitter. Dem Splitter gibt man zumeist vor, wie groß die einzelnen Stücke (in etwa) sein sollen, und ob sie ggf. überlappend sein sollen.

## Langchain – der Werkzeugkasten

Langchain ist ein Python-basiertes Open Source Framework zur Orchestrierung von KI Anwendungen. Es ermöglicht einen Programmablauf unabhängig von den Details der verwendeten Werkzeuge. Und auch unabhängig, ob die Verarbeitung in der Cloud oder auf der lokalen Maschine oder gemischt erfolgt.

Mittels langchain ist es möglich, ein KI Python Programm in wenigen Codezeilen zu erstellen.

## PCnewsGPT – der Prototyp

PCnewsGPT ist so aufgebaut, dass außer dem initialen Herunterladen von Bibliotheken, Modellen, … die komplette Verarbeitung ausschließlich auf dem lokalen Computer erfolgt. Es besteht im Wesentlichen aus zwei Programmen.

### Die PC-News Wissensdatenbank befüllen – Programm import.py

Dieses Programm durchsucht ein Verzeichnis/Ordner auf dem Computer nach Dokumenten (PDFs) und verarbeitet diese. Dabei werden diese Dokumente gelesen und in Text umgewandelt (PDF auf Text mittels „PyMuPDF“). Diese Textdaten werden dann mittels „SpaCy“ zerstückelt und mit Attributen wie Textelementinhalt, Dateiname, Position und Quelldatum versehen. Für jedes Textelement wird auch ein zugehöriger Embeddingvektor mittels „HuggingFace/SentenceTransformers“ erzeugt. Jedes Element wird mit allen Informationen in einer „chroma“ Vektordatenbank gespeichert.

Import.py hat dabei zwei Betriebsarten, den Initialimport mit dem Erzeugen einer neuen Vektordatenbank, falls keine vorhanden ist, und dem Hinzufügen von weiteren Dokumenten zu einer bereits bestehenden Vektordatenbank.

### Die PC-News KI Abfragen – Programm abfrage.py

Mit diesem Programm erfolgt eine KI-Abfrage der Wissensdatenbank. Dies erfolgt per lokalem LLaMa-basiertem KI Modell und der Vektordatenbank mit den Embeddings. In abfrage.py ist auch ein Berücksichtigen des Zeitbezuges der wichtigsten Wissensfragmente, zusätzlich zur Embedding-Distanz, implementiert.  
  
Die Eingabe der eigentlichen Anfrage an das LLM sieht dabei im Text wie folgt aus:

*Es folgt eine Liste von Informationen, die wichtigsten sind am Ende:   
{hier werden die passenden Kontextinformationen aus der Vektordatenbank eingesetzt, aufsteigend nach Wichtigkeit sortiert} Ant­worte auf die folgende Frage nur mit diesen Informationen:* {hier wird die Benutzerfrage eingesetzt} Antwort:

Das LLM analysiert diese Anfrage, und liefert eine dazu passende Antwort. Zumeist ohne Halluzinationen.

Weil die initiale Erkennung der doch längeren Anfrage durch das LLM meist länger dauert, erfolgt nach der Eingabe zunächst auch eine Zeitabschätzung für die Wartezeit. Zu Testzwecken kann abfrage.py auch die genauen verwendeten Kontextinformationen mit ausgeben.

Ich hoffe dieser Artikel lieferte einen brauchbaren Überblick über das sich sehr dynamisch Entwickelnde Thema KI-Wissensabfragen, und wünsche auch noch viel Spaß beim Experimentieren damit!