BACKEND

Backend

Mit *[sci]mmary* möchten wir öffentlich verfügbare wissenschaftliche Publikationen auf unserer Plattform in Form eines eigenen Templates bereitstellen. Hierfür müssen zunächst verfügbare Originalpublikationen im Internet ausfindig gemacht werden. Daraufhin müssen sie auf ihre Qualität untersucht und für die Weiterverarbeitung in unserer Datenbank gespeichert werden. Befinden sich diese Originalpublikationen erst mal in der Datenbank, so müssen sie anschließend Filterschleifen durchlaufen. Diese ermöglichen, Text- und Bildinhalte in kleinste Einzelkomponenten zerlegen zu lassen, um sie darauffolgend in unsere vordefinierten Templates einfügen und gegebenenfalls neu arrangieren zu können. Angesichts unterschiedlicher Informationstiefen, die wir für Texte und Bilder anbieten wollen, müssen diverse Formulierungen und Darstellungen von Texten und Abbildungen noch zusätzlich berechnet werden.

Abbildung 151: [sci]mmary und künstliche Intelligenz [Quelle: Eigene Darstellung]

Involvierung einer KI

Warum brauchen wir eine KI?
Für einen reibungslosen und automatisierten Ablauf arbeitet *[sci]mmary* mit verschiedenen Techniken der künstlichen Intelligenz, die sich im Bereich des Machine Learnings verorten lassen. Auf diese Weise sollen vor allem neue Studien einen schnellen Weg in unsere Datenbank finden und so rasch wie möglich verarbeitet werden können. Dieses Vorgehen ist besonders wichtig, wenn in krisenbedingten Situationen (z. B.: die *Covid-19-Pandemie*, Naturkatastrophen etc.) eine hohe Menge an fundierten Informationen für die Bevölkerung zeitnah bereitgestellt werden sollen.

GPT-3 – Die zukünftige Text-Bildsynthese
Wir haben uns mit dem Thema „Text-Bild-Synthese" und der aktuellen AI-Revolution namens *GPT-3* beschäftigt. *GPT-3* ist ein neuronales Netzwerk und ein autoregressives Language Model, das von *OpenAI* entwickelt wurde (vgl. Kilian Visuals3D 2020). Sie ist in der Lage, Texte zu verfassen, Texte durch berechnete Predictions (dt. Vorhersagen) zu ergänzen, Bilder zu generieren und Fragen zu beantworten. Zudem ist sie aktuell das größte Language Model, das bis zu 175 Milliarden Parameter besitzt und mit den größten Datensätzen der Welt trainiert wurde (vgl. Code mit FloW 2020). Seitdem das Paper (Link zum Paper: Brown 2020) für *GPT-3* Anfang 2020 veröffentlicht wurde, hat sich einiges getan. Im Januar 2021 wurde das Projekt *Dall:e* veröffentlicht, das beweist, wie gut diese KI mittlerweile in der Lage ist, Text-Bild-Synthesen durchzuführen (vgl. OpenAI 2021). Neben zweidimensionalen Grafiken ist sie imstande, dreidimensionale Modelle aus verschiedenen Perspektiven zu erstellen. Außerdem wird ein API (dt. Programmierschnittstelle) von *GPT-3* angeboten, um eigene Projekte realisieren zu können. Jedoch muss man sich dafür zunächst auf eine Warteliste eintragen, da die Schnittstelle aktuell nur für ausgewählte Testnutzer zugänglich ist (vgl. Code mit FloW 2020).

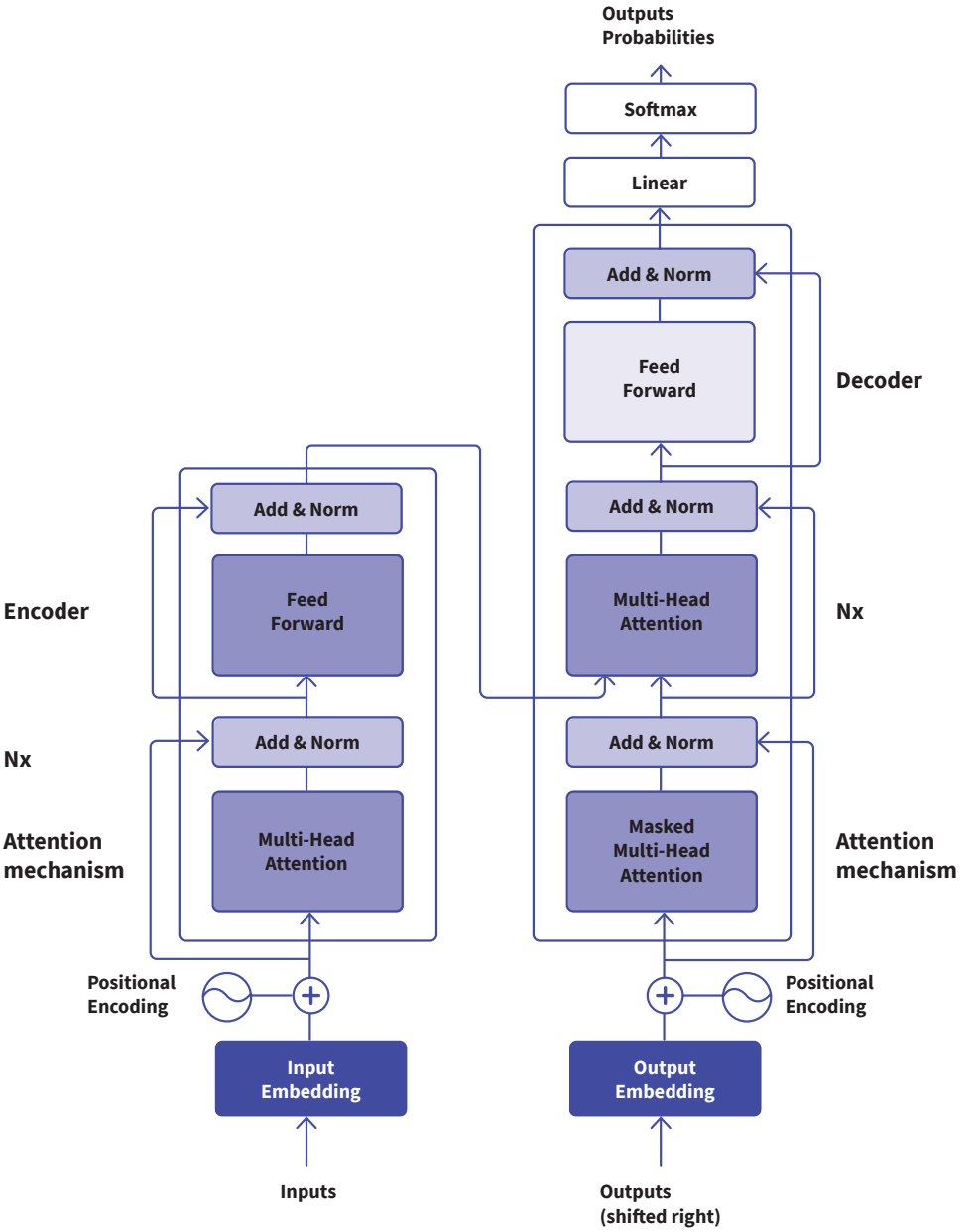

Abbildung 152: GPT-3-Modell
[Quelle: in Anlehnung an Ray 2020]

Diese Technologie wollen wir uns zunutze machen, um eine mögliche Realisierbarkeit von [sci]mmary aufzuzeigen und belegen zu können. Daher haben wir uns Gedanken zum Backend von [sci]mmary gemacht und eine mögliche Backend-Infrastruktur entwickelt, die den ungefähren Ablauf des Systems demonstrieren soll.

So funktioniert [sci]mmary

Vorarbeit – Start
Bevor eine wissenschaftliche Publikation nach unseren Vorgaben überhaupt verarbeitet werden kann, muss ein Datensatz von der Publikation erstellt werden. Davor muss die Publikation gefunden, kopiert, in ihre Einzelteile zerlegt werden, um sie anschließend in unserer Datenbank abspeichern zu können. Da mittlerweile immer mehr Studienplattformen Open Access-Publikationen bereitstellen, können diese einfach durch unseren selbst programmierten Searchbot (Crawler - Erläuterung dazu s. Nächster Abschnitt) aufgespürt und durch einen Scraper heruntergeladen werden. Diese Kopien werden dann vom Scraper an die Datenbank weitergeleitet, wo diese in weiteren Verarbeitungsschleifen in ihre Einzelteile zerlegt und schlussendlich in unserem Template neu zusammengefügt werden. Um diese Aufgaben bewältigen zu können, brauchen wir folglich mehrere technische Komponenten, die diesen Prozess unterstützen.

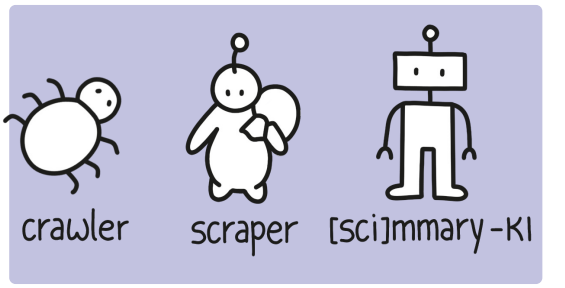

Abbildung 154: Crawler, Scraper, [sci]mmary
[Quelle: Eigene Darstellung]

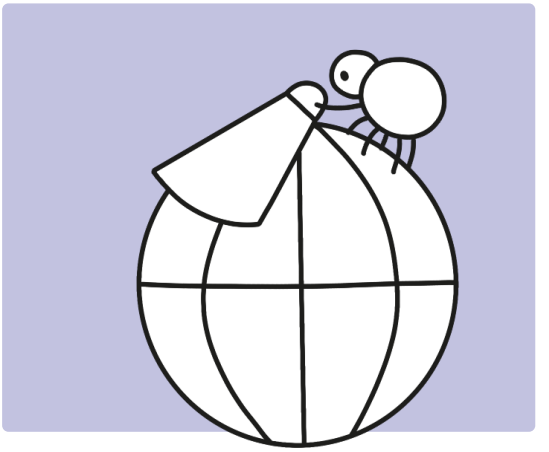

Abbildung 153: Vorarbeit
[Quelle: Eigene Darstellung]

Crawler
Ein Crawler ist ein programmierter Searchbot, der dabei hilft, in allen möglichen wissenschaftlichen Publikationsdatenbanken verfügbare und zugängliche Publikationen zu finden. Da wir hauptsächlich an wissenschaftlichen Publikationen interessiert sind, nutzen wir einen sogenannten Focused Crawler. Das sind Crawler, die für einen festgelegten Themenbereich arbeiten (in unserem Fall: verfügbare wissenschaftliche Publikationen). Der Crawler wird zusätzlich mit Tags wie beispielsweise „Open Access" versehen, die als zusätzliche Filter bei der Suche helfen sollen, zwischen nicht öffentlich und öffentlich zugänglichen Publikationen zu unterscheiden.

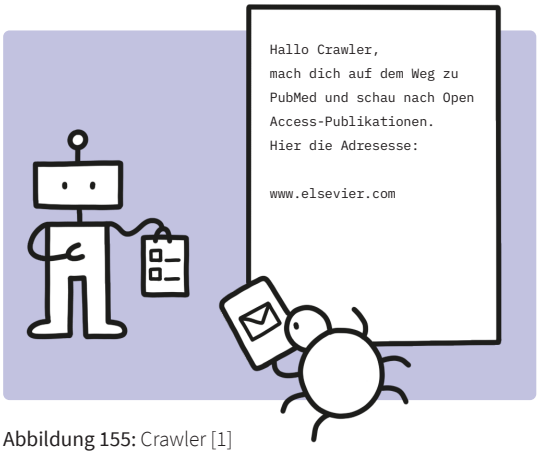

Abbildung 155: Crawler [1]
[Quelle: Eigene Darstellung]

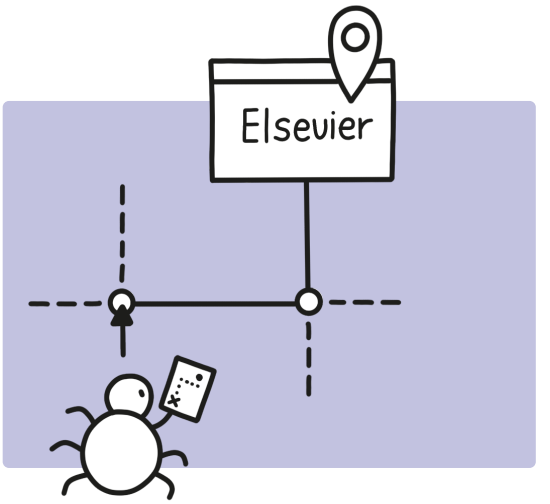

Abbildung 156: Crawler [2]
[Quelle: Eigene Darstellung]

Im nächsten Schritt wird der Crawler von uns mit einer URL als Zielort (Target) auf eine Plattform geschickt (zum Beispiel Elsevier mit der URL: http://www.elsevier.com/), welche von ihm gründlich durchsucht wird. Diese Plattform ist wiederum mit anderen Plattformen durch weitere Hyperlinks verbunden. Das bedeutet auch, dass Publikationen mehrmals auf verschiedene Plattformen existieren können. Der Crawler kann wie eine Spinne auf einem Spinnennetz (hier: World Wide Web) sich von einem Link zum nächsten fortbewegen und automatisch nach Duplikaten und weiteren potenziellen Publikationsplattformen suchen.

So braucht es kaum neue Targets, da er automatisiert nach weiteren Publikationsplattformen suchen kann. Diese ganzen Links werden nach dem Durchlesen in einem Zwischenspeicher (Cache) gespeichert, ausgewertet und auf der Datenbank indexiert. Auf diese Weise kann der Crawler besuchte Links in seinem Gedächtnis (hier: Index) speichern und in Zukunft immer direkt wieder besuchen. Außerdem kann er regelmäßig kontrollieren, ob neue Publikationen auf einer Plattform erschienen sind oder ob eine schon in der Datenbank abgespeicherte Publikation verändert wurde. Somit kann der Crawler eine veränderte Publikationsdatei mit der gespeicherten Publikation aus der Datenbank immer wieder abgleichen und gegebenenfalls ersetzen.

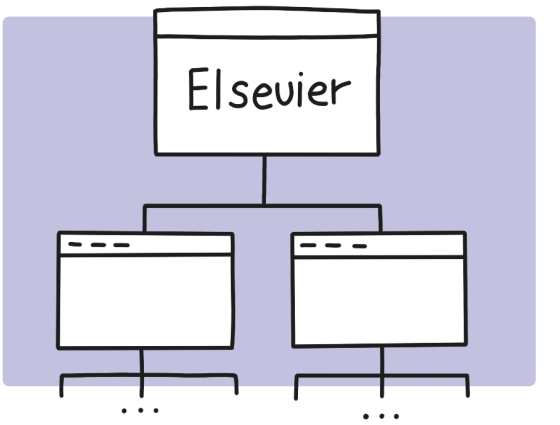

Abbildung 157: Crawler [3]
[Quelle: Eigene Darstellung]

Scraper
Der Scraper wird oft mit dem Crawler verwechselt. Dabei unterscheiden sie sich in ihren Funktionen stark voneinander. Während der Crawler nur suchen, lesen und indexieren kann, verfügt ein Scraper sowohl über manuelle als auch automatisierte Downloadunktionen.

Der Crawler kann eine ganze Index-Liste zusammenstellen und diese an den Scraper übermitteln, der sich dann auf dem Weg macht, die aufgelisteten Webseiten zu besuchen.

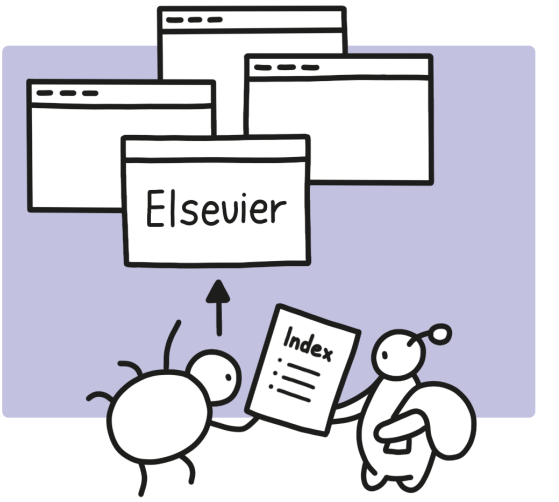

Abbildung 158: Scraper [1]
[Quelle: Eigene Darstellung]

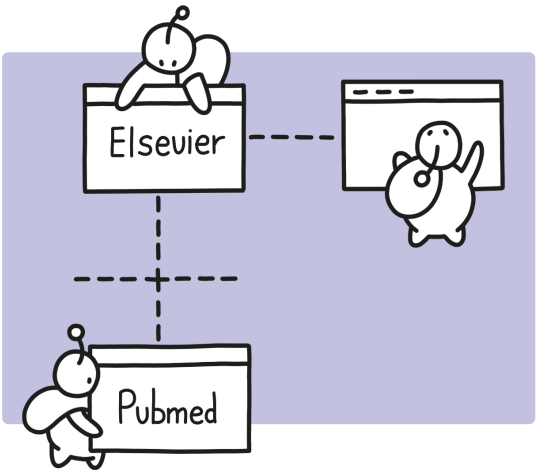

Abbildung 159: Scraper [2]
[Quelle: Eigene Darstellung]

Der Scraper besucht daraufhin alle aufgelisteten Webseiten, lädt diese entsprechenden Publikationen herunter, kopiert diese und leitet diese dann weiter an die *[sci]mmary*-Datenbank, wo sie dann weiterverarbeitet werden kann.

Abbildung 160: Scraper [3]
[Quelle: Eigene Darstellung]

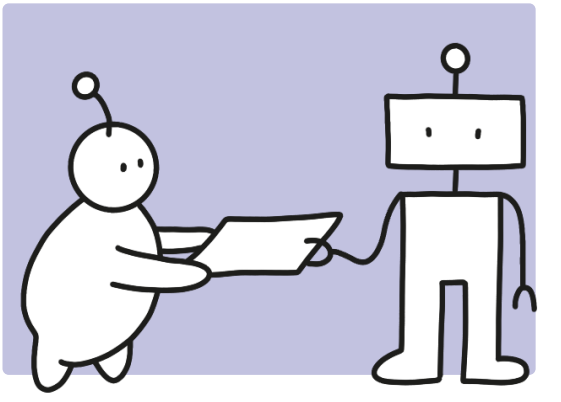

Abbildung 161: Scraper [4]
[Quelle: Eigene Darstellung]

Scraper VS API
Manchmal kann ein Scraper umgangen werden, wenn Plattformen freiwillig eine API zur Verfügung stellen. Eine API ist ein Zugang zu Plattformen, die von sich aus die Möglichkeit geben, Einsicht in ihre Daten zu gewähren. Da dies nicht immer der Fall ist und bestimmte APIs nur einen begrenzten Zugang ermöglichen, muss ein Scraper programmiert werden.

Es gibt aber noch einige weitere Voraussetzungen, die man beachten muss, wenn man Scraping technisch und legal durchführen möchte. Beispielsweise muss die Webseite lesbar sein, damit sie überhaupt durchgescannt werden kann. Auch darf im Quellcode das Dokument „robot.txt" für Bots nicht blockiert werden. Darüber hinaus muss beachtet werden, wie oft ein Bot auf eine Seite zugreifen darf. Diese "conservative crawl rate" (Beispiel: 1 Request / pro 10 Sekunden) muss daher angepasst werden. In unserem Fall müsste man schauen, wie oft pro Sekunde eine neue Studie veröffentlicht wird und wie hoch die Wahrscheinlichkeit ist, dass eine Studie sich mehrmals verändert. Am sichersten ist es, wenn Webseiten eigene APIs zur Verfügung stellen. Möglich ist aber auch, sich die AGB der einzelnen Plattformen durchzulesen.

Dort beschreiben sie, welche Grenzen Plattformanbieter aufziehen und was es zu respektieren gilt. Wir als *[sci]mmary* können unsere Bots mit sogenannten „agent strings" versehen, in denen gespeichert wird, dass wir die Daten mit guten Absichten holen. Dort könnten wir auch erklären, dass wir legal arbeiten, einen gesamtgesellschaftlich relevanten Zweck erfüllen wollen und nicht beabsichtigen, Paywalls (dt. Bezahlmechanismen) zu durchbrechen, wie dies zum Beispiel die Plattform SciHub macht.

Sind diese Daten erst einmal kopiert und in unserer Datenbank hinterlegt, beginnt der nächste Schritt - die Qualitätssicherung.

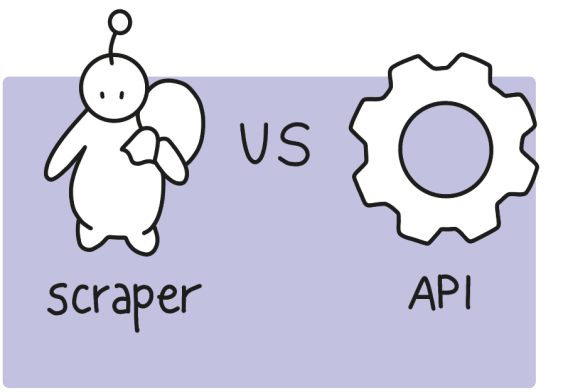

Abbildung 162: Scraper vs. API
[Quelle: Eigene Darstellung]

Qualitätssicherung
An dieser Stelle folgen Kontrollen der kopierten Publikation. Sie dienen der Qualitätssicherung, da sie bestimmte Qualitätskriterien erfüllen müssen, um überhaupt von uns veröffentlicht zu werden. Dazu gehören beispielsweise die korrekte und saubere Anwendung von wissenschaftlichen Methoden, die Offenlegung von Finanzierungsquellen bzw. Interessenkonflikten und einige andere Kriterien. Damit möchten wir verhindern, dass unsere Plattform Predatory Publishing unterstützt.

Darauf wird die Publikation zum Parser geschickt. Seine Aufgabe ist es, sich um „die Zerlegung und Umwandlung des Inputs (in unserem Fall vorhandene Textdateien und mögliche Grafiken) in ein brauchbares Format für die Weiterverarbeitung" (Ryte Wiki (o. J.) zu kümmern. Parser werden oft verwendet, um „einen Text in eine neue Struktur" (Ryte Wiki o. J.) überführen zu können. Das kommt uns sehr entgegen, da wir wissenschaftliche Publikationen neu gestalten möchten. Um einen Text eine neue sinnvolle Struktur geben zu können, muss zunächst der Parser mithilfe eines Lexers eine lexikalische Analyse durchführen. So werden aus dem Text geeignete Einzelkomponenten (Tokens) erstellt, die darauffolgend in einer sinnvollen Hierarchie aufgelistet werden.

Als nächstes folgt eine syntaktische Analyse, in der der Parser Regeln und Logiken erstellen kann (z. B.: in Form eines Syntaxbaumes). Diese werden für die Weiterverarbeitung benötigt, damit Einzelkomponenten mit ihren entsprechenden Regelungen an den Compiler, der aus Einzelkomponenten eine überarbeitete Komposition ableitet und dafür einen neuen Code generiert, weitergeleitet werden können.

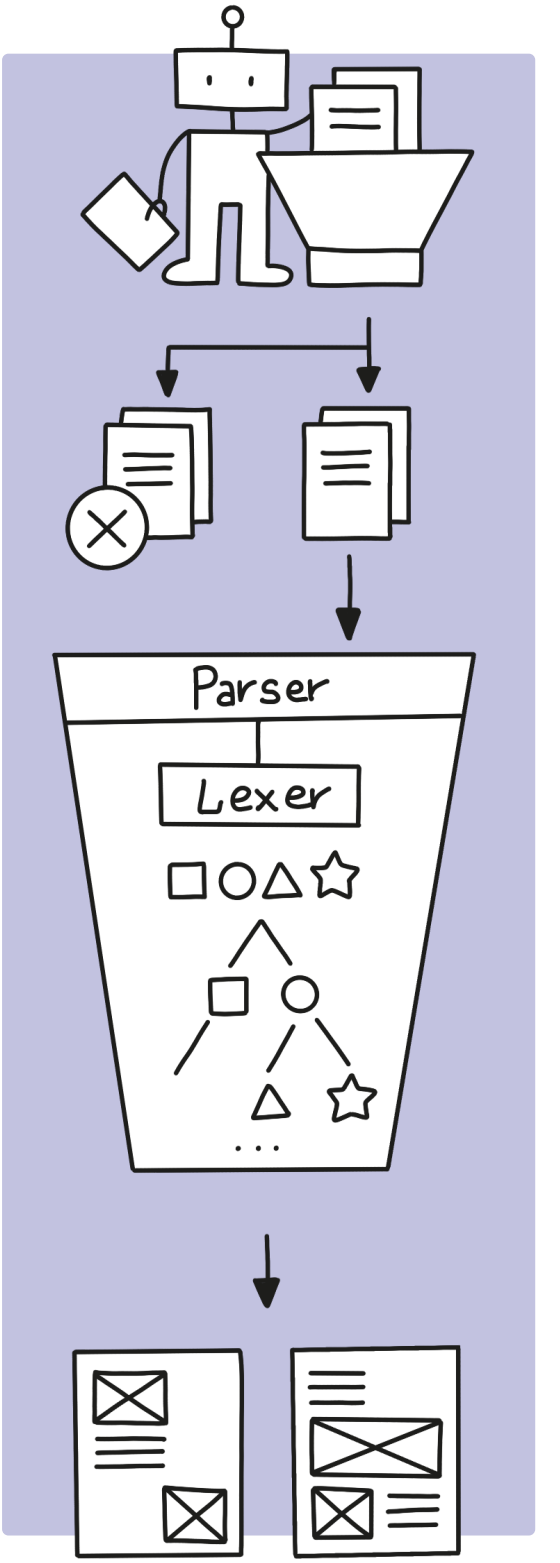

Abbildung 163: Qualitätssicherung
[Quelle: Eigene Darstellung]

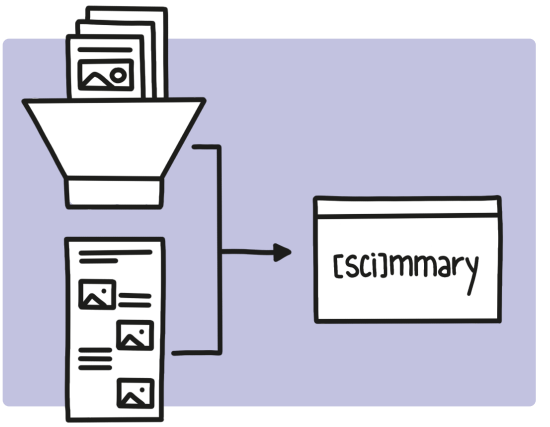

Abbildung 164: Template
[Quelle: Eigene Darstellung]

Danach werden die Einzelkomponente in bestimmte Kategorien (z. B.: Einleitung, Ergebnisse etc.) in unsere Vorlage eingefügt. Für jeden Publikationstyp gibt es ein eigenes Muster. Denn eine klinische Studie muss beispielsweise anders als ein systematisches Review dargelegt werden. Dadurch soll die wissenschaftliche Korrektheit beibehalten werden. Damit ist der Vorgang aber noch nicht abgeschlossen, denn es müssen noch Alternativen für unterschiedliche Informationstiefen generiert werden.

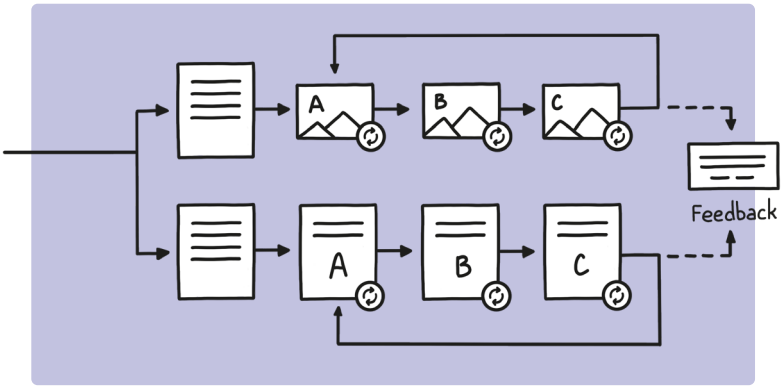

Abbildung 165: Refresh Funktion
[Quelle: Eigene Darstellung]

Closed Loop

Unser System ermöglicht ein geschlossenes Kreislaufsystem, dass wir „Closed-Loop" nennen. Das bedeutet, dass ein Text zu einem Bild und umgekehrt auch ein Bild in einen Erklärtext umgewandelt werden kann. Zudem ist es möglich, einen komplexen Text zu einem einfacheren und kürzeren Text zusammenzufassen. Darüber hinaus können alternative Vorschläge für Texte und Bilder generiert werden, die es ermöglichen, auf verschiedene Informationstiefen der Nutzer einzugehen. Mit einem Klick auf die Schaltfläche "Aktualisieren" (Refresh-Button) wird ein alternatives Bild aus der Datenbank geschickt, das bei der Bearbeitung der wissenschaftlichen Publikation von der KI vorbereitet wurde. Gleichzeitig trackt unser System, wie oft der Benutzer diese Schaltfläche zum Aktualisieren angeklickt hat. Um im Nachhinein Schlüsse ziehen zu können, ob die vorberechnete Einstellung der Informationstiefen für den Nutzer in diesem Wissenschaftsbereich angemessen war. So verbessert sich das Trainingsmodell mit der Zeit und die personalisierten Berechnungen können immer akkurater werden.

Text- und Bildsynthese

Text zu Text
Nachdem unser System die einzelnen Textbestandteile klassifiziert, organisiert und die Quintessenz herausgelesen hat, wird die Kernaussage zusammengefasst und leicht verständlich dargestellt. Mithilfe der künstlichen Intelligenz können wir Originaltexte dem Wissensstand der Leser anpassen lassen. Ist in der Anfangsphase die Einschätzung nicht akkurat genug, wäre es möglich, mittels einer Schaltfläche „Aktualisieren" (Refresh-Button) die Textstelle neu paraphrasieren zu lassen. Es ist mehrmals möglich, diese Schaltfläche zu drücken, solange bis das Herunterbrechen ausgeschöpft ist.

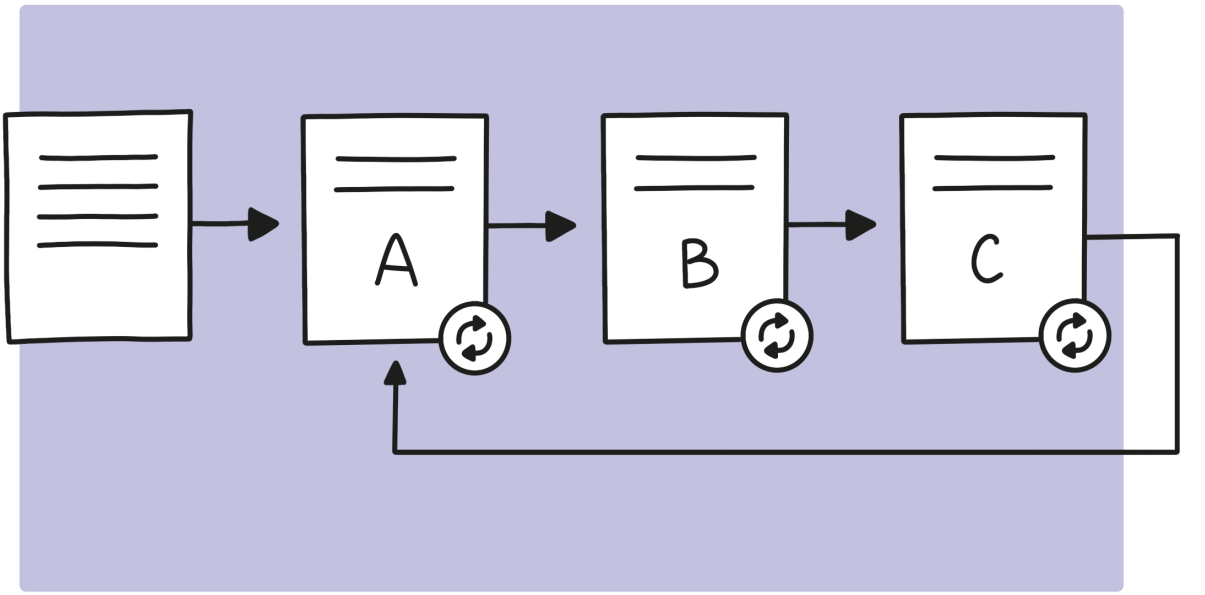

Abbildung 166: Text zu Text
[Quelle: Eigene Darstellung]

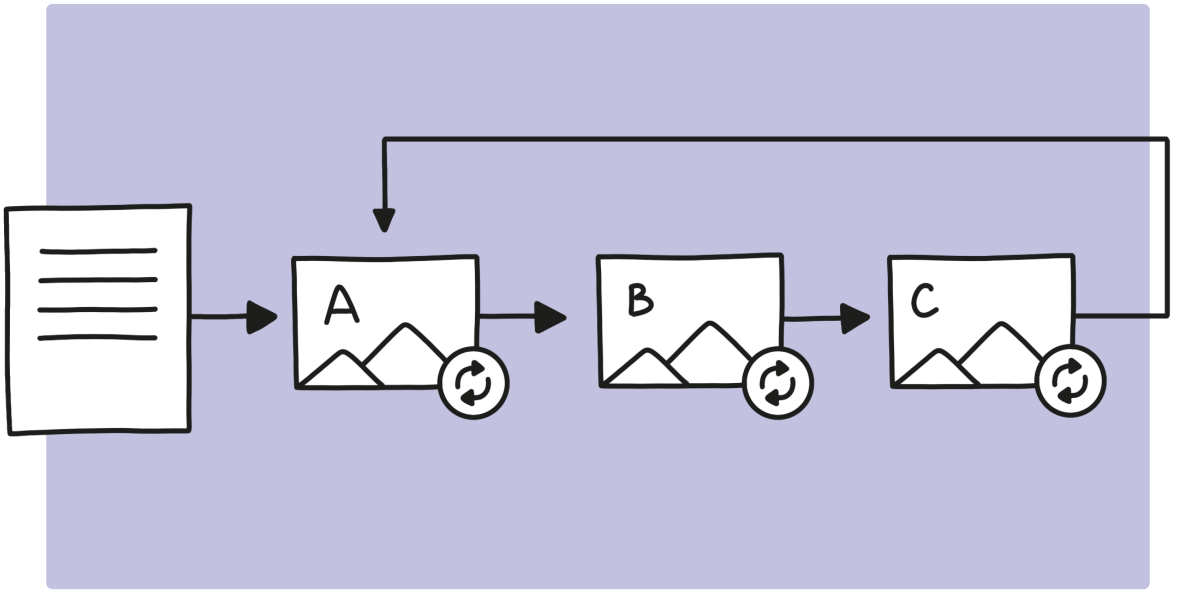

Abbildung 167: Text zu Bild und Bild zu Bild
[Quelle: Eigene Darstellung]

Text zu Bild / Bild zu Bild

Ein Text kann auch in ein Bild umgewandelt werden. Das hat den Vorteil, dass Menschen komplexe Informationen auch visuell vorgestellt und komplizierte Texte umgangen werden können. Durch die Fragmentierung des Textes in einzelne Komponenten werden sie beim Parsing zusätzlich hierarchisch angeordnet. Diese hierarchische Struktur wird zur Einteilung des Vorder-, Mittel- und Hintergrundes genutzt. Diese werden dann zusammengesetzt und so ergibt sich ein Bild. Ähnlich wie im „Text zu Text"-Beispiel können Bilder auch mehrmals heruntergebrochen und variiert werden, bis auch hier die Bilderbibliothek dafür ausgeschöpft ist.

Die bereits vorhandenen Grafen aus den wissenschaftlichen Publikationen sind oft sehr mathematisch und komplex gestaltet, daher ermöglichen wir eine Vereinfachung zu einem reduzierten Bild, das weniger statistisch ist und intuitive Darstellungsweisen anbieten kann. Auch hier können die erzeugten Alternativen ausgeschöpft werden.

Ist die Bibliothek irgendwann ausgeschöpft, kann das für den Nutzer frustrierend sein, daher muss auch hier ein Fallback eingeplant werden. Am Ende könnte ein Popup-Fenster erscheinen und nach einem Feedback dazu fragen. Der Nutzer kann dort beispielsweise mitteilen, was ihm gefehlt hat, um den Inhalt des Bildes besser verstehen zu können. Auf diese Weise sollte der Algorithmus verbessert und Schritt für Schritt ein positiveres Nutzererlebnis ermöglichen.

Herausforderungen und Lösungsansätze

Mögliche Herausforderungen
Neben den einzelnen Features gibt es einige Herausforderungen, die in unserem Konzept mitbedacht werden müssen: Wir haben mehrere Szenarien ausgearbeitet, die dem Nutzer bei der Benutzung unserer Plattform im Weg stehen könnten. Passend dazu haben wir mehrere Lösungsstrategien entwickelt, die ein mögliches Problem umgehen könnten. Diese Vorsichtsmaßnahmen haben wir als Fallback bezeichnet.

Halbsätze markieren
Der Nutzer kann Textstellen mit einem Marker aus unserer Toolbox markieren und diese in ein Bild umwandeln lassen. Problematisch kann es werden, wenn der User willkürlich Halbsätze markiert, woraus kein sinnvolles Bild zusammengestellt werden kann. Daher haben wir als Fallback die Idee entwickelt, dass der Algorithmus schon vorher im Hintergrund festgelegt, wo welche Bilder im Text erzeugt werden können und diese mit einem Symbol versieht. Klickt man auf das Icon, erscheint das Bild auf einem Pop Up-Screen. Alternativ kann, um Platz zu sparen und die Anzahl der Icons zu reduzieren, das klassische Textmarkieren angewendet werden. Sobald ein ausreichender Anteil des Satzes markiert wurde, erscheint ein Tooltipp, das in Kombination mit einem Symbol zeigt, dass das Bild bereit ist, generiert zu werden. Der Nutzer kann die Markierung jetzt loslassen. Hier erscheint dann ein Bild als Pop Up-Screen.

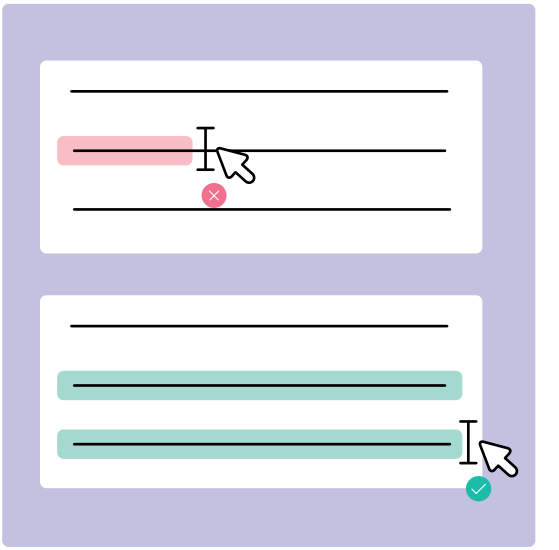

Abbildung 168: Herausforderung - Markierung
[Quelle: Eigene Darstellung]

Informationslevel
Unser System passt den Inhalt der Publikation nach einem bestimmten Informationslevel an. Sie gibt an, wie weite komplexe Informationen verarbeitet und heruntergebrochen werden müssen, wie viel zusätzlicher Kontext für den Nutzer benötigt wird und welche Art von Bildern zum Nutzer passen könnte.

Zu Beginn haben wir erst fünf verschiedene Informationslevel. Mit der Zeit und der Weiterentwicklung der KI wird es eine feinere, granularere Einteilung geben, die durch eine Personalisierung automatisch den Inhalt gleich anpasst und viel “refreshen“ zu müssen.

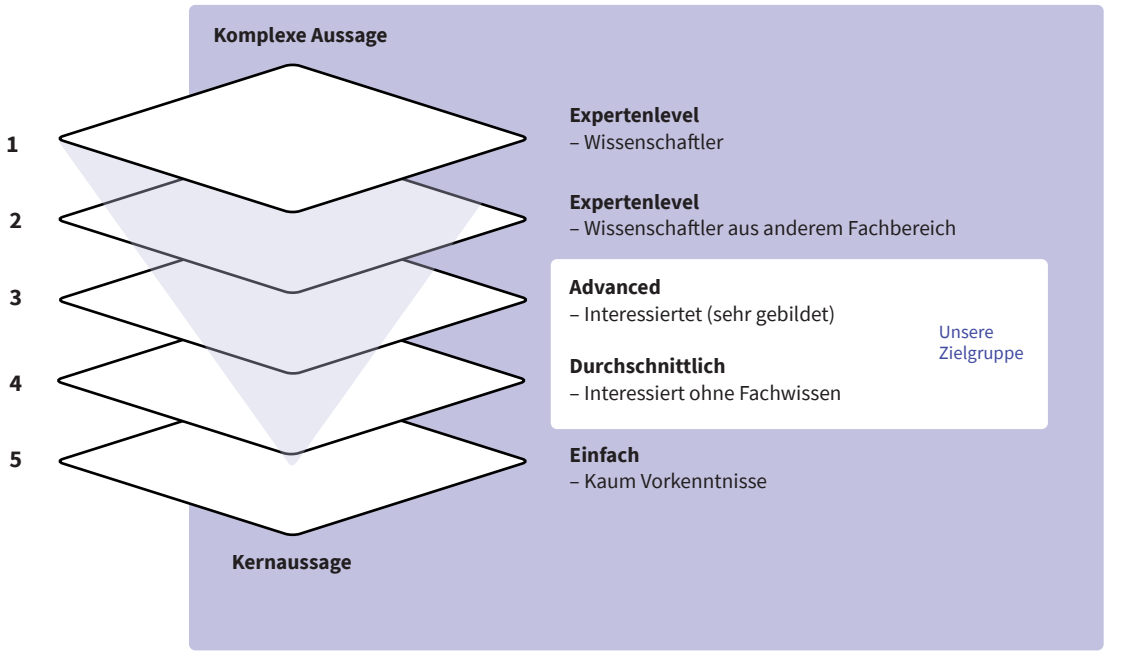

Abbildung 169: Kommunikationsebene [1]
[Quelle: Eigene Darstellung]

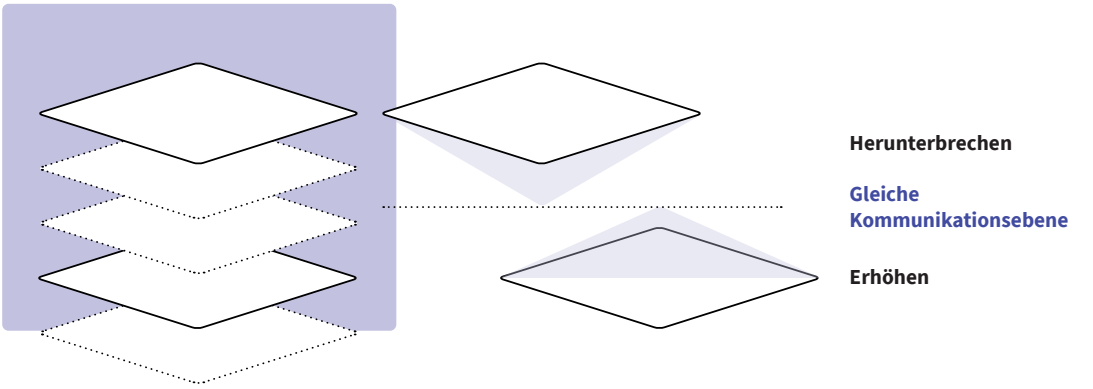

Abbildung 170: Kommunikationsebene [2]
[Quelle: Eigene Darstellung]

Kommunikationsebene
Viele wissenschaftliche Texte werden von Laien nicht verstanden. Ein Gründ dafür kann der unterschiedliche Informationslevel sein, in der sich die jeweiligen Gruppen befinden (z. B.: Wissenschaftler und Laie). Um sich gegenseitig verstehen zu können, müssen entweder Experten ihr Niveau senken und versuchen, komplexe Sachverhalte herunterzubrechen oder Laien müssen versuchen, ihr Fachwissen schnell und mühsam aufzuholen. Das kann für beide Seiten sehr anstrengend sein.

[sci]mmary kann von allen möglichen Informationsniveaus aus den Inhalt so anpassen, sodass der Nutzer weder sein Informationsniveau senken noch krampfhaft erhöhen muss. Somit verstehen sich beide Parteien, ohne ihr Sprachniveau verlassen oder umstellen zu müssen.

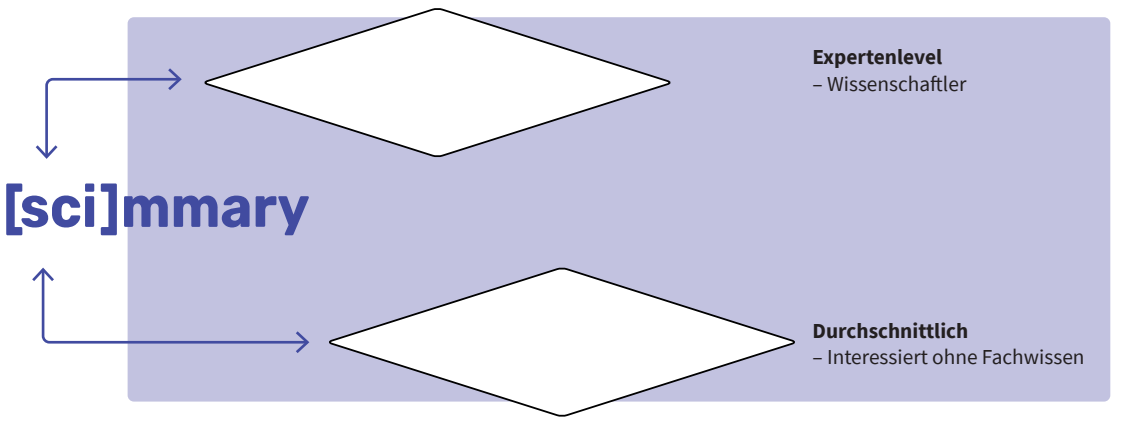

Abbildung 171: Anpassung Informationsniveau
[Quelle: Eigene Darstellung]

Geduld des Users

Zu Beginn wird der Algorithmus mehrere Anläufe brauchen, bis er bessere Vorhersagen über den Wissensstand des Nutzers treffen kann, um so ein ständiges Aktualisieren der Bilder oder Texte zu verhindern. Mit der Zeit wird der Algorithmus so gut sein, dass er automatisch das anzeigen wird, was zum Profil des Nutzers passt. Die Voraussetzung dafür ist aber, dass der Nutzer einen Account erstellt, damit seine Daten getrackt werden können. Als Extraleistung bekommt er nicht nur eine personalisierte Ansicht der Publikationen, sondern dazu noch eine Profilübersicht mit dem vom Algorithmus geschätzten Wissensstand in den einzelnen Wissenschaftsbereichen. Theoretisch könnte der Nutzer seinen Wissensstand in den Bereichen auch manuell selbst anpassen, die nachträglich vom Algorithmus hoch oder herunter korrigiert werden können.

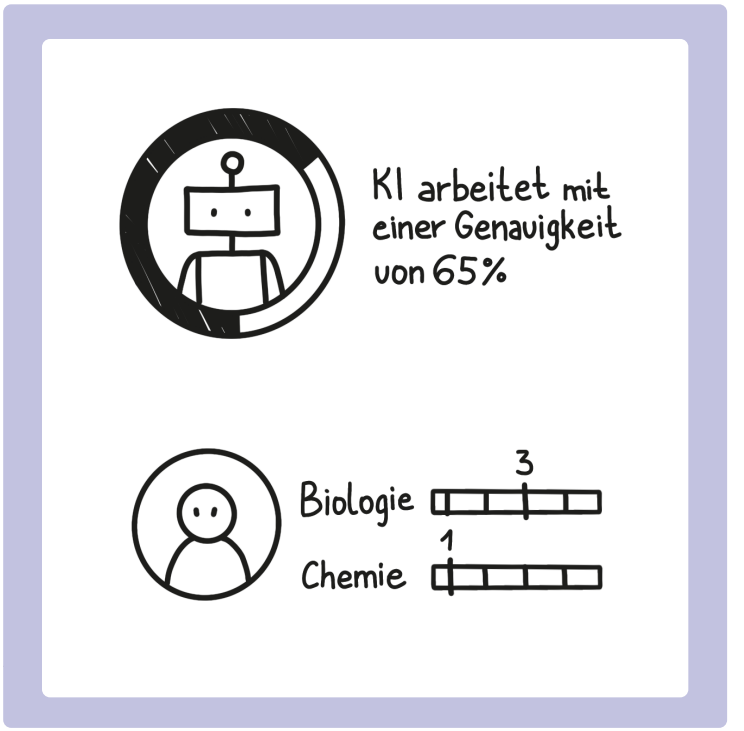

Abbildung 172: Verbesserung des Algorithmus
[Quelle: Eigene Darstellung]

Verbesserung des Algorithmus
Denkbar wäre, eine Übersicht im Profil zeigen zu lassen, die besagt, wie genau der Algorithmus bereits arbeitet oder mit zusätzlichen Feedback-Maßnahmen experimentieren. In regelmäßigen Abständen kann so unser System abfragen, ob der Nutzer zufrieden ist und welche Änderungen er sich noch wünscht.

Fazit
Zusammenfassend kann man sagen, dass es aktuell immer besser werdende Text-Bild-Synthese-Technologien gibt und die Realisier- und Machbarkeit unseres Projektes somit potenziell gegeben ist. Dennoch müssen weiterhin mehrere Fallback-Varianten eingeplant werden, bis stärkere und präzisere Algorithmen diese Sicherheitsmaßnahmen obsolet machen.