Universität Rostock

Traditio et Innovatio

Automatische Sprachübersetzung von LaTeX-Dokumenten

Name:	Hendrik Theede
Matrikelnummer:	221201256
Abgabedatum:	02.12.2025
Betreuer und Gutachter:	Prof. Dr. rer. nat. habil. Clemens H. Cap Universität Rostock Fakultät für Elektrotechnik und Informatik

Bachelorarbeit am Lehrstuhl für Informations- und Kommunikationsdienste

Fakultät für Elektrotechnik und Informatik

Abstrakt

placeholder

Inhaltsverzeichnis

1 Einleitung

Die schnellstmögliche und einfache Erstellung von Dokumenten beliebiger Natur (formlos) wird heutzutage oftmals über Produkte bekannter Anbieter abgewickelt (bspw. Microsoft's Word, PowerPoint, etc., die vergleichbaren „LibreOffice“, sowie Apple-Produkte). Unterliegen Dokumente allerdings strengeren stilistischen Vorgaben (bspw. bei wissenschaftlichen Abhandlungen) entsteht der Vorteil, dass sich diese Vorgaben wie ein Regelsatz behandeln lassen, aus welchem bestimmte, feste Dokumenten-Strukturen hervorgehen. Hierzu existiert bereits ein geläufiges System namens LaTeX (mit dem *La* nach einem der ursprünglichen Entwickler Lamport (1994)), welches selbst auf dem von Knuth (1986) entwickelten Zeichensetzungs-System und der verbundenen Programmiersprache TeX basiert. Die TeX-Syntax selbst basiert auf englischen Begriffen, allerdings ist nicht davon auszugehen, dass nur englischsprachige Menschen LaTeX und TeX nutzen werden. Quelltexte und Dokumentenbeschreibungen werden also nicht immer in einer rein englischsprachigen Form vorliegen (z.B. `\chapter{Erstes Kapitel: Einleitung}` oder der Quelltext dieses Werkes). Ein Zurückführen solcher Dokumente in die englische Sprache ist einfach, insofern ein Verständnis der deutschen Sprache besteht (`\chapter{First Chapter: Introduction}`). Die andere Richtung wirft allerdings eine Mehrzahl an Problemen auf, wenn ohne Vorkenntnisse von TeX (bzw. dessen syntaktische Elemente) übersetzt wird. In genanntem Beispiel würde dann das Wort `chapter` aufgegriffen werden und die Zeichenkette `\Kapitel{Erstes Kapitel: Einleitung}` entstehen (ohne weitere, jedoch mögliche Anpassungen entsteht hier keine Kapitelüberschrift mehr. Das erstere „Kapitel“ würde ignoriert werden und die innerhalb der Klammern stehende Zeichenkette als einfacher Fließtext gedruckt werden).

Shannon (1948) beschäftigte sich bereits 1948 mit wesentlichen Grundlagen der heutigen Darstellung und Übertragung von Informationen, insbesondere der menschlichen Sprache und Kommunikation. Heutige maschinelle Systeme zu diesem Zweck (bspw. ChatGPT, DeepL, Gemini und co.) wirken zunächst wie „magische“ Blackboxen, arbeiten jedoch auf Grundlage von statistischen Modellen. Das zugrundeliegende Konzept kann jedoch sehr schnell auf den Punkt gebracht werden: Eine künstliche Intelligenz (KI) erhält einen Input, für welchen ein bestimmter Output erwartet wird (Beispiel: „Einfügen“ als nächstes Wort eines zu übersetzenden Satzes und „insertion“ im Kontext innerhalb des Satzes als Erwartung (substantiviertes Verb)), und produziert einen Output, welcher mit dem Erwarteten abgeglichen wird. Sollte das Resultat von der Erwartung abweichen (z.B. „insert“ entstehen), so kann dieser Fehler erkannt werden und im Modell dazu beitragen, dass (gegeben einer bestimmten, sequentiellen Folge von Wörtern (in der Satzstruktur)) dieser „Fehler“ von nun an seltener passiert. Allerdings wird klar, dass eine KI unabdingbar Fehler machen muss, denn nur so kann diese „lernen“. Diese theoretische Grundlage führt dazu, dass immer alle möglichen Permutationen einer Übersetzung in Betracht gezogen werden müssen, so unwahrscheinlich sie auch seien. Angemerkt sei zu dem Vorherigen, dass bekannte und bereits rein in den menschlichen Sprachen auftretende Problem (bzw. mögliche Missverständnisse bereits im Verstehen einer Sprache, ausgelöst durch Mehrdeutigkeiten von Wörtern) in dieser Arbeit nicht näher verfolgt werden.

Bekannte Technologien, wie z.B. Google Translate, DeepL und co. scheinen rein wort-interne Probleme zu lösen. Sprachliche Missverständnisse könnten aber immer dann entstehen, wenn sich mehrere Sprachen miteinander vermischen, welche sich ein Lexikon teilen (bspw. hier: TeX,LaTeX, ... und die, zunächst, englische Sprache). Die Frage ob ein Wort übersetzt werden darf oder nicht, unterscheidet einzelne Problemarten (Fälle). Die Hoffnung besteht, dass diese Probleme bereits gelöst sind, allerdings wird *ein*

Übersetzer in der folgenden Schilderung einzelner Problemfälle Fehler machen können *müssen*.

2 Problemfälle

2.1 Herangehensweise

2.1.1 Vorgehen

Auf dem Weg LaTeX-Dokumente zu übersetzen, startet man unausweichlich bei der Betrachtung der Entstehungsweise eines solchen Dokumentes. Üblicherweise soll ein fertiges Dokument in einem allgemein verbreitetem und zugänglichen Format vorliegen. Etabliert hat sich hierbei das Portable Document Format (PDF), welches ursprünglich von der Firma Adobe entworfen und spezifiziert wurde.Ein TeX-Compiler produziert ein solches Dokument ausgehend von einer einzelnen `.tex` Datei, welche ihrerseits Quellcode trägt, welcher mehrere Sprachen vermengt (die Programmiersprache TeX und menschliche, druckbare Sprachen).Bereits ein TeX-Code besitzt (theoretisch) das Potential unendlich viele Fehler zu tragen, sollten alle von einem Übersetzer vorgefundene Zeichenketten übersetzt werden. Der sprachlichen Syntax folgend könnte man bestimmte Elemente eines Quelltextes umgehen, aber die genaue semantische Bedeutung einzelner Strings im Quelltext muss immer dahingegen evaluiert sein, ob die vorliegende Zeichenkette Teil des „fertigen“ Dokumentes werden soll, oder nicht.

Klar ist also, dass etwaige Lösungen nicht auf Grundlage der sprachlichen Struktur des Quelltextes arbeiten dürfen, sondern die logische Struktur des Dokumentes verstehen/erkennen sollten um mögliche Fehlinterpretationen von Strings zu vermeiden. Abhängig von der Größe der innerhalb von TeX/LaTeX-Dokumenten auftretenden kleineren, logischen Strukturen (insb. den nach Knuth (1986) vorgesehenen) treten verschiedene Fehlerquellen auf. Fehler in den Kleinsten bewirken immense Änderungen auf die Inhalte und den Aufbau des entgültigen Dokumentes. Sie tragen zur Folge, dass der übersetzte Quelltext kein äquivalentes Dokument (sprachlich, als auch strukturell) zu der originalen Beschreibung erzeugen kann.

Ähnlich wie bei der Erstellung von Dokumenten insgesamt teilt sich jedes LaTeX-Dokument in verschiedene Bereiche. Diese können sich sowohl echte/relative Fläche auf einer Seite (einem Papier/Bildschirm), textliche Paragraphen/Abschnitte, Graphiken, Überschriften und dergleichen beziehen, aber auch auf größere Konzepte, wie Inhaltsverzeichnisse, Literaturverzeichnisse (Zitationen, insb. Links) oder spezifischere Vorgaben für Dokumente beziehen (bspw. die Form wissenschaftlicher Arbeiten).

Wohingegen man solche Flächen und Texte auf einem Papier als sehr direkt zugänglich betrachten kann, muss eine rein textliche Beschreibung solcher Dokumente alternativen finden und zum Beispiel Farben und Flächen durch Wörter und Zahlen beschreiben. Zu erwarten wäre also, dass die kleinste zu betrachtende Struktur von TeX-Quelltexten einzelne Zeichen sind. Diese allein bilden allerdings noch keine interessante (als auch logische) Struktur in TeX, sondern erst konkrete Befehle/Funktionen, sowie diesen übergebbare Parameter. Hierbei zeigen sich diverse Wege, wie sowohl die Übergabe dieser Parameter aussehen *könnte*, als auch die Parameter selbst und ob ein solcher Parameter zu übersetzen ist, oder nicht.

Fernab der Sinnhaftigkeit vereinzelter Formen solcher Parameter/-übergaben, sind alle möglichen Äußerungsformen von TeX-Quellcodes zu betrachten. Aufbauend auf diesen kleinsten Strukturen muss sich nun ein Weg in Richtung größerer Strukturen gebahnt werden, welche teilweise auf andere Quelltexte zugreifen und nicht zwingend denselben, einheitlichen syntaktischen Vorgaben unterliegen.Besonders Quelltexte, deren absolute Position im System (meint: ein beliebiges Betriebssystem) nicht immer bekannt

ist, können vereinzelt Strings tragen, deren Übersetzung erforderlich wäre, aber schnell übersehen werden könnten.

Aus Sicht des Endnutzers sollen hierbei natürlich alle ablaufenden Prozesse verborgen sein, sodass er/sie/etc. nur das (übersetzte) Dokument wahrnimmt. Insbesondere müssen Fehler vermieden sein, welche überschüssige Wörter im Dokument produzieren oder Texte von einer Übersetzung ausschließen, bzw. dafür sorgen, dass vereinzelte Texte (beliebiger Größe) nicht den vorgesehenen Weg in das Dokument finden. Zudem muss darauf geachtet werden, dass präzise Formulierungen gewählt werden, sollte ein Text einen spezifischen fachlichen Kontext tragen, in denen bestimmte Übersetzungen von Wörtern erwartet werden, welche von den intuitiven sprachlichen Pendants dieser Wörter abweichen. Vereinzelte Fällen zeigen Ausnahmen, welche technisch realisierbare Probleme hervorrufen, aber keine eindeutig richtigen Lösungen mit sich führen. Probleme/Fehler dieser Art lassen sich sprachlich, als auch graphisch finden.

2.1.2 Darstellung der Beispiele

Für einzelne Beispiele werden erwartete **Soll**-Übersetzungen und mögliche **Ist**-Übersetzungen gegenüber gestellt. Als Parade soll hierbei das Wort *set* dienen, welches kontext-abhängig eine andere Übersetzung verlangt. Gewünscht wäre in dem hierigen Beispiel, dass das Wort zu *Menge* im Deutschen übersetzt wird, da der Satz „Let's consider a finite set of vertices and edges." als „Betrachten wir nun eine endliche Menge an Knoten und Kanten." erwartet wird. [1] Würde man alle Wörter einzeln, für sich selbst, betrachten, dann würde man zu einer Übersetzung von *set* zu *setzen* tendieren. Dies würde allerdings sowohl sprachlich keinen Sinn mehr ergeben, als auch die syntaktischen Regeln der deutschen Sprache brechen (denn: Verben werden nicht durch Adjektive beschrieben, sondern durch ein Adverb). Die Darstellung des erläuterten Beispiels gilt musterhaft für folgende Beispiele.

[1] In diesem Kontext wäre auch die Formulierung „abgeschlossene" statt „endliche" denkbar

2.2 Elemente in einem Quelltext

2.2.1 Paragraphen, Überschriften und Anmerkungen

Die einfachste Form in welcher ein Übersetzer zu übersetzende Zeichen- / Wortketten innerhalb eines Quelltextes vorfinden kann, sind reine Fließtexte. So wäre das einfachste Dokument, welches übersetzt werden könnte, ähnlich zum Original im beispielhaften Problem 2.1.

Es würden hierbei insbesondere dann Fehler entstehen, wenn Befehle selbst (wie `documentclass`) und deren Parameter (wie `document`) übersetzt werden. Idealerweise sehen sich nur die Strings eines Dokumentes übersetzt, welche am Ende auch (nach dem Kompilieren durch einen TEX-Compiler) in diesem zu sehen sein sollten.

Problem 2.1 (Anfang)

Original

```
1 \documentclass{article}
2 \begin{document}
3 Knowledge
4 \end{document}
```

Potentieller Fehler

```
1 \Dokumentenklasse{Artikel}
2 \Beginne{Dokument}
3 Wissen
4 \Ende{Dokument}
```

Soll-Übersetzung

```
1 \documentclass{article}
2 \begin{document}
3 Wissen
4 \end{document}
```

Alle Wörter in einem TEX-Quellcode zu übersetzen produziert garantiert einen Fehler.
Nur bestimmte Inhalte dürfen übersetzt werden.

Problem 2.1 (Ende)

Strukturlose und unformatierte Blocktexte erschweren einem Leser (Endnutzer) aber das Nachvollziehen der Inhalte und verhindern es, dass zum Beispiel Inhalte von Paragraphen mit deren Überschriften assoziiert werden könnten, um hierbei den Fokus von der exakten Wortwahl hin zur Bedeutung des Themas zu rücken.Als einfachstes stilistisches Mittel sind hier, wie angedeutet, Überschriften oder Kapiteltitel denkbar. Diese sind (vorerst) einfach an einer endlichen Menge an Befehlen erkennbar. Wichtig ist in solchen Fällen, dass — selbstverständlich — die eigentlichen Befehle nicht übersetzt werden, damit ihre Bedeutung innerhalb der Programmiersprache TEX erhalten bleibt. Konkreter sollte zum Beispiel `\chapter{Bracelets}` zu `\chapter{Armbänder}` werden und nicht in `\Kapitel{Armbänder}`. Parallel hierzu sollten auch Untertitel (Captions), Randbemerkungen (Fußzeile, Randbereiche, als auch sonstige Orte) und der Titel des Dokumentes selbst übersetzt sein. Technisch sind solche Wort- bzw. Zeichenketten im Quelltext in der Form `\command{Translatable content}` vorliegend, wobei Kommandos generell nicht übersetzt werden sollten.

2.2.2 Listen und Tabellen

Weitere stilistische Mittel, welche sich innerhalb von Dokumenten finden lassen, sind beispielsweise Listen oder Tabellen. Diese erscheinen auf den ersten Blick nicht als neues Problem, da innerhalb dieser Strukturen freistehende (Fließ-) Texte für gewöhnlich ihren Weg in das Dokument finden sollen. Problematisch können allerdings Fälle werden, in denen solche Strukturen Inhalte tragen, die nicht übersetzt werden dürfen.

2.2.3 Mathematische Typsetzung

Üblicherweise wird man in einem TEX-Dokument mathematische Inhalte vorliegen haben, bspw. Formeln, Matrizen oder Gleichungen. TEX selbst wechselt hier durch ein Dollarzeichen `$` in den *math mode* hinein (und heraus), aber ein Übersetzer steht zunächst nur vor Fragezeichen. Im ersten Moment wäre davon auszugehen, dass solche „Mathematik-Bereiche“ in einem Dokument keine sprachlichen Inhalte tragen und daher ohnehin nicht übersetzt werden würden. Es muss allerdings darauf geachtet werden, dass diese Bereiche sprachliche Inhalte tragen *könnten* und das in einer entweder direkt erkennbaren Form (da bspw. mittels `\text{}` innerhalb des Bereiches ein Textbereich erzeugt wird) oder in einer Form, welche Interpretationsspielraum lässt. So wäre es für zum Beispiel `$x = 4 \textbf{means: something is equal to four.}$` klar, dass zu `$x = 4 \textbf{bedeutet: etwas ist gleich vier.}$` übersetzt werden soll. In der Zeichenfolge `$x = 4 means: something is equal to four.$` fehlt ein solcher Indikator allerdings, wodurch es fraglich ist, ob eine Übersetzung `$x = 4 bedeutet: etwas ist gleich vier.$` entstehen wird. Genauso könnte der Fall eintreten, dass an der Stelle von einzelnen Zeichen, ganze Wörter als Variablennamen genutzt werden. Beispielsweise sollte `$cars = 3, price = 10\$ $` nicht zu `$Autos = 3, Preis = 10\$ $` werden, da ein Verändern der Variablennamen das Nachvollziehen von mathematischen Herleitungen unmöglich macht. Dementsprechend wäre ein Übersetzen von Wörtern innerhalb mathematischer Umgebungen zwar nicht immer falsch, aber sicherer wäre ein Absehen von diesem (da es vorgesehene und erkennbare Wege gibt, Texte in mathematischen Bereichen zu integrieren).

2.2.4 Umgebungen und Verweise

Zuvor beschriebene „mathematische Bereiche“ sind nur eine Art spezielle Einstellungen für bestimmte Flächen eines Dokumentes vorzunehmen. Ob innerhalb einer Umgebung übersetzt werden soll, muss für diese spezifisch ausgewertet werden (worauf in 2.2.8 näher eingegangen wird). Wichtiger ist zunächst, dass die syntaktischen Regeln bei der Nutzung von Umgebungen innerhalb TEX erhalten bleiben. So darf `\begin{escape}` nicht zu `\begin{Flucht}` werden und `\end{escape}` nicht zu `\end{Flucht}`. Was zunächst trivial wirkt, kann aus technischer Sicht Probleme hervorrufen. Entgegen der in 2.2.1 geschilderten Art und Weise mit welcher zu übersetzende Strings erkannt werden könnten, sind in gleicher Form auch Zeichenketten vorliegend, welche nicht übersetzt werden dürften. Dies gilt für mehr als nur die Arbeit mit Umgebungen, welche vermutlich seltener auftreten könnten, denn auch (Hyper-) Referenzen und Links fallen in diese Kategorie. Einfache URL zu erkennen, ist auf Grund deren vorgegebener Struktur sehr leicht , wodurch man davon ausgehen könnte, dass selbst der Versuch diese zu übersetzen unveränderte Inhalte produziert (da ein URI eine etablierte Struktur ist). Möchte man jedoch Schlüsselwerte nutzen, wie bspw. bei `\label` und `\ref` darf nicht die Situation entstehen, in welcher z.B. aus `\ref{topic:content}`

plötzlich `\ref{Thema:Inhalt}` wird oder das ein veränderter Schlüssel in einem Label zu einem fehlenden Anker für jegliche Referenzierungen des Labels (bzw. seiner Dokumenten-internen Position) führen. [2]

Inhalte von geschwungenen Klammern sind demnach individuell auszuwerten. Allerdings ist dies nicht die einzige Art von Klammer, welche eine syntaktische Bedeutung in TEX trägt. Beispielsweise können benannte Labels auch mit `\hyperref` referenziert werden. Dieser Befehl erlaubt es den Schlüssel in eckige Klammern zu schreiben und danach in geschwungenen Klammern den Text zu denotieren, welcher im Dokument als Maske für die Verknüpfung dienen soll. Es zeigt sich also eine weitere Form *zu übersetzende* Texte zu erkennen und zwar `\command[non-translatable string]{translatable content}`.

2.2.5 Verzeichnisse für bestimmte Elemente

2.2.6 Verbatim und Kommentare

2.2.7 Die Präambel

Das gleichnamige Paket, aus welchem dieser `hyperref`-Befehl stammt, zeigt, wie es einige Pakete erlauben Schlüssel-Wert Paare für das Setzen einiger spezifischer Optionen zu übergeben. Beispielsweise würde der Befehl `\usepackage[pdftitle=something]{hyperref}` den Titel der entstehenden PDF (für einen PDF-Reader/Renderer) zu „something“ ändern. Nachdem das Dokument übersetzt wurde, wäre es durchaus wünschenswert, wenn auch der Titel übersetzt wurde (`\usepackage[pdftitle=Irgendetwas]{hyperref}`). Demnach können auch eckige Klammern Strings enthalten, welche es zu übersetzen gilt. Neben Paketen, wie `hyperref`, erlaubt die Präambel einer Vielzahl an solchen strukturellen Änderungen oder anderen Einstellungen, welche sich über das ganze Dokument erstrecken. Insbesondere ist in dieser die Definition von eigenen Umgebungen von Interesse, da diese textliche Inhalte in verschiedenen Formen tragen können, welche es alle zu übersetzen gilt. [3]

2.2.8 Eigene Makros

Eigene Umgebungen

2.2.9 Der Weg zu anderen Quelltexten (und Dateien)

Technisch gesehen trifft man hier nicht direkt auf ein neues Problem, aber insbesondere das Übersehen von direkt eingebundenen, anderen TEX-Quelltexten darf nicht stattfinden. Das Übersetzen vereinzelter Teile bestimmter TEX-Befehle kann diese Situation provozieren. Ein einfaches Beispiel zeigt die (hier) imaginäre Quelltextdatei `clock.tex`, welche in einem ursprünglichen Dokument eingefügt werden soll. Würde beispielsweise `\include{clock}` zu `\include{Uhr}` übersetzen werden, kann dieser Befehl nun nicht mehr als `\include{clock.tex}` interpretiert werden und wird als `\include{Uhr.tex}` aufgegriffen. Unabhängig davon, ob `Uhr.tex` existiert, werden fälschlicherweise die Inhalte von `clock.tex` nicht mehr aufgegriffen und werden daher nicht Teil des beschriebenen Dokumentes (*beschrieben* durch den Quelltext). Das Gleiche

[2] Hierbei entsteht allerdings die Ausnahmesituation, in welcher *alle* Schlüsselwerte erkannt und übersetzt wurden. Dieser Fall wird genau dann problematisch, wenn dieser Schlüssel in einem anderen Label bereits verwendet wurde.

[3] Makros lassen sich im gesamten Dokument definieren. Umgebungen allerdings ausschließlich in der Präambel.

gilt auch für den Befehl `\input` und `\includepdf` (oder auch: `\includegraphics` und Weitere). Interessant wird an dieser Stelle auch, inwiefern die Inhalte von anderen Dokumenten (PDF) erkannt werden, bzw. wie mit diesen Umgegangen wird oder werden sollte/könnte. Dies wird im späteren Kapitel 2.5.2 näher betrachtet. Fehlerquellen dieser Art sind im Quelltext in der Form `\command{Non-translatable content}` vorliegend.

2.3 Probleme für mehrere Quelltexte

2.3.1 Triviale Fälle

Bereits bekannt ist, dass Befehle, welche andere Quelltexte einbeziehen, um ein gesamtes Dokument zu beschreiben, nicht übersetzt werden dürfen, dürfen nun aber auch die Inhalte dieser Dateien nicht unübersetzt bleiben. Wichtig ist bei einem Einbinden von anderen Quelltexten auf diese Art und Weise, dass die Dateipfade relativ zum Ausgangsquelltext vermerkt werden. Sieht ein Übersetzer also ein `\input{clock.tex}`, dann muss er diese Datei beginnend im *root*-Verzeichnis des Dokumentes suchen und darf nicht ausgehend von der aktuellen Datei, in welcher er sich befindet, eine Suche beginnen. `\include` können sich nur im ursprünglichen Quelltext befinden, `\input` kann allerdings auch in einem Quelltext stehen, der seinerseits bereits über ein `\include` oder `\input` eingebunden wurde und könnte in einem Unterverzeichnes von *root* liegen.

2.3.2 Pakete und Klassen

Viele Pakete erlauben es, dass die in Ihnen vorliegenden textlichen Inhalte, die auch im enstehenden Dokument gedruckt werden, angepasst werden können. Dies ist allerdings nicht immer (einfach) möglich. Nicht jedes Paket liefert einen eigenen Befehl hierfür oder muss die spezifischen Makros und deren Definitionen erkenntlich/öffentlich machen, damit deren Funktion reproduziert und gewünschte Strings geändert werden können. Allerdings müssen alle mit einem Dokument assoziierten Dateien (zu welchen Pakete `.sty` zählen) vorliegen, sobald dieses entstehen soll. Nun

2.3.3 Zitationsstil und Bibliotheken

2.3.4 Darstellung von Quellcode

2.4 Spezielle/Spezifische Probleme

2.4.1 Category Codes

2.4.2 Nutzer-eigene Klammern

2.4.3 Neudefinition bekannter Makros

2.4.4 PDF-Kommentare

2.5 Andere Probleme

2.5.1 Provozierte Schwierigkeiten

2.5.2 Erkennung und Auswirkung von PDF

3 Stand der Technik

3.1 Übersicht

Erste Ansätze zur automatischen Be- und Verarbeitung von LaTeX-Dokumenten lassen sich bis in die 1990er Jahre verfolgen Wilson (1997) (abgesehen von einem Compiler natürlich). Die Übersetzung von den rein wort-sprachlichen, textuellen Inhalten eines Dokumentes[4] zeigt sich wiederum als ein Problem, welches eine Niche bedienen zu scheint. Von besonderem Interesse für die Sprachübersetzung heutzutage sind große Sprachmodelle, bzw. „künstliche Intelligenzen“, wie zum Beispiel DeepL, Google Gemini (Cloud Translate) oder andere GPT-Modelle, welche in diesem Kontext trainiert wurden (GPT=generative pre-trained text). Deshalb seien auch im Kontext der Sprachübersetzung von LaTeX-Dokumenten Ansätze der Art in den Vordergrund gerückt und jegliche betrachtete Technologie kurz aufgeführt.

3.1.1 Nicht auf TeX spezialisierte Werkzeuge

ChatGPT Als „All-Rounder“ unter den heutigen künstlichen Intelligenzen ist es denkbar, dass auch ein *general-purpose* Sprachmodell, gegeben eines passenden *prompting*, dazu in der Lage sein **könnte** zuvor beschriebene Fehlerquellen zu meiden. Ein Ansatz dieser Art könnte allerdings genau dann scheitern, sobald einige benötigte Informationen (bspw. die Definition eigener Makros, Umgebungen, . . .) nicht mehr in einem einzigen Quellcode vorliegt.

GitHub Copilot Ein auf die Arbeit mit Quellcode spezialisiertes Tool, wie benanntes „GitHub Copilot“, könnte unter Umständen noch effektiver/effizienter/besser mit Quelltexten umgehen und ist ähnlich wie ChatGPT zu behandeln.

DeepL Konkurrierend zu dem jahrelangen Marktführer (in Hinsicht auf durch Computer realisierte Sprachübersetzung) Google, gewann DeepL zunehmend an Bekanntheit und ist neben, Google Translate, ein viel genutztes Werkzeug, insbesondere im Kontext von Anwendungsgebieten, in welchen das Übersetzen durch einen Menschen zu viel (Echt-) Zeit beanspruchen würde. Die API lässt hierbei die Übergabe von einem *Kontext* zu, welcher sich allerdings in der gewählten Wortwahl innerhalb der Ausgabe zeigen wird und keinen Einfluss auf die eingelesenen Inhalte haben sollte. (In dem Sinn, dass durch einen Kontext „LaTeX“ nicht davon auszugehen ist, dass Quellcode als Input erwartet ist, sondern Texte über die benannte(n) Technologie(n)). Von besonderem Interesse ist bei einem Ansatz dieser Art also, inwiefern diese Technologie selbstständig dazu fähig ist Quellcode zu erkennen und weniger, ob sie *perfekte* Ausgaben (fehlerfrei) liefert.

Google Cloud Translate Natürlich ist besagter Marktführer für die Übersetzung menschlicher Sprache im gleichen Kontext, wie auch DeepL, zur Bewältigung derselben Aufgabe heranziehbar. Nur muss zunächst

[4] `pgfplots` wäre bspw. dazu in der Lage Audiosignale darzustellen, welche sich maschinell interpretieren ließen. Solche Ansätze werden hier allerdings nicht weiter verfolgt

klargestellt sein, dass Google Translate nicht gleich Google Cloud Translate ist. Google Translate bezeichnet üblicherweise das bekannte Browser-Tool, welches höchstwahrscheinlich (hoffentlich) mittlerweile auch auf einem GPT aufbaut. Da ein solches Tool auf eine möglichst einfache Bedienbarkeit zugeschneidert ist, verliert es an Transparenz hinsichtlich der eigentlichen zugrundeliegenden Technik. Wirklich sicher, dass ein solcher, moderner Ansatz (auf Grundlage großer Sprachmodelle, statt reinen statistischen Modellen) [5]. genutzt wird, kann man sich allerdings nur sein, sollte man vollständige Einsicht in die Zugrundeliegenden Quelltexte erhalten. Dies ist aber, aus verständlichen, marktwirtschaftlichen Gründen nicht immer möglich. Daher wird darauf vertraut, dass auf Angabe des Unternehmens die „Google Cloud Translate"-API so arbeitet, wie es versprochen wird.

Microsoft Translate und Weitere Google, Microsoft und DeepL sind nicht die Ersten und werden auch nicht die letzten Anbieter von maschineller Sprachübersetzung sein. Statt hier noch weitere unkonkrete Ansätze zu listen, wird nun spezifischere Technologien in den Vordergrund gerückt.

3.1.2 Existierende Technologien und Ansätze

GPT-LaTeX-Translator Der von Suñé und Arcuschin (2023) entwickelte Ansatz verfolgt die zuvor beschriebene Idee, ein *general purpose* Sprachenmodell heranzuziehen. Ihre genaue herangehensweise geht aus Zeile 57 des Quellcodes (Python) „GPTTranslator.py" hervor, in welchem sie lediglich zugrundeliegende `.tex`-Dateien so weit zerlegen, dass sie der Anwendungsschnittstelle (damalig) gerecht werden und dann der künstlichen Intelligenz die Information mitteilen, dass LaTeX-Quellcode vorliegt.

Textsynth/trsltx Helluy (2025) arbeitet an einem sehr spezifischen Ansatz, welcher sich der gegebenen Aufgabenstellung sehr gezielt zu nähern scheint. Allerdings zeigt er selbst auf, inwiefern seine Herangehensweise Fehler in LaTeX produzieren kann. Insbesondere verweist er auf Schwächen hinsichtlich eigener Makros und anscheinenden Labels, Referenzen und Zitationen, die die KI eigenständig hinzufügt. Auch bemerkt er, dass einige Umstände zu unvollständigen Übersetzungen führen können, welche es zu erproben gilt.

MathTranslate Sun (2025) präsentiert auf der WebSite von seinem Tool „MathTranslate" einen Ansatz, welcher zunächst äußerst vielversprechend erscheint. Ein näheres Betrachten des vorliegenden Quelltextes zeigt aber schnell potentielle Schwächen auf, welche auf die gewählten Übersetzungs-Softwares zurückführbar sind. Der Quelltext `translate.py` weist hierbei nur auf eine Nutzungsmöglichkeit von entweder (zuvor entgegen argumentiertem) Google Translate oder aber auf Tencent hin. Hierbei wird zweitere Engine näher betrachtet.

[5] *Große Sprachmodelle*: Lernen die Sprache, bzw. werden auf Grundlage einer Sprache trainiert, wohingegen *statistische Modelle* sich rein auf die wörtlichen Übersetzungen fokussieren. Wesentlichster Unterschied der Ansätze ist, dass große Sprachmodelle eine Semantik in den Wörtern suchen (in dem Sinn, dass sie die Inhalte „versuchen zu verstehen", wohingegen einfachere statistische Modelle nur die Wörter und deren grammatikalischen Zusammenhänge untersuchen, um daraus die treffenste Übersetzung zu finden)

TransLaTeX Der von Erken /2023 präsentierte Ansatz beschäftigt sich so konkret mit der Hürde der automatischen Sprachübersetzung von LaTeX-Dokumenten, wie kaum ein Anderer. Jedoch scheint hier der Fortschritt bei (selbst eingeschätzten) 88% vor circa 2 Jahren stehen geblieben zu sein und offenbart in der Beschreibung der Website / des GitHub-Repositories vereinzelte Schwächen `https://gitlab.math.unistra.fr/cassandre/translatex`. Zudem ist dieser Ansatz auf einer interpretierten Sprache (Python) basierend, welche stets einen performanten Nachteil gegenüber (optimierten) prozeduralen Sprachen zeigen wird, da Zweitere kein *Interpreter*- Modul benötigen.

PolyMath Translator Ohri und Schmah (2020) gingen einen Weg über die Konvertierungs-Software `pandoc`, um in dieser den abstrakten Syntax-Graphen/Baum zu ermitteln, auf dessen Grundlage sehr klar ist, welche Inhalte textliche Strings sind, welche in einem kompilierten Dokument angezeigt werden sollen und welche Inhalte lediglich Teile der Dokument-Beschreibung sind. Allerdings zeigt sich ihre veröffentlichte Lösung als heute (Ende 2025) nicht mehr zugänglich, sodass nur noch der Ansatz getestet werden könnte.

3.1.3 Nicht auf Sprachübersetzung konzentrierte Werkzeuge

Pandoc

Translate Package

3.2 Testverfahren

Entlang der zuvor geschilderten Problemfälle ließen sich theoretisch gesehen unendlich viele explizite Fehlerbeispiele produzieren, welche ihrerseits inhaltlich einem *Lorem Ipsum* gleichen könnten („Lorem Ipsum“ bezieht sich auf ein häufig genutztes Beispiel, wenn es darum geht Texte und deren graphische Aufbereitung zu visualisieren. Dieser Platzhalter-Text sieht aus wie Latein, ist inhaltlich aber sinnfrei (**??**)). Da man bei der Nutzung von TeX aber immer die hauptsächliche Nutzung für wissenschaftliche/mathematische Kontexte im Vordergrund belassen sollte, werden insbesondere die auf ArXiV verfügbaren TeX-Quellcodes von Interesse (wobei eine Kompilation von älteren Texten der Art durch **?** zur Verfügung gestellt sind). Dies bedeutet, dass zunächst solche Quelltexte getestet werden, und inwiefern sie „richtige“ Übersetzungen liefern. Dem hierigen Autor ist ein Bestätigen in dieser Hinsicht nur für das Sprachpaar EN-DE möglich, allerdings existieren Technologien (Methoden), wie (Sacre-) BLEU oder TeXBLEU, um algorithmische, maschinelle Bewertungsgrundlagen zu schaffen.

Diese Methodik bedient allerdings bisher nur eine Auswertung der rein sprachlichen Richtigkeit ggb. menschlichen Sprachen und der Maschinensprache TeX. Auf diese Art und Weise werden einige spezifischere Fälle und „Ausnahmen“ nicht näher betrachtet, wodurch gezielt auf diese geprobt werden muss.

Neben den Bulk-Datensätzen von ArXiV entstanden daher auch eigene, kleinere Beispiele, welche dem Anhang **??** zu entnehmen sind.

3.3 Tests

3.3.1 Erzielte Werte

3.3.2 Auswertung

3.3.3 Übrige Probleme

4 Eigener Beitrag

4.1 Existierenden Herangehensweisen

4.1.1 Übersetzung von Quellcode

Diese Herangehensweise verfolgt den Weg den gesamten Quelltext, ohne größere Interpretation, als Eingabe für einen Übersetzer zu nutzen, um daraus einen neuen, übersetzten Quelltext zu erhalten, in welcher sämtliche LaTeX relevanten Inhalte unverändert (ggb. dem Original) vorzufinden sind. Dieser neue Quelltext soll dann, genauso wie der Originale, mit Hilfe eines bekannten TeX-Compilers (bspw. `pdflatex`, `xelatex`, `lualatex`, ...) in eine PDF überführt werden.

4.1.2 Quelltext-filternde Tokenizer

Jeder TeX-Compiler muss einen ihm vorliegenden Quelltext in seine einzelnen Token zerlegen und (in dem Fall, dass der Quelltext ein Dokument beschreibt) in eine DOM-ähnliche Struktur überführen. Diese abstrakte, aber vollständige und vollständig beschriebene/detaillierte Struktur des Dokumentes kann als Grundlage genutzt werden, um aus dieser rein textliche/sprachliche Inhalte zu extrahieren und nur noch diese einem Übersetzer zu präsentieren. Danach kann diese vollständige Struktur mit übersetzten sprachlichen Inhalten in das beschriebene Dokument übersetzt werden.

4.1.3 Ausweichen in andere Formate

Dieser Ansatz ist ähnlich zu Vorigem, unterscheidet sich jedoch in der Hinsicht, dass auf ein „Zwischenformat“, also einer Hilfsdatei ausgewichen wird (welche theoretisch gesehen Grundlage für eine künftige Verarbeitung des Dokumentes sein/werden könnte).

4.2 Andere Lösungswege

4.2.1 Abstrakte Beschreibung

Mitwirkung im Compiler Ähnlich wie 4.1.2 könnte auch ein TeX-Compiler selbst angepasst werden.

4.2.2 Technische Umsetzungsmöglichkeiten

4.2.3 Konzeptionelle Probleme

4.3 Empfohlene Maßnahmen

5 Fazit

5.1 Zusammenfassung

5.2 Ausblick

5.3 Weiterführend

6 Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt und ohne fremde Hilfe verfasst habe. Dazu habe ich keine außer den von mir angegebenen Hilfsmitteln und Quellen verwendet und die den benutzten Werken inhaltlich und wörtlich entnommenen Stellen habe ich als solche kenntlich gemacht. Ich versichere, dass die eingereichte elektronische Fassung mit den gedruckten Exemplaren übereinstimmt.

Rostock, den 02.12.2025

Hendrik Theede

Literatur

Knuth, D. E. (1986), *The TeXbook, ISBN: 9780201134476*, Addison-Wesley Professional.

Lamport, L. (1994), *LaTeX: A Document Preparation System, 2nd Edition, ISBN: 9780201529838*, Addison-Wesley Professional. available at: https://www.latex-project.org/help/books/tlc3-digital-chapter-samples.pdf (last Access: 04.10.2025).

Shannon, C. E. (1948), 'The mathematical theory of communication', *The Bell System Technical Journal, ISSN: 0343-6993* **(vol. 27)**. Harvard Reprint available at https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf (last Access: 16.10.2025).

Wilson, P. R. (1997), 'Ltx2x: A latex to x auto-tagger', https://latex.us/support/ltx2x/ltx2x.html (last Access: 28.10.2025) und https://ctan.org/tex-archive/support/ltx2x (last Access: 28.10.2025).

A Anhänge