Über die automatische Transliteration des Arabischen nach dem Standard der Deutschen Morgenländischen Gesellschaft

Bachelorarbeit

von Rashid Harvey

an der

Freien Universität Berlin

am

Institut für Mathematik und Informatik

Betreut von Professor Lutz Prechelt

Juni 2024

Abstract

German

Die automatische Transliteration des Arabischen ist ein offenes Problem, deren Lösung Wissenschaftlern, die sich mit der arabischen Sprache befassen, die Arbeit wesentlich erleichtern könnte. Bei der Transliteration geht es darum, einen arabischen Text so darzustellen, dass ein Laie ihn lesen kann und zugleich für den Experten Eindeutigkeit gewährleistet ist.

Das Ziel dieser Bachelorarbeit ist, Beiträge zur Lösung des Problems der automatischen Transliteration zu liefern. Dazu analysiert sie verschiedene Quellen zur Bestimmung der Anforderungen. Dann betrachtet sie eine Vielzahl von regelbasierten Systemen, Heuristiken und modernsten Machine-Learning-Systemen.

Im Ergebnis stellt diese Arbeit fest, dass sowohl ein regelbasierter Ansatz als auch ein Ansatz unter der Verwendung von künstlicher Intelligenz vielversprechend erscheinen. Aufgrund der Komplexität der arabischen Sprache, kann noch kein Ansatz perfekte Ergebnisse liefern.

English

The automatic transliteration of Arabic to German is an open problem, even though a solution might provide scientists a convenient way to do transliteration. Transliteration is the process of representing Arabic in a way that an uninitiated can read it while avoiding ambiguity.

This thesis aims to contribute to a possible solution to the problem by firstly analysing various sources to elicit the requirements. Secondly, it gives an overview of possible solutions like rule-based systems, heuristics and most modern machine learning systems.

The thesis concludes that rule-based approaches and approaches dependant on artificial intelligence provide great solutions. However, it seems that the complexity of Arabic can not yet be overcome completely by either approach.

Für die Unterstützung und Begleitung

danke ich

Nadia El-Ali,

Nizar Romdhane,

Doğa Akpınar,

Dr. Theodore Beers,

Dr. Jonas Müller-Laackman,

Dr. Michael Becker,

Victoria Mummelthei,

und Dr. Till Grallert

Inhaltsverzeichnis

I. Einführung

Transliteration bzw. auch Transkription oder Umschrift ist die Repräsentation einer fremden Schrift in einer bekannten Schrift. Dabei ist der Grundgedanke, einem der fremden Schrift Unkundigen das Lesen eben dieser zu ermöglichen. [1] So können die meisten Deutschen "القرآن" nicht lesen, sehr wohl jedoch "al-Qurʾān" (der Koran) so wie es nach der Deutschen Morgenländischen Gesellschaft (DMG) transliteriert werden soll. Diese Umschrift ist im deutsch-akademischen Bereich sehr verbreitet und wird unter anderem auch von Wikipedia verwendet [8]. Das System der DMG ist jedoch deshalb eine Transliteration und im wissenschaftlichen Kontext so verbreitet, weil es auch einen einheitlichen wissenschaftlichen Standard aufrechterhält, der verlangt, dass "die Wörter so umgesetzt werden, dass ihre Schreibweise in der Originalsprache erkennbar wird" (Eindeutigkeits- oder Rekonstruierbarkeitsprinzip). [1] Die Umschrift "Koran" kann diesem Standard etwa nicht standhalten. Der erste Buchstabe ist "falsch" (ك statt ق), es gibt im Arabischen kein O und die Länge der Vokale wird nicht unterschieden (das u ist kurz, das a ist lang).

Das Problem, das sich diese Bachelorarbeit widmet, ist nun diese Transliteration zu automatisieren oder zumindest zu semi-automatisieren. Die Lösung dieses Problems wäre eine echte Bereicherung für die wissenschaftlichen Bereiche, die sich mit der arabischen Sprache beschäftigen, also explizit: Arabistik, Islamwissenschaft und Orientalistik. Im Laufe meiner Bachelorarbeit stieß dieses Thema immer wieder auf großes Interesse, was mich ermutigte, etwas zu entwickeln, das einen konkreten Nutzen haben würde.

> *"Alles in allem ist das ein schönes BA-Projekt und ich möchte Sie sehr dazu ermutigen, sich nicht entmutigen zu lassen von der Sprache oder der schlechten Forschungs- und Softwarelage." — Dr. Jonas Müller-Laackman*

Die Transliteration der DMG wurde 1935 auf dem 19. internationalen Orientalistenkongress in Rom konzipiert und im selben Jahr in der Denkschrift "Die Transliteration der arabischen Schrift in ihrer Anwendung auf die Hauptliteratursprachen der islamischen Welt" festgehalten. [9] Danach hat sie jahrzehntelang "die islam-wissenschaftlichen Publikationen weltweit dominiert" [3] und war beispielgebend für alle weiteren Normen für die Transliteration des Arabischen in lateinische Schriftsysteme; zum Beispiel DIN 31635 und ISO 233. [2]

Doch 1935 gab es noch keine Computer. Die DMG-Transliteration wurde 1935 für die Handschrift entworfen und ist nicht für das digitale Zeitalter und moderne Computer ausgelegt. Arne Ambros geht noch einen Schritt weiter:

> *"[Das System der DMG, R. H.] war im Grunde bereits im Jahre seiner Proklamation veraltet, setzte es doch zu einer Zeit, da die Schreibmaschine ihren Siegeszug bereits angetreten hatte, den mit der Hand schreibenden Gelehrten voraus, dem zur Verbreitung seiner Forschungen über einen allerengsten Kreis hinaus ein in anonymem Dunkel wirkendes Heer von Spezialisten (Schriftsetzern und Buchdruckern) zu assistieren hatte." — Arne Ambros*

Die ersten Computer verbesserten die Situation nicht. Denn nahezu die Hälfte der arabischen Sprachlaute könnten, so Ambros, aufgrund der Zusatzzeichen mit einer gewöhnlichen Tastatur nicht getippt werden [Er redet dabei eigentlich über Schreibmaschinen, aber es gilt das Gleiche für Computertastaturen]. [3] Die Situation hat sich in den über dreißig Jahren seit 1991, dem Erscheinungsjahr seiner Publikation "Probleme der Transliteration des Arabischen im EDV-Zeitalter",

nicht wirklich gebessert. Heutzutage hantieren die Orientalisten entweder mit einer arabischen Tastatur und eine optimierten “deutschen” Tastatur, die die vielen nötigen Extrazeichen beinhaltet, oder sie haben sogar drei verschiedene Layouts: eine deutsche, eine arabische und eine für die Transliteration. Alternativ müssen die Zeichen mühsam von irgendwo gecopypasted werden. So zum Beispiel alle in diesem Dokument.

> *“The drudgery of manual romanization is something that scholars in fields like Arabistik and Islamwissenschaft would love to put behind them.” — Dr. Theodore S. Beers* [1]

Die DMG gibt in ihrer Denkschrift Richtlinien zur Transliteration des Arabischen, Persischen und Türkischen. Ich möchte mich jedoch in dieser Bachelorarbeit ausschließlich der Transliteration des Arabischen widmen. Dies aus mehreren Gründen:

1. weil meine Kenntnisse nur im Arabischen ausreichen, um eine korrektes und vollständiges Verständnis zu gewährleisten, welches notwendig ist, um die nötigen Algorithmen zu verstehen und zu entwickeln
2. weil Arabisch die am weitesten verbreitete und damit die Sprache ist, für die es die meisten Tools und Bibliotheken gibt
3. weil alle weiteren Sprachen voraussichtlich den Rahmen dieser Bachelorarbeit gesprengt hätten.

Zusätzlich war es mir – wie bereits erwähnt – wichtig, einen praktischen Nutzen zu erzielen. Aus diesem Grund habe ich ein grafisches Benutzerinterface (GUI), genauer gesagt eine Webanwendung, entwickelt und implementiert. Ziel war es, der Öffentlichkeit meine Ergebnisse unmittelbar und möglichst dauerhaft sowie benutzerfreundlich zur Verfügung zu stellen[2].

II. Forschungsstand

Bereits vor der Erwägung dieses Themas für meine Bachelorarbeit überprüfte ich, ob das Problem bereits gelöst war. Die vorläufigen Recherchen ergaben ein eindeutiges negatives Ergebnis. Dies ist vermutlich darauf zurückzuführen, dass es sich um ein Nischenproblem handelt, dessen Lösung sowohl Programmier- als auch Arabischkenntnisse sowie ein Interesse an der Transliteration des Arabischen ins Deutsche erfordert.

> *“Anyway, it's very much an open topic in research. You could probably come up with a solution that would be of interest to people, even if it weren't flawless.”*
>
> *— Dr. Theodore S. Beers*

Außerhalb dieser Nische existieren jedoch einige Lösungen. Zunächst hat Google ein eigenes System zur Umschrift entwickelt, welches unter anderem in Google Translate zum Einsatz kommt. Google bietet sogar eine dedizierte URL (google.com/transliterate) an, die mittlerweile auf die „Google Input Tools“ weiterleitet. Dabei nutzt Google die Rückmeldungen seiner zahlreichen Nutzer, indem es sie nach korrekten Transkriptionen fragt. Es ist anzunehmen, dass Google diese Rückmeldungen verwendet, um automatische Transliterationsalgorithmen zu trainieren.

---

[1] https://www.theobeers.com/

[2] transliteration.eu.pythonanywhere.com

Des Weiteren hat die Arabic Alphabet Academy ein Transliterationstool entwickelt. [13] Dieses ist jedoch auf die ALA-LC-Romanisierung (American Library Association & Library of Congress) ausgerichtet, welche das Pendant zum DMG-Standard für den US-amerikanischen Bereich darstellt. [19] Die Autoren beschreiben ohne große Detailtiefe die Funktionsweise ihres Tools und erwähnen, dass ihre Modelle einen umfangreichen Korpus von Trainingsdaten nutzen, um Genauigkeiten von 94 % bis 95 % zu erreichen. Es bleibt unklar, wie diese Genauigkeiten ermittelt wurden. Zudem geben die Autoren an, Mishkal, eine Diakritisierungssoftware, zu verwenden, wodurch sie die Genauigkeit auf [sic] 6 % erhöht hätten. [20] Zuletzt wird erwähnt, dass das Tool täglich aktualisiert werde.

Diese Bachelorarbeit würde also eine anspruchsvolle Aufgabe darstellen, da die notwendigen Ressourcen fehlen. Diese umfassen:

- Akademisches Wissen, Erfahrung, etc.
- Daten bzw. Nutzer
- Rechenressourcen
- Zeit

III. Methode

Wenn man sich gerade neu mit dem Problem der automatischen Transliteration beschäftigt und sich Tabellen wie zum Beispiel in Abb. 1 anschaut, dann könnte man denken, dass eine Lösung sehr einfach ist; ein einfaches Mapping mit Buchstabenersetzung. Im Großen und Ganzen ist Transliteration auch nichts Anderes, wenn man sich zum Beispiel die Buckwalter-Transliteration anschaut. Sie ist ein perfekt umkehrbares, genaues eins zu eins Transliterationssystem, das hauptsächlich entwickelt wurde, um arabischen Text digital mit (englischen) ASCII-Zeichen zu verfassen und zu speichern. [4] Es ist jedoch nicht wirklich für den menschlichen Leser gemacht, da viele Zeichen genutzt werden, deren Lauttonung nicht offensichtlich ist und die den Lesefluss hemmen.

DIN 31635

arab.	ا	ب	ت	ث	ج	ح	خ	د	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	ف	ق	ك	ل	م	ن	ه	و	ي
Umschrift	ʾ/ā	b	t	ṯ	ǧ	ḥ	ḫ	d	ḏ	r	z	s	š	ṣ	ḍ	ṭ	ẓ	ʿ	ġ	f	q	k	l	m	n	h	w/ū	y/ī
Code	02BE, 0101			1E6F	01E7	1E25	1E2B		1E0F				0161	1E63	1E0D	1E6D	1E93	02BF	0121									016B	012B
IPA	ʔ/aː	b	t	θ	dʒ/g/ʒ	ħ	x	d	ð	r	z	s	ʃ	sˤ	dˤ	tˤ	ðˤ/zˤ	ʕ	ɣ	f	q	k	l	m	n	h	w/uː	j/iː

Abbildung 1: Einfache Übersicht der DIN 31635 [5]

Das System der DMG soll jedoch in der Praxis gelesen werden können und muss daher auch die tonale Ausprägung des arabischen Textes in der Transliteration wiedergeben. Diese Repräsentierung der tonalen Ausprägung ist eine grundlegende Schwierigkeit bei der Transliteration. Deshalb wird das System der DMG auch oft als Mischform zwischen Transliteration und Transkription bezeichnet. [10]

Ein weiteres grundlegendes Problem ist, dass es eigentlich gar keinen so einheitlichen Standard gibt. Die DMG hat vieles offen gelassen, vieles gilt als veraltet und viele Universitäten haben eigene Regeln zur Transliteration, die sie ihren Studenten in Seminaren beibringen, aber glücklicherweise teilweise auch veröffentlichen. Die Regeln variieren sogar innerhalb einer Universität zwischen Instituten und innerhalb Instituten je nach Professor und bei den Professoren abhängig vom Seminar.

> *“Die zentrale Konvention im deutschen Sprachraum ist ein System, das in den 1930er Jahren im Auftrag der Deutschen Morgenländischen Gesellschaft (DMG) erarbeitet wurde, aber erstens nur Empfehlungen enthält, zweitens zum Teil unterschiedliche Varianten anbietet und drittens nicht mehr in allen Punkten aktuell ist.” — Thomas Hildebrandt.* [1; A.4]

Es gibt beispielsweise auch die DIN-Norm 31635, deren Wikipedia-Artikel ganz oben auftaucht, wenn ich auf Google nach “dmg transliteration” suche, denn zumindest “bezüglich Arabisch und Persisch beruht sie auf der Umschrift der Deutschen Morgenländischen Gesellschaft (DMG)” [5] und kann auch in der Praxis kaum unterschieden werden. So kann ich bis zum heutigen Zeitpunkt trotz etwaiger Recherche nicht sagen, ob es überhaupt einen grundlegenden Unterschied gibt und wenn ja, welchen.

Die DMG selbst sagt zu ihrer eigenen Denkschrift, dass sie “für viele Jahrzehnte einen Standard darstellte und auch heute noch oft beachtet wird.” [7] Durch die Nutzung der Vergangenheitsform wird klar, dass die DMG die Denkschrift selbst als veraltet und überholt ansieht. Diese kann also auf jeden Fall nicht als alleinige Quelle herangezogen werden.

Zusammengefasst kann man sagen, dass auch schon die Bestimmung der Anforderungen, also die Definition der DMG-Transliteration, schon ein großer Teil dieser Bachelorarbeit ist. Um diese Anforderungen zu strukturieren, werde ich in den folgenden Kapiteln jeweils die Anforderung bestimmen und dann direkt auch die jeweiligen Lösungen beschreiben, erklären und versuchen zu bewerten. Dadurch verflechte ich also den Problembereich und den Lösungsbereich, um das Lesen dieser Bachelorarbeit für alle Leser so angenehm wie möglich zu gestalten. Die Quellen zur Anforderungsbestimmung sind hauptsächlich die Quellen der DMG von 1935 [9] und deren Ergänzung [14], der Wikipedia, der Universitäten und der rege Austausch mit verschiedenen Experten aus dem deutsch-akademischen Bereich. Ich möchte hier auch schon mal vorgreifen und sagen, dass mir eine quantitative Bewertung der Lösung sehr schwergefallen ist.

Die Programmiersprache der Wahl für dieses Projekt ist Python. Es ist einfach zu schreiben und zu lesen und betont, dass es nicht auf Millisekunden ankommt. Zuletzt ist das Ökosystem ausgereift und enthält viele Bibliotheken für die Verarbeitung von natürlicher (arabischer) Sprache.

IV. Grundlegende Konzepte

Die Buchstaben des arabischen Alphabetes

Ich fange mit dem gleichen Kapitel an, mit dem auch die Denkschrift der DMG beginnt. [9; I.] Dort wird in einer Tabelle jedem arabischen Buchstaben ein oder mehrere Sonderzeichen oder lateinische Buchstaben mit Diakritika zugeordnet. Diese Tabelle bildet die Grundlage der DMG-Transliteration und jegliche Abweichungen beziehen sich immer auf diese Tabelle. Sie ist seit 1935 unverändert geblieben und ist auch in Abb. 1 zur DIN 31635 wiedergegeben. In Abb. 1 wurde jedoch schon die Wiedergabe der Vokale ي و ا als *ā ū ī* berücksichtigt.

Im letzten Abschnitt Lösungsansatz wurde schon erwähnt, dass auch die technische Umsetzung dieser Tabelle sehr einfach ist; zum Beispiel mit einer Hashtabelle. Es gilt der "Grundsatz, in der Transliteration Wörter „Buchstabe für Buchstabe" wiederzugeben". [1] Der resultierende Algorithmus hat eine Laufzeitkomplexität von O(n), wenn n die Länge des zu transliterierenden Textes ist und der Lookup in der Hashtabelle O(1) ist.

Vokale

Das Arabische kennt die drei Vokale A, U und I jeweils in kurzer und langer Form. Die Kurzvokale werden dabei als diakritische Zeichen (َ, ُ und ِ) auf dem zuvorkommenden Konsonanten dargestellt und in der Transliteration als *a*, *u* und *i* wiedergegeben. [9] Wie schon in Die Buchstaben des arabischen Alphabetes erwähnt, stellen die Buchstaben ا (Alif), و (Wāw) und ي (Yā') die jeweiligen Langvokale dar, wiedergegeben als *ā*, *ū* und *ī*. [9] Dabei spielen diese drei Buchstaben eine besondere Rolle als Halbvokale. Das Alif bzw. eigentlich das sogenannte Hamza ist ein glottaler Stopp, sowie etwa das * in "Schüler*innen" und wird im Deutschen normalerweise nicht geschrieben; mehr dazu in Das Hamza. Das Yā' und Wāw sind ähnlich zum lateinischen I und V: Das Yā' ist entweder ein I- oder ein J-Laut, das Wāw ist entweder ein W-Laut (so wie im englischen "**w**on") oder ein U-Laut. Das Yā' hat dabei als Halbvokal auch Ähnlichkeiten zum Englischen Y und wird daher auch so wiedergegeben. (Man möge sich hier nochmal vergegenwärtigen, dass das System der DMG eigentlich als grundsätzlich **internationaler** Standard auf dem 19. **internationalen** Orientalistenkongress vorgestellt, aber dort nicht angenommen wurde [11], sondern erst 1961 in der ISO-Norm 233)

"كون" zum Beispiel könnte sowohl *kawana* als auch *kūn* sein. Das liegt unter anderem daran, dass die Kurzvokale im Arabischen (oft) nicht geschrieben werden. Dazu mehr in Vokalisation.

Diphthonge

Wenn einer der Halbvokale einem *a* innerhalb einer Silbe folgt, dann entsteht ein Diphthong. Die DMG sagt dazu 1935: "Diphthonge au und ai; doch ist auch die Wiedergabe durch aw und ay zulässig." [9] In ihren Ergänzungen unter 5. schreiben sie jedoch "Diphthonge werden stets als aw oder ay notiert, da das w bzw. y hier die konsonantische Silbenkoda bildet: bayt „Haus" oder mawt „Tod"." [14] Diese Ergänzung scheint jedoch, obwohl begründet, nicht allgemein anerkannt zu sein. Die Uni Kiel schreibt beispielsweise: "Diphtonge [sic] werden als ai oder au geschrieben (بَيْت bait, يَوْم yaum). (Die Verwendung von ay und aw wird nicht empfohlen.)" [10; 3.] Siehe auch Abb. 2.

Es gibt also hier zwei Möglichkeiten, die beide zulässig sind. Wir kommen außerdem auch zu einer Erkenntnis: Software (und natürlich auch Hardware) kann feste Standards so erzwingen, wie es eigentlich nichts Anderes kann. Daher hat der Entwickler eine große Verantwortung, die verschiedenen Möglichkeiten zu beachten, um nicht eine davon zu verdrängen oder, allgemeiner, zu

benachteiligen. Es folgt also auch die erste systemische Anforderung, nämlich dass der Nutzer selbst wählen können muss, wie Diphthonge transliteriert werden sollen.

Zusatzzeichen

Im Arabischen gibt es noch einige Zusatzzeichen.

- Das Sukūn ist ein Nullvokal. Es signalisiert also, dass nach einem Konsonanten kein Kurzvokal kommt, "und wird nicht wiedergegeben." [15]
- Die Nunation (Tanwīn; ً, ٌ, ٍ) steht im Arabischen am Ende von unbestimmten Substantiven und wird mit -an, -un und -in wiedergegeben.
- Das Schadda (ّ) deutet auf die Verdopplung (Gemination) eines Konsonanten [16] und wird dementsprechend auch so transliteriert: "مُحَمَّد Muḥammad" [13; C.1]
- Das Hamza hat viele verschiedene Schriftformen und zwei verschiedene semantische Formen. Wir werden in Das Hamza intensiver einsteigen.
- Das ة (tāʾ marbūṭa) ist ein besonderer Buchstabe zwischen ت (tāʾ; gleiche Punkte) und ه (hāʾ; gleiche Form), der im Normalfall weibliche Wörter markiert und manchmal t und manchmal h gelesen wird. Weiteres in Das Tā' marbūta.
- Das Alif maqsūra (ى) kann in der Transliteration genau wie ein Alif behandelt werden. [5]
- Das Madda (ٓ) ist äquivalent zu einem Hamza und einem Alif. [17]

Beim Shadda auf einem der beiden Halbvokale gibt es ähnlich zu den Diphthongen zwei verschiedene Möglichkeiten (konsonantisch und vokalisch), die auch hier in den unteren beiden Spalten von Abb. 2 dargestellt sind. Die Denkschrift der DMG sagt dazu: "In Fällen von verdoppeltem Wāw oder Yā' ist das erste als Vokalbezeichnung zu behandeln [...]; doch ist auch die voll konsonantische Wiedergabe zulässig [...]." [9] Auch hier ist es also notwendig, dass der Nutzer seine Präferenz wählen kann.

ـَيْ	ai (ay)
ـَوْ	au (aw)
ـِيّ	īy (iyy)
ـُوّ	ūw (uww)

Abbildung 2: Ausschnitt aus "Vokale (*ḥarakāt*) und Diphthonge" [11][15]

Spätestens jetzt ist klar, dass eine einfache Buchstabenersetzung nicht mehr ausreicht. Stattdessen kann man jedoch, abhängig von den Nutzerpräferenzen, eine Reihe von regulären Ausdrücken über den Text laufen lassen, um die entsprechenden Regeln anzuwenden. Hier ein paar Beispiele:

- Für das Shadda: `"(.)" + shaddah: lambda match: match.group(1) * 2`

  Wir matchen also auf irgendeinen Buchstaben (`.`) und ein Shadda und nutzen dann eine Lambda-Funktion (`lambda match`), um stattdessen den Buchstaben einzusetzen, der in der ersten Gruppe aufgefasst wurde (`match.group(1)`), aber zweimal (`* 2`).

- Für die vokalische Wiedergabe, eines verdoppelten Halbvokals:

  ```
  double_vowels_map = {
  ```

```
        "uww": "ūw",
        "iyy": "īy",
    }
```

Das Hamza

“Dī visenšaftliḩe 'umšrift des ʿarabišen nāḫ dēn rēgeln der doičen morgenlendišen gezelšaft” nennt Hamza “[d]ie große Ausnahme”. Es erklärt weiter: “Phonetisch ist ein Hamza ein sogenannter *glottal stop* (Stimmansatz) [IPA /ʔ]. Im Deutschen tritt er z.B. bei Worten wie *Theater* usw. auf. (Orientalistisch würde man die Aussprache als *tē’āter* wiedergeben.) Im Arabischen ist Hamza ein eigenständiger Konsonant mit Lautwert. Die arabische Grammatik unterscheidet zwischen zwei Typen von Hamza: dem *hamzat al-qaṭʿ* (“Trennungs-Hamza”) und dem *hamzat al-waṣl* (“Verbindungs-Hamza”).” [15]

hamzat al-qaṭʿ

Das *hamzat al-qaṭʿ* kommt überall im Wort vor und wird als ʾ wiedergegeben. Wenn es jedoch am Anfang des Wortes vorkommt, dann muss es nicht wiedergegeben werden. Die DMG schreibt: “Hamzah wird nur im Innern und am Ende eines arabischen Wortes wiedergegeben, am Wortanfang nicht”. [9; p. 10] Das ergibt Sinn, “weil sich der Stimmabsatz [wahrscheinlich Stimmansatz, R.H.] in der Aussprache von selbst ergibt”. [1; C.2.1]

In ihren Ergänzungen schreiben sie jedoch: “hamzat al-qaṭʿ wird konsequent notiert, da es ja ein arabisch-semitisches Phonem repräsentiert[...] Ausnahmen hiervon sind im Anlaut von Eigennamen möglich”. [14] Auch das macht Sinn, denn sonst kann man den Unterschied zwischen den beiden Hamzas nicht mehr erkennen und die Eindeutigkeit geht ohne ausreichende Sprachkenntnisse verloren. [1] Auch hier brauchen wir also eine Einstellung, ob das hamzat al-qaṭʿ am Beginn eines Wortes wiedergegeben werden sollte.

hamzat al-waṣl

Das *hamzat al-waṣl* kommt nur zu Beginn eines Wortes vor und wird selbst nie wiedergegeben. Es ist lediglich ein Träger für einen Kurzvokal. Dieser wird jedoch auch nicht gelesen, wenn das vorhergehende Wort auf einem Vokal endet. [13; C.2.2] Außerdem kann der Kurzvokal auch von dem vorherigen Wort beeinflusst werden. Wie schauen uns zum Ende ein paar einleuchtende Beispiele an. Außerdem hat das Hamza **viele** verschiedene Schreibformen (ا , ئ , ى , إ , أ , ء , ٱ , ؤ , usw.). Zu Zwecken der Transliteration können diese jedoch alle fast gleich behandelt werden; außer:

Das sogenannte Alif waṣl (ٱ) ist ein besonderes Zeichen, dass explizit ein *hamzat al-waṣl* anzeigt. Stattdessen wird jedoch oft das gewöhnliche Alif (ا) verwendet. Das häufigste Vorkommen findet man wahrscheinlich beim Artikel (ال). Bei einem explizit geschriebenen Hamza, also in allen anderen Fällen, kann man davon ausgehen, dass es sich um ein *hamzat al-qaṭʿ* handelt.

Beispiele

- imām (إمام) “Imam” beginnt mit einem *hamzat al-qaṭʿ*
- intifāḍa (انتِفاضَة) “Aufstand” beginnt mit einem *hamzat al-waṣl*. Dieses wird selbst nicht wiedergegeben, sondern nur das *i*, was standardmäßig auf einem *hamzat al-waṣl* steht
- uḫruǧ (اخرُج) “Geh!” Wenn der Kurzvokal auf dem übernächsten Konsonanten ein *u* ist, dann wird auch der Kurzvokal auf dem *hamzat al-waṣl* zum *u*
- li-ntifāḍa (لِانتِفاضَة): Wenn wir nun ein Wort (li-) davor tun, dann wird der Kurzvokal nicht mehr benötigt und er verschwindet
- al-intifāḍa (الانتِفاضَة): Außer das Wort davor (al-) endet auf einem Konsonanten
- li-imām (لِإمام): Beim *hamzat al-qaṭʿ* ist es komplett irrelevant, was davor ist

- ʿan il-bayt (عَن البَيت): “Bei Anschluss an vokallos auslautende Personalpronomen, Präpositionen oder Partikel wird als Hilfsvokal in den meisten Fällen ein „i“ eintreten (und in der Transkription statt des hamzat waṣl geschrieben)” [11; 6.2]
- hum ul-bayt (هُم البَيت): Bei *hum*, sowohl alleinstehend als auch suffigiert, und *antum* ist der Hilfsvokal stattdessen *u* [11]
- min al-madīna (مِن المَدينَة): Bei *min* ist es so, als wäre kein Wort davor [11]
- allaḏīna (الَّذينَ): Einige Wörter haben einen “versteckten” Artikel

Algorithmus

Als möglichst einfache Heuristik, bietet sich folgender Algorithmus für Wörter mit *hamzat al-waṣl* an: Wenn vor dem Wort ein anderes Wort steht, das auf einen Vokal auslautet, dann verschwindet das Alif (waṣl) und jeglicher Kurzvokal darauf. Sonst, muss es einen Kurzvokal geben:

- Der Kurzvokal, der dort vielleicht schon steht, denn dann wurde er explizit geschrieben und muss beachtet werden
- sonst, *u* nach hum oder antum
- sonst, *i* nach einem vokallos auslautenden Stopword
- sonst, *u* wie bei *uḫruǧ* erwähnt
- sonst, *a*, wenn der nächste Buchstabe ein *l* ist. Das behandelt erstens sowohl “versteckte” als auch unbemerkte Artikel (s. Präfixe)
- sonst *i*

Die einzige Schwäche ist ein Wort, welches eigentlich mit einem *il* beginnen müsste, also mit einem *hamzat al-waṣl* gefolgt von einem *l*, worauf ein Buchstabe folgt, der kein *u* trägt.

Das Tā' marbūta

Das Tā' marbūta ist, wie schon in Zusatzzeichen erwähnt, eine besondere Mischung aus Tā' und Hā' und in der Regel weibliche Worte markiert. Das Tā' marbūta kann, entsprechend der Aussprache, entweder als *t* oder als *h* oder gar nicht transliteriert werden. [27] Tatsächlich haben wir alle diese Varianten der Umschrift in diesem Dokument schon gesehen, und zwar bei Hamza (همزة). Dieses endet nämlich auf einem Tā' marbūta (Buchstaben im Arabischen sind feminin). Ich habe es hier, angelehnt an Wikipedia und die gängige Aussprache, gar nicht wiedergegeben. Am Ende von *marbūta* ist genau so ein “unsichtbares” Tā' marbūta. In der Denkschrift der DMG wird jedoch “Hamzah” geschrieben, wie es im zweiten Paragrafen von Das Hamza zu sehen ist. Noch davor haben wir die Transliteration mit *t* kennengelernt: *hamzat al-qaṭʿ* und *hamzat al-waṣl*. Die Regeln zu Tā' marbūta können wir kurz so zusammenfassen: [9]

- Wenn irgendetwas dem Tā' marbūta folgt, dann muss es als *t* wiedergegeben werden
- Wenn das Wort, welches auf Tā' marbūta endet, der “Regens eines folgenden Genitivs” – oder in anderen Worten – das erste Wort einer Genitivkonstruktion ist, dann muss es als *t* wiedergegeben werden
- Sonst, wenn das Tā' marbūta einem langen Vokal (*ā ū ī*) folgt, dann muss es als *h* wiedergegeben werden. “[D]ie selbst in wissenschaftlichen Abhandlungen üblich gewordene Wiedergabe [mit *āt*] sollte fallen gelassen werden!”
- Sonst darf es wahlweise entweder als *h* oder “, der Vulgärsprache folgend,” gar nicht wiedergegeben werden. Es ist, denke ich, relativ einfach zu verstehen, dass ein H-Laut am Ende eines Wortes oft kaum zu verstehen ist

Beim Beispiel des Hamzas von oben könnte man *hamzat al-waṣl* etwa als “Verbindungs-Hamza” übersetzen, aber noch “genauer” wäre die Übersetzung als “Hamza der Verbindung”. Hier kann man

auch im Deutschen die Genitivkonstruktion gut erkennen und auch, dass “Hamza” das erste Wort dieser Genitivkonstruktion ist. Aus diesem Grund muss also das Tā' marbūta – wieder entsprechend der Aussprache – als *t* transliteriert werden

Dazu ein kurzer Einstieg in die arabische Grammatik: Anhand des letzten Kurzvokals eines deklinierbaren Wortes (Flexionsendung; der sogenannte Iʿrāb) kann man den Fall und die Bestimmtheit erkennen. *u* zeigt den Nominativ an, *i* den Genitiv und *a* den Akkusativ. Eine Nunation zeigt an, dass das Wort unbestimmt ist. Ein Wort ist dementsprechend genau dann Regens eines folgenden Genitivs, wenn folgende Bedingungen erfüllt sind:

1. Das Wort ist ein bestimmtes Nomen, aber trägt keinen Artikel
2. Das folgende Wort ist auch ein Nomen, endet aber auf *-i* oder *-in*

Dazu muss sichergestellt werden, dass zwischen dem jeweiligen Wort und dem nächsten kein anderes, vor allem suffigiertes oder präfigiertes, Wort außer einem möglichen Artikel steht. Die korrekte Transliteration des Tā' marbūta stellt damit eines der größeren wortübergreifenden grammatikalischen Probleme dar.

Glücklicherweise konnte ich auf die Arbeiten von Taha Zerrouki aufbauen, der im Rahmen seiner Doktorarbeit einige Python-Bibliotheken entwickelt hat. Eine von ihnen, Qalsadi, erlaubt für jedes Wort die Bestimmung von Wortart, Fall und Suffixen und Präfixen. [20][40] Leider ist sie nicht immer ganz zuverlässig und kann insbesondere auch nicht zwischen echten Nomen und Adjektiven unterscheiden.

Ein anderes Problem, das unter anderem auch das Yā' hat, ist, dass “des Öfteren – sowohl im handgeschriebenen als auch im gedruckten Schriftbild – […] die diakritischen Punkte fortgelassen werden.“ [27] Hier wird auf den Nutzer vertraut, diese nötigenfalls zu ergänzen.

Präfixe

Im Arabischen gibt es einige Wörter, die an andere angehängt werden. Für die Transliteration relevant sind hauptsächlich die Präfixe. Denn diese müssen mit einem Bindestrich vom Hauptwort getrennt werden. Zunächst gibt es die beiden Konjunktionen *wa* und *fa* und die seltene Fragepartikel *a*. Diese können vor jedem Wort und auch vor weiteren Präfixen stehen, aber nur jeweils eins davon. Dann gibt es das Futurpräfix *sa*, welches nur bei Verben vorkommt. Zuletzt gibt es noch einige Präpositionen *bi*, *li*, *ka* und den Artikel *al*. [8; Anmerkungen] Die DMG erwähnt auch noch die Schwurpartikel *ta-* und *la-* [9; p. 14], welche jedoch ausgesprochen selten sind. Aufgrund ihrer ausgesprochenen Seltenheit und die möglicherweise auftretenden Komplikationen ist es sinnvoll, sowohl diese beiden als auch die Fragepartikel zunächst auszuklammern. Der Nutzer muss gegebenenfalls einen Bindestrich einfügen, oder ein *li* in ein *la* umwandeln.

Der Artikel muss außerdem auch korrekt assimiliert werden. Er beginnt mit einem *hamzat al-waṣl*, welches den Kurzvokal *a* trägt. Aber auch das *l* muss assimiliert werden. [9; p. 13] Bei den 14 sogenannten Sonnenbuchstaben wird es in den entsprechenden Buchstaben umgewandelt. [28] Beispiel: *wa-li-ṣ-ṣadīq* (وَلِلصَّديق; eigentlich *walilṣadīq*) “und dem Freund”. [14; 7.]

Die Schwierigkeit hierbei ist die korrekte Erkennung der Präfixe. Nicht jedes *sa* am Anfang eines Wortes stellt notwendigerweise das Futurpräfix dar. Auch die Qalsadi-Bibliothek kann die Präfixe nicht zuverlässig erkennen und hat außerdem eine etwas andere Definition der Präfixe (die ich außerdem nicht genau kenne).

Ich hatte zu einem früheren Zeitpunkt einen Algorithmus verwendet, der sehr aggressiv Bindestriche gesetzt hatte und habe sofort viele Hinweise von Nutzern darauf erhalten. Dies hat offenbart, dass

ein falsch-positives Ergebnis in diesem Fall die Nutzerzufriedenheit weitaus mehr beeinträchtigt als ein falsch-negatives. Daher sollte eine gewählte Lösung die Bindestriche eher konservativ setzen.

Die Flexionsendungen

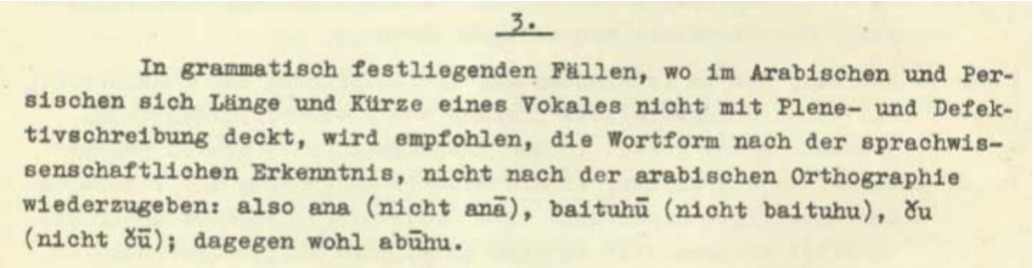
Bei der Transliteration gibt es zwei grundlegend verschiedene Arten zu transliterieren: die Pausal- und die voll vokalisierte Form. [1; C] Bei der gebräuchlicheren Pausalform werden die Flexionsendungen weggelassen außer bei: [10; 11.][1]

- Verben
- Präpositionen
- Pronomen, auch suffigiert
- Konjunktionen
- bei Akkusativendungen [1] bzw. nur bei adverbialem Gebrauch [10]
- bei indeterminierten Präsenzpartizipien hinten schwacher Wurzeln

Aus informatischer Sicht werden Präpositionen, Pronomen und Konjunktionen alle als sogenannte Stopwords zusammengefasst. Auch die Erkennung von Verben und Akkusativendungen ist möglich. Die Erkennung von “indeterminierten Präsenzpartizipien hinten schwacher Wurzeln” schien mir jedoch zu schwer, alleine weil mir dafür die nötigen Arabischkenntnisse fehlen.

Weiteres

Es gibt bei der Transliteration auch einige besondere Formen, die berücksichtigt werden müssen. Bei diesen weicht die Aussprache von der orthografischen Schreibweise ab oder die gewöhnlichen Transliterationsregeln würden fehlgreifen. Weil der ursprüngliche arabische Text immer noch klar erkennbar ist, darf hier von der Einheitlichkeit abgewichen werden (s. Abb. 3). [11; p. 2]



3.

In grammatisch festliegenden Fällen, wo im Arabischen und Persischen sich Länge und Kürze eines Vokales nicht mit Plene- und Defektivschreibung deckt, wird empfohlen, die Wortform nach der sprachwissenschaftlichen Erkenntnis, nicht nach der arabischen Orthographie wiederzugeben: also ana (nicht anā), baituhū (nicht baituhu), ǒu (nicht ǒū); dagegen wohl abūhu.

Abbildung 3: Dritte Regel der DMG zur Vokalisation [9; p. 12]

Unregelmäßige Längen

Vokale, die im Arabischen (meist aus historischen Gründen) nicht lang geschrieben, aber lang gesprochen werden, werden grundsätzlich lang transkribiert:

هذا	hāḏā	dieser
ذلك	ḏālika	jener
لكن	lākin/lākinna	aber
الله	Allāh	Gott
الرحمن	ar-Raḥmān	der Barmherzige
طه	Ṭāhā	Ṭāhā

Abbildung 4: Langschreibung nicht geschriebener langer Vokale [1; C.13.1]

Zunächst gibt es manche Worte, die mit einem Alif gesprochen werden, das jedoch nicht geschrieben wird: *hāḏā*, *ḏālika*, *lākin/lākinna*, *Allāh*, *raḥmān*, *Ṭāhā*, (s. Abb. 4) und *kaḏālika*. [1; C.11.1] Wikipedia fügt außerdem noch hinzu: *hāḏihi* und *hākaḏā*. [35] Gleiches gilt für Ibrāhīm, auch wenn das Alif mittlerweile meist geschrieben wird. [10; 20]

Dann gibt es noch das genau gegensätzliche Phänomen, nämlich dass ein Alif oder Wāw geschrieben, aber nicht gesprochen wird: "*ana* (nicht *anā*)" (s. Abb. 3)[3], *ʿAmr* nicht *ʿAmrū* [10; 20], *ulāʾika* nicht *ūlāʾika* [10; 12]. Zudem muss die Verbalendung *Wāw-Alif* beachtet werden, bei der das finale Alif nicht gelesen wird: *qālū* nicht *qālūā* oder Ähnliches. [1; C.9]

Die Längung des Personalpronomens der dritten Person Singular Maskulin

Die allermeisten Quellen sind sich einig, dass das suffigierte Personalpronomen der 3. P. S. M. seiner Aussprache nach wiedergegeben werden soll (s. Abb. 3), aber nicht muss. Dabei gilt, dass das Personalpronomen verlängert wird, wenn es einer kurzen Silbe, folgt. Praktisch heißt das, dass es verlängert wird, wenn es nicht einem Langvokal folgt.

In Kiel denkt man jedoch: "Die Längung des Personalpronomens der 3. Pers. Sing. Mask. nach kurzen Silben wird in der Umschrift nicht berücksichtigt: كِتَابُهُ kitābuhu (statt kitābuhū)". [10; 16]

Nisba-Suffix

Ein weiterer Sonderfall ist auch das sogenannte männliche Nisba-Suffix, das Zugehörigkeit angeben soll (Wir werden es bei den Eigennamen gleich nochmal sehen). Es soll nicht als *īy* oder *iyy* wiedergegeben werden, sondern lediglich als *ī*. Also *al-ʿarabī* (Araber oder arabisch; Nisba), aber *nabīy* o. *nabiyy* (Prophet; keine Nisba), obwohl beides auf *iyy* endet. [1; C.8] Die DMG empfiehlt stattdessen, dass man bei verdoppeltem Endwāw oder -yā' immer mit *ī* transliteriert. [9; p. 15]

Alle weiteren Assimilationen sollen nicht berücksichtigt werden, "da das sonst mehr Verwirrung als Klarheit schafft". [13; C.12.2]

Eigennamen

Im Arabischen gibt es keine Groß- und Kleinschreibung. In der deutschen Umschrift sollen Eigennamen jedoch großgeschrieben werden: Muḥammad (محمّد). Ein sehr einfacher Ansatz ist, Named Entity Recognition (NER; Eigennamenerkennung) zu verwenden.

Dazu hatte ich zunächst einen Transformer von Huggingface benutzt [34]. Dieser konnte Namen zwar im Allgemeinen gut erkennen, war aber von der Geschwindigkeit her nicht ausreichend für eine flüssige UX und war insbesondere sehr ressourcenhungrig, was das Hosting auf einem Gratisdienst, wie pythonanywhere.com verbot. Dies ist zwar nicht absolut notwendig, aber wünschenswert. Daher machte ich mich auf den Weg, um ein anderes, einfacheres, vorhersagbares und vor allem effizientes System zu entwickeln. Ich wusste aber, dass ich zur Not darauf zurückgreifen könnte, falls mir nichts Besseres einfiel.

Meine erste nächste Idee war wieder ganz einfach anzufangen: Ich habe mir einen manuell erstellten NER-Datensatz mit Texten von Wikipedia heruntergeladen [29] und einfach alle Wörter ausgewählt, die nur als Namen auftauchen. In einem Text können dann alle solche Wörter als Namen markiert werden. Ich erhoffte mir dadurch eine sichere Basislinie, die mit unglaublich geringer

[3] Thomas Hildebrandt widerspricht dem: "Ein Alif am Wortende wird grundsätzlich lang transkribiert, auch wenn es, wie es bisweilen der Fall ist, kurz gesprochen wird" [1; C.5] Widerspricht sich dabei jedoch selber.

Wahrscheinlichkeit ein falsch-positives Ergebnis erzeugen würde. Das war für mich an der Stelle deshalb wichtig, weil ich bei der Präfixerkennung zumindest gelernt hatte, dass ein falsch-positives Ergebnis den User oft viel mehr stört als ein falsch-negatives. Nun war es aber so, dass diese Lösung überhaupt kaum positive Ergebnisse hatte.

Es gibt natürlich erstmal auch ein systematisches Problem. Nämlich, dass diese Methode Namen wortweise bestimmt. Für jedes Wort wird also bestimmt, ob es Teil eines Namens ist oder nicht. Dabei sind Namen im Allgemeinen natürlich wortübergreifend. Zur weiteren Analyse können wir zunächst zwischen persönlichen und unpersönlichen Namen unterscheiden.

Unpersönliche Namen

Bei unpersönlichen Namen wird meist die Kategorie des Namens in einer Genitivverbindung mit dem Namen verbunden. Man könnte es vielleicht im Deutschen vergleichen mit sowas wie zum Beispiel "die Stadt Berlin(s)" oder die "die Stadt von Berlin", oder "das Buch (von) der Analysis". Dies ist jedoch im Deutschen nicht (mehr) gebräuchlich und oft wird einfach der Name erwähnt. Im Arabischen sind diese Kategorien wie كِتاب (kītāb; Buch), مَدِنَة (madinah; Stadt) oder شَرِكَة (šarikah; Firma) aber Hinweise auf darauffolgende Namen und daher ergibt es Sinn, sie in der weiteren Analyse mit aufzunehmen. Die Denkschrift zum DMG-Standard sagt dazu auch:

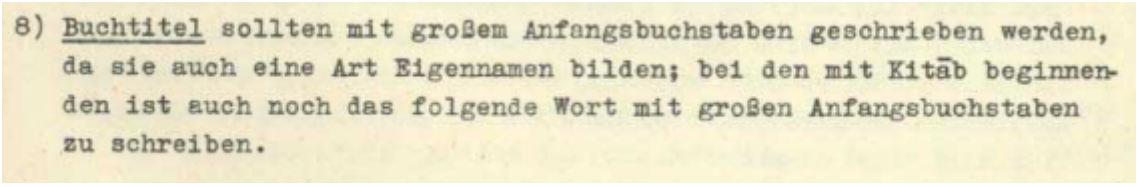

8) Buchtitel sollten mit großem Anfangsbuchstaben geschrieben werden, da sie auch eine Art Eigennamen bilden; bei den mit Kitāb beginnenden ist auch noch das folgende Wort mit großen Anfangsbuchstaben zu schreiben.

Abbildung 5: DMG-Regeln zur Großschreibung von Buchtiteln [9; p. 15]

Auch hier muss ein Programm, welches diesen Standard erfüllt, *kītāb* besonders behandeln. Gleiches gilt für andere ähnliche Wörter, ohne dass ich hier eine abschließende Liste angeben könnte.

Persönliche Namen

Wikipedia gibt fünf grundlegende Bestandteile arabischer Namen an: *ism*, *kunya*, *nasab*, *nisba* und *laqab* (s. Tabelle 1). Die europäischen Namen aus Vor- und Familiennamen sind in den arabischen Ländern erst seit der Kolonialzeit, insbesondere seit dem 20. Jhd., üblich. [25]

Teil	Titel (*laqab)*	Vater von/ Mutter von (*kunya*)	Persönlicher Name (*ism)*	Abstammung (*nasab*)	Herkunft (*nisba)* s. Nisba-Suffix	Ehrenname (*laqab)*
Beispiel	*ustāḏ* Professor	*abū Mūsā*	*Muḥammad*	*ibn Baṭṭūṭa*	*al-Fārisī* der Perser	*al-ḥāǧǧ* [38] Pilgerer

Tabelle 1: Die verschiedenen Bestandteile eines traditionellen arabischen Namens [25]

Ein paar Beispiele:

- **Muhammad ibn Ismāʿīl ibn Ibrāhīm ibn al-Mughīra al-Buchārī al-Dschuʿfī**, besser bekannt als **Imam al-Buchārī**. "Imam" ist ein Titel, "Muhammad" ist sein persönlicher Name, "al-Buchārī" heißt derjenige, der aus *Buchārā* kommt, etc. [36]
- **Abū ʿAbdallāh Muhammad ibn Battūta**, besser bekannt als **Ibn Battūta** [37]
- **DMG: *Ǧalāl ad-Dīn Muḥammad b. Šaiḫ Bahā' ad-Dīn Muḥammad b. Ḥusain-i Balḫī-yi Rūmī***, besser bekannt als Rumi [39]

- **Ibn Sīnā** (Avicenna), **al-Chwarizmi** (Algorismi), etc.

Arabische Namen sind sehr kompliziert und historische Persönlichkeiten sind oft unter verschiedenen Arten von Namen bekannt. Menschen, die jung Bekanntheit erlangt haben, sind oft nach ihren Eltern benannt (Ibn Battūta) und Menschen, die später Bekanntheit erlangten, werden oft nach ihren Kindern benannt, um ihre Elternschaft zu ehren (Abū Tālib), andere nach ihrer Herkunft oder gewissen Ehrennamen.

Im Deutschen oder Englischen sind Namen auch ohne die Großschreibung meist einfach vom restlichen Text zu unterscheiden. Im Englischen sieht man manchmal (Mädchen)namen wie Rose, Summer, Joy, Hope, Robin, aber auch unpersönliche Namen wie Apple. Natürlich haben auch andere Namen im Deutschen oder im Englischen wie Magnus, Alexander, Emelie oder Philipp Bedeutungen, aber dann aus fremden oder toten Sprachen (Altgermanisch, Latein, Altgriechisch, Hebräisch, etc.).

Doch im Arabischen haben die allermeisten Namen eine Bedeutung **im Arabischen**. Muḥammad ist der häufigste Name der Welt, [26] bedeutet aber eigentlich einfach nur der Gepriesene oder gepriesen. [25] *Mālik* heißt etwa König, *Amīr* Prinz und *Zahra* (nicht zu verwechseln mit Sara) Blume.

هُوَ كَرِيم bedeutet “Er ist großzügig“, aber genauso gut “Er ist Karim”. Die Transliteration könnte also *huwa karīm* oder *huwa Karīm* sein. Es ist absolut unmöglich (also auch als Mensch) in diesem isolierten Beispiel, eine “richtige” Entscheidung zu treffen. Es kann nur aus dem Kontext geschlossen werden. In den allermeisten Fällen ist NER jedoch für einen Menschen relativ einfach. Ein regelbasierter Ansatz wie der oben beschriebene ist hier jedoch hoffnungslos verloren.

Eine besondere Lösung

Es ist also aus verschiedenen Gründen ziemlich schwer, mitten im Text komplizierte Namen zu erkennen. Fehler wären unmöglich zu vermeiden und dadurch würden die Erwartungen nicht erfüllt werden. Mir wurde empfohlen, stattdessen Namen separat zu transliterieren. Das ist deswegen sinnvoll, weil Namen oft einzeln transliteriert werden müssen. Entweder in einen deutschen Text eingebunden, oder sie werden in Referenzen oder bibliografischen Einträgen erwähnt:

> “Anfang der Beschreibung von Baġdād [Bagdad] nach al-Qazwīnī, Āṯār al-bilād, ed. Wüstenfeld, S.209” — Denkschrift [9; p. 16]

Dies hat einige enorme Vorteile bei der Implementierung, da ganz klar ist, dass die gesamte Eingabe ein einziger Name ist, und man sich so auf die Besonderheiten von Namen fokussieren kann. Die Erwartungen des Nutzers sind so auch besser zu erfüllen. Leider konnte ich für diese besondere Transliteration von Namen noch kein eigenes UI bauen, aber es ist implementiert.

Man muss nur auf ein paar Dinge achten. Bei Personen- und Ortsnamen alles groß [9; p.15], bei Buchtiteln wie erwähnt (s. Abb. 5), *ibn* bzw. *bin* und *bint* nur groß, wenn es am Anfang steht, sonst klein oder abgekürzt *b.* und *bt.* [9] Das Schwierigste ist die Zusammenschreibung von Namen. Sie wird verschieden definiert:

- “Häufig vorkommende Zusammensetzungen (vor allem solche mit Abū, ʿAbd, und mit ad-dīn und Allāh)” [9]
- An der Uni Kiel soll nur *ʿAbdallāh* und *ʿUbaidallāh* geschrieben werden, aber auch eine Zusammenschreibung bei *ʿAbd* wird erlaubt. [10; 19]
- Die Uni Leipzig umschreibt die DMG-Denkschrift, nimmt aber *Abū* aus. [11; 9]
- Thomas Hildebrandt empfiehlt die Zusammenschreibung nur bei *Allāh* [1; C.14]

“Häufig vorkommend” ist natürlich sehr subjektiv und kann von Person zu Person variieren. Alle sind sich jedoch einig über *ʿAbdallāh*, also ist es sicher, zumindest das zu behandeln.

Vokalisation

In arabischer Prosa werden die diakritischen Zeichen, also insbesondere die Kurzvokale **nicht** geschrieben. Ein Arabischsprechender kann diese Texte jedoch trotz Ambiguitäten lesen. Manchmal werden in formalen Texten zur Erhaltung der Klarheit dennoch einige Zeichen geschrieben. In Gedichten oder zum Beispiel im Koran werden alle diakritischen Zeichen geschrieben. Man kann also zwischen unvokalisierten, teil-vokalisierten und voll-vokalisierten Texten unterscheiden.

Vokalisation (auch Diakritisierung) beschreibt die Umwandlung von unvokalisierten oder teil-vokalisierten Texten in voll-vokalisierte Texte. Dies ist für die Transliteration notwendig, denn diese muss ja auch die Kurzvokale enthalten, um gelesen zu werden. *Muḥammad* kann ohne Vorkenntnisse gelesen werden, *Mḥmd* nicht und ist daher keine valide Transkription. [13; A.3] Für automatische Transliteration ist also auch automatische Vokalisation notwendig. Es ist jedoch auch erwünscht, dem Nutzer Introspektion zu geben und ihm zu erlauben, mögliche Fehler im arabischen Text zu korrigieren. Deswegen wird die Vokalisation auf Knopfdruck eines “magischen” Knopfes (s. Abb. 6) auf den arabischen Text im Eingabefeld angewandt. Das hat noch einen weiteren Vorteil: Die Erwartung an die Zeit, die die Vokalisation brauchen darf, ist nicht im Bereich von Millisekunden, sondern von Sekunden, was also viel kostspieligere Lösungen erlaubt. Zum Beispiel erlaubt es die Verwendung von neuronalen Netzen und HTTP-Aufrufen.

Abbildung 6: Ein “magischer” Knopf

Es muss auch hier klar sein, dass eine regelbasierte Vokalisation nicht zufriedenstellende Ergebnisse liefern kann. Taha Zerrouki hat jedoch in seiner Dissertation eine solche regelbasierte automatische Vokalisation mit dem Namen Mishkal präsentiert. [20] Es wurde in Ähnliche Arbeiten beim ALA-LC-Transliterationstool der Arabic Alphabet Academy schon erwähnt.

> *“It actually does a decent job most of the time.” — Dr. Theodore S. Beers*

Ich möchte bei Mishkal auch nicht allzusehr ins Detail gehen, denn obwohl es eine valide Lösung ist, scheint eine intelligente(re) Lösung hier angesichts der Anforderungen sinnvoller. Farasa ist laut eigenen Angaben “the state-of-the-art full-stack package to deal with Arabic Language Processing.” Sie bieten unter anderem auch Vokalisation als REST-API an. Leider war diese unbrauchbar, weil sie jedwede Teilvokalisierung einfach ignorierte. Stattdessen habe ich die nächste KI-basierte Lösung genutzt: Shakkala. [30] Ein Paper aus 2019 kommt zu dem Schluss, dass Shakkala eindeutig besser sei als alle anderen untersuchten Ansätze. Bei dem Vergleich wurde auch Farasa mitberücksichtigt. Außerdem sei Mishkal das Beste unter den nicht-neuronalen Ansätzen. Dennoch könne seine Leistung nicht mit der von Shakkala verglichen werden. [32]

V. Large Language Models

Künstliche Intelligenz (KI), Machine Learning (ML) und insbesondere Neuronale Netzwerke (NNs) haben sich als sehr effektiv gezeigt, um Probleme im Bereich Natural Language Processing (NLP) zu lösen. Das sieht man einfach daran, wie viele Probleme heutzutage nicht mehr mit traditionellen regelbasierten Systemen oder Heuristiken gelöst werden, sondern mit KI. Eine sehr neue Entwicklung ist die Transformerarchitektur (2017) und Large Language Models (LLMs).

```
assert transliterate("وَلِلاستِشراق") == "wa-li-l-istišrāq"
```

Abbildung 7: GitHub Copilot schlägt die richtige Transliteration vor

Mir ist sofort aufgefallen, dass Programme wie ChatGPT und GitHub Copilot das Transliterationsproblem lösen konnten (s. Abb. 7). GitHub Copilot ist ein Programm, dass eigentlich zum Schreiben von Code konzipiert ist und im Code-Editor Vorschläge macht. Unter anderem hat es mir regelmäßig auch Vorschläge zur Transliteration gemacht und hat mir selbst so auch viel Arbeit erspart. Es ist zu erwarten, dass LLMs darin gut sind, denn die Übersetzung ist etwa ein viel schwereres, aber konzeptionell-ähnliches Problem, das diese Modelle schon sehr gut beherrschen.

Es war auch schnell klar, dass mir im Rahmen dieser Bachelorarbeit in sämtlicher Hinsicht die Ressourcen fehlen würden, um eine KI von Grund auf zu entwickeln. Da kommen die sogenannten Foundational Models ins Spiel. Diese liefern eine Grundlage, auf der man dann eine Lösung für ein spezifisches Problem finden kann. Dabei gibt es grundsätzlich zwei mir bekannte Methoden: Low Rank Adaptation (LoRA) und Prompt Engineering.

LoRA

Bei LoRA (2021) wird das Modell selbst abhängig von den Trainingsdaten verändert (adapted). Dabei wird aber nur ein kleiner Teil verändert (Low Rank). Dadurch wird erstens der Trainingsaufwand verringert. Zweitens wird dadurch das Risiko verringert, dass das Modell etwas "vergisst", also Wissen, das in dem Modell gespeichert ist, überschrieben wird. Zuletzt scheint es so, als ob es für die finale Genauigkeit nicht nachträglich ist. [21] QLoRA (Quantized Low Rank Adapter) ist eine brandneue Verbesserung aus 2023, die durch bestimmte Tricks noch performanter trainiert, ohne dabei im gleichen Maße Genauigkeit zu verlieren. [22]

Prompt Engineering

Beim Prompt Engineering versucht der Prompt Engineer, also in dem Fall ich, einem fertigen LLM korrekte Antworten zu entlocken, indem er die richtigen Anweisungen gibt. Er muss also der Maschine so gut wie möglich beschreiben, wie die Transliteration funktioniert. [23] Bei Modellen mit einem großen Kontext kann man beispielsweise auch einfach ganze PDF-Dokumente als Text einspeisen. "Prompt Engineer" ist mittlerweile zu einem Jobtitel geworden und verlangt viel Erfahrung mit verschiedenen KI-Modellen, um zu lernen, wie man die KI dazu bringen kann, bestimmte Dinge zu produzieren. Beispielsweise kann man den Schriftton oder das Ausgabeformat (Tabelle, Liste, etc.) anpassen. Glücklicherweise gibt es auch Hilfestellungen wie zum Beispiel von Microsoft, welche verschiedene Konzepte wie Anweisungen, Hauptinhalte, Beispiele oder besondere Stichworte einfach erklären. [24]

In beiden Fällen, insbesondere aber bei LoRA, sind Trainingsdaten, also Beispiele, von denen der Computer lernen kann (es heißt ja nicht umsonst Machine Learning), essenziell. Da ich keine Datensätze finden konnte, habe ich von Hand einige Beispiele gesammelt. Trotzdem reicht es nicht

einmal ansatzweise. Ich kann vielleicht ein paar Dutzend oder, wenn ich mir wirklich viel Mühe gebe, ein paar Hundert Beispiele sammeln, aber es wären wahrscheinlich eher Tausende nötig.

Der Vorteil von Prompt Engineering ist dahingehend, dass es auch möglich ist, Anweisungen zu geben, und man relativ leicht ausprobieren kann, ohne dass man viel Zeit zum Trainieren verliert. Dieser schnellere Iterationsprozess beschleunigt die Entwicklung erheblich. Man muss jedoch auf die Kontextgröße achten, diese ist jedoch bei neueren Modellen eigentlich immer ausreichend. Das ist nämlich der nächste Vorteil von Prompt Engineering: Man kann viel größere Modelle nehmen.

Ich habe zur Veranschaulichung ChatGPT-4 (hier Link einfügen) benutzt. [48] Ich gehe jedoch davon aus, dass andere Modelle vielleicht in einigen Dingen besser oder schlechter sind, aber grundlegend auch die gleichen Probleme haben, da sie architektonisch die gleiche Grundlage haben: den Generative Pretrained Transformer (GPT). Bei ChatGPT kann ich jedoch nennenswerter Weise einen Link zum Chat teilen, was zu Reproduktions- bzw. Nachvollziehbarkeitszwecken sehr vorteilhaft ist. [4] Mir war zuvor schon aufgefallen, dass es besser ist auf Deutsch zu sprechen, weil die KI, wenn man auf Englisch spricht, trotz der expliziten Aufforderung nach dem Standard der **Deutschen** Morgenländischen Gesellschaft zu transliterieren, einfach die englischen Systeme nimmt.

Zunächst wäre es doch interessant zu wissen, ob ChatGPT die Transliteration der DMG überhaupt kennt. Die Antwort ist Ja, es kennt auch die Entsprechungen der arabischen Buchstaben, die Grundlage der Transliteration. Wir tasten uns langsam voran und fangen mit dem arabischen Modellwort schlechthin an: *al-kitāb*. Das Ergebnis ist erstaunlich: ChatGPT gibt nicht nur die korrekte Transliteration, sondern zeigt auch seine Reasoningfähigkeiten und schlüsselt die Lösung genau auf und gibt noch die Übersetzung “das Buch”. Als Nächstes schauen wir, ob die Assimilation der Sonnenbuchstaben funktioniert. Hier sieht es ähnlich aus.

Es scheint also, dass wir mal was Schwereres probieren können. Was ist mit einem unvokalisierten Text mit wortübergreifenden Assimilationen. Das Ergebnis ist in Pausa und sieht recht gut aus, die Vokalisation hat gut funktioniert. Nur die Flexionsendung des Verbes und auch die Assimilation hat nicht funktioniert:

هذا الكتاب الجديد الطالب. هو يقرأ الكتاب الجديد.

ChatGPT mit Korrektur: hāḏā al-kitāb al-ǧadīd aṭ-ṭālib. huwa yaqraʾu al-kitāb al-ǧadīd.

Nun gibt es zwei weitere Ansätze: Versteht ChatGPT den Unterschied zwischen Pausa, und vollvokalisierter Wiedergeben und kann man die Fehler aufbessern? Die Bitte nach vollvokalisierter Transliteration offenbart die fehlenden Assimilationen noch mehr. Es zeigt jedoch auch nochmal die Exzellenz der Vokalisation: Das Schwierigste sind gerade die Flexionsendungen, denn für die Flexionsendungen ist ein Verständnis der Bedeutung nötig, die ChatGPT auch wieder explizit beweist. Auf Hinweis, assimiliert ChatGPT auch den Artikel und erreicht damit das perfekte Ergebnis.

Im Weiteren probieren wir aus, was ChatGPT auf Anhieb kann und wofür es einen Hinweis braucht. Daraus kann dann ein umfänglicher Prompt zusammengestellt werden. Leider wurde ChatGPT zwischenzeitlich auf GPT-3.5 zurückgesetzt, was die folgenden Ergebnisse verzerrt hat, aber hier sind Anweisungen, die sich anbieten würden:

- “Transliteriere eng nach dem Standard der **Deutschen Morgenländischen Gesellschaft”**
- Eine Tabelle der Entsprechungen für die arabischen Buchstaben

[4] https://chatgpt.com/share/7ac4965d-f56a-41f9-8420-bef773fd504f

- “Fasse dich ganz kurz, gib keine Aufschlüsselung oder Erklärungen und gib nur die Transliteration an”
- Abhängig von der erwünschten Form:
    - “Transliteriere **vollvokalisiert**. Gib alle Endungen mit an**”**
    - “Transliteriere in **Pausa.** Gib die **Endung von Verben** mit an nicht die von **Verbalsubstantiven”**
- “Trenne die Konjunktionen wa und fa, den Futurpartikel sa, die Präpositionen bi, li und ka und den Artikel al mit einem Bindestrich: البَيت al-bayt, بِالْبَيت bi-l-bayt, كاليَوم ka-l-yawm, للامتِحان li-l-imtiḥān, وَلِلاستِشراق wa-li-l-istišrāq”
- “Assimiliere den Artikel mit dem vorhergehenden Wort: al-maktabatu l-kabīratu”
- “Schreibe Namen groß: Ṭāhā, Ibrāhīm”
- Spezifischere Anweisungen (z.B.):
    - “Ersetze ein hamza wasl mit einem u, wenn es auf hum oder antum folgt [mit Beispiel]”
    - Hier wäre mehr Forschung nötig

Bei einem weiteren Versuch, einen noch längeren arabischen Text zu transliterieren, halluziniert die KI wieder und wird vollkommen inkohärent. [5] Man könnte hier bei ausreichender Kontextgröße einfach eine massive Zusammenfassung aller Quellen benutzen. Leider kann man diese meist nicht einfach direkt kopieren, weil sie entweder Scans sind, oder nicht Computer-verständlich formatiert sind. Man könnte aber zum Beispiel den, leider unvollständigen, Wiktionary-Artikel zur DMG-Hilfe als Basis nehmen, der in Unicode verfasst ist. [8]

State of the art

Tatsächlich war ich extrem überrascht von der Leistung von GPT-4, aber sobald ich wieder auf GPT-3.5 abgestuft wurde, passierte wieder, was ich schon oft gesehen hatte: Das Modell fällt immer wieder in alte Fehler zurück und halluziniert und man hat als Prompter das Gefühl, man tritt auf der Stelle. Dies liegt möglicherweise an der geringeren Kontextgröße, wie an dem weiteren Verlauf erkennbar wird. Das Problem könnte also durch einen zusammenhängenden Prompt gelöst werden.

Der Fakt, dass GPT-4 so viel besser ist als GPT-3.5 könnte darauf hindeuten, dass wir gerade an einem Wendepunkt sind, wo die neuesten Modelle (the state of the art) das Problem effektiv lösen können, aber ältere Modelle nicht. Es erwartet uns also eine sehr spannende Zeit.

Bewertung

Die Leistung der KI war sehr beeindruckend. Denn sie kann das, was mich Monate gekostet hat und was ich nicht ohne einige Programmbibliotheken geschafft hätte, einfach auf Anhieb. Gleichzeitig ist sie auch sehr inkonsistent. Aber ich gehe davon aus, dass ein guter Prompt mit einem der neusten Modelle eine gute Grundlage bieten könnte, die auch schon die meisten Probleme von Transliterierenden lösen kann. Die KI hat außerdem die Kapazität sich schnell an neue Anforderungen anzupassen, ist also flexibel.

[5] https://chatgpt.com/share/d8af613b-34ec-48cc-85f7-e42a67ffae50

VI. Der Code

Der Code ist wie erwähnt in Python geschrieben. Die Applikation ist mittels eines Flask-Servers implementiert, der ein Web-Frontend anbietet, welches die Transliteration und Vokalisation über eine Mini-API exponiert.

Das Frontend

Das Frontend besteht funktional aus jeweils einem Ein- und Ausgabefeld, einem Knopf zur automatischen Vokalisation und einer Sidebar zur Tätigung von Einstellungen. Ich habe versucht ein gutes Mittelmaß zwischen Funktion und Form zu finden. Es gibt das eine Extrem, bei dem die Form wichtiger ist als die Funktion, was natürlich Quatsch ist. Gleichzeitig soll es natürlich auch angenehm sein, so ein Tool zu verwenden.

Zunächst ist dafür ein gutes Farbschema wichtig. Ich habe mich für ein Gold-Violett-Komplementärkontrast entschieden und natürlich darf auch der obligatorische dunkle Modus nicht fehlen. Als Frontend-Bibliothek habe ich Bootstrap verwendet. Dann ist für mich sehr wichtig, Feedback zu erhalten, deshalb gibt es gleich zwei Möglichkeiten Feedback zu geben: Eine schnelle mit Daumen hoch und runter, und eine ausführliche, bei der man auch beschreiben kann, was genau einem gefehlt hat. Die Einstellungen können einfach bedient werden, und werden durch Hovern automatisch näher beschrieben und mit Beispielen versehen. Außerdem gibt es eine "ausklappbare" arabische Tastatur und man kann die meisten Buchstaben auch mit einer deutschen Tastatur tippen. Es fehlt jedoch noch eine ausführliche Anleitung. Zuletzt habe ich viel Aufmerksamkeit auf die Ein- und Ausgabe gelegt: Es wird erst transliteriert, wenn die Eingabe für eine kurze Zeit gestoppt wurde. Es war ziemlich schwer, sicherzustellen, dass alle Eingabemethoden regulär mit Strg+Z umkehrbar sind. Die Ausgabe kann durch einen Klick sofort kopiert werden. Fehlermeldungen passieren sauber mithilfe von sogenannten Toasts. Insgesamt ist das Frontend sehr gut abgerundet und läuft ohne Fehler. Das ist wichtig, um den Fokus auf das Backend zu legen.

Eigentlich wollte ich das Frontend noch viel komplexer machen, und zum Beispiel bei "schwierigen" Wörtern mehrere Möglichkeiten anbieten, die man durch Anklicken auswählen kann. Doch aufgrund der Web-REST-Architektur stellte sich das als eine echte technische Schwierigkeit heraus, an die ich mich nicht gewagt habe.

Unicode

Bisher wurden arabische Buchstaben nur in ihrer abstrakten Form erwähnt. Im Computer werden diese abstrakten Buchstaben jedoch sehr real als Zahlen gespeichert. Welche Zahl, welchen Buchstaben oder welches Zeichen repräsentiert, hängt dabei vom jeweiligen Code ab. Ein sehr verbreiteter Code ist der Unicode-Standard. Die arabischen Zeichen haben ihre eigenen Codeblocks. Ein Problem dabei ist, dass viele Zeichen mehrere mögliche Darstellungsweisen haben, es gibt außerdem auch Ligaturen und andere besondere Zeichen, die alle ihre eigene Zahl bekommen. Dadurch ist es zunächst notwendig, den Eingabetext zu normalisieren und zu bereinigen. Dann können die Zeichen in dem daraus resultierenden Text wieder als abstrakte Zeichen behandelt werden. Alle unbekannten Zeichen (also zum Beispiel persische, russische, chinesische oder auch lateinische) können einfach ignoriert werden. Zusätzlich gibt es noch einige Details, wie die Tokenisierung und die Behandlung von Zahlen und Satzzeichen, die nicht relevant genug sind, um genauer darauf einzugehen.

Qualitätssicherung

Zur Qualitätssicherung habe ich einige Codedurchsichten gemacht, insbesondere mithilfe des Python-Debuggers in Visual Studio Code (VSCode). Um weitere Fehler zu finden, habe ich verschiedene Leute gebeten, meine Applikation zu testen und mir von ihren Ergebnissen zu berichten. Das hat schon zur Entdeckung einiger Fehler geführt.

Im Allgemeinen ist die Software gut durchgetestet. Das ist glücklicherweise bei dieser Aufgabe ziemlich einfach, denn ich kann die Beispiele aus den verschiedenen Spezifikationen ziehen und dann überprüfen, ob mein Programm mit den entsprechenden Einstellungen zum gleichen Ergebnis kommt, wie die entsprechende Spezifikation. Oft habe ich diesem Muster folgend erfolgreich Test-Driven-Development (TDD) betrieben, bei der man zuerst fehlgeschlagene (rote) Tests schreibt und sie dann durch eine korrekte Implementierung zu grünen Tests macht. Außerdem helfen mir Tests, Regressionen zu verhindern. Ich kann also mit meinen 200 Zeilen Tests in dem extrem komplexen System, das NLP beansprucht, ruhig an einem Teilproblem arbeiten und weiß, dass ich keinen anderen Teil beeinträchtigt habe, solange meine Tests noch laufen (Man darf sich aber natürlich, insbesondere am Anfang, nicht allzu sehr in Sicherheit schwelgen).

Durch die Tests weiß ich aber auch genau, wo es noch hakt. Nämlich vor allem bei Präfixen und bei gewissen Nomen. Dies liegt jedoch hauptsächlich an der benutzten Bibliothek Qalsadi. [40] Die Tests schlagen fehl, weil die Bibliothek fehlschlägt, also keine morphologischen Ergebnisse liefert.

Bibliotheken zur Transliteration und Vokalisation

Mein Code baut fundamental auf den Vorarbeiten von Taha Zerrouki auf, die zum großen Teil aus seiner Dissertation “Towards An Open Platform For Arabic Language Processing” hervorgehen. [20]

- PyArabic 0.6.15: Bietet grundlegende Funktionen und Konstanten zur Manipulation arabischer Buchstaben und Texte [41]
- tashaphyne 0.3.6: Hauptsächlich Stemming und Segmentierung. Es nutzt dazu einen angepassten Endlichen Automaten [42][43]
- Naftawayh 0.4: Word-Tagging in die Tags Nomen, Verben und Stopwords [44]
- libqutrub 1.2.4.1: Konjugation [45]
- qalsadi 0.5: Schnürt die vorhergehenden Bibliotheken in ein Paket [40]
- asmai 0.1: Semantische Analyse [46]
- mishkal 0.4.1: Vokalisation [20]
- gradio-client 1.0.1: Ein Client zum Empfang der Vokalisation von Huggingface

Ich habe sehr gemischte Gefühle zu den Bibliotheken von Taha Zerrouki. Auf der einen Seite sind sie undokumentiert oder sogar falsch dokumentiert und inkohärent. Der Code ist ein komisches Amalgamat aus Python2 und Python3 und wurde offensichtlich von jemandem geschrieben, der weder das eine noch das andere beherrscht. Auf der anderen Seite wäre ich ohne die Bibliotheken niemals so weit gekommen und deswegen bin ich Dr. Taha Zerrouki zu großem Dank verpflichtet. Und zum Glück konnte ich einige der Probleme lösen, indem ich operationsähnliche Eingriffe in die Bibliotheken unternahm, und den Output nochmal überprüft und nachbearbeitet habe.

Natürlich stellt sich die Frage, nach Alternativen. Zunächst gibt es Farasa, die nicht nur Vokalisation anbieten. Leider gibt es ihre Bibliotheken nur als Javapakete oder per HTTP-API. Beides war aufgrund der technischen Bedingungen nicht einfach realisierbar. Es gab noch einige andere Alternativen, die ich erst zum Ende meiner Arbeiten finden konnte. Die meisten konnten aufgrund von fehlender Funktionalität ausgeschlossen werden. Bei anderen wie Camel Tools musste eine Kosten-Nutzen-Abwägung gemacht werden. [49] Die möglichen Kosten waren die Einbindung der

Bibliothek in die Codebase, der Nutzen war nicht abzusehen. Denn abgesehen von dem jüngeren Alter, der größeren Anzahl an Mitarbeitenden und der übersichtlicheren Dokumentation gab es keine Beweise dafür, dass die Transliteration sich dadurch verbessern würde. Es war auch nicht klar, wie man das hätte messen können. In Anbetracht der fehlenden Zeit, schien es also sinnvoll, den Fokus auf Anderes zu legen.

Struktur der regelbasierten Transliteration

Bei der Transliteration wird zunächst ein Unicode-Cleanup (Unicode) durchgeführt, dann wird tokenisiert und der Text in Sätze aufgeteilt. Dann werden die Namen erkannt und jedes Wort wird grammatisch analysiert (Bibliotheken zur Transliteration). Die grammatische Analyse ergibt Lemma, Wortart, Fall, Bestimmtheit, Präfix, Verbflexion und Suffix. Dann wird der letzte Kurzvokal (Die Flexionsendungen) entfernt, wenn das Wort ein Nomen ist, dass nicht im Akkusativ steht und kein Suffix hat. Dann wird das Präfix überprüft und direkt transliteriert. Dann erst können wir uns besonderen Worten (Transliteration - Weiteres) zuwenden. Dann wird das hamzat al-waṣl behandelt, sowohl für das jetzige Wort als auch für das nächste. Dann wird die Idafah behandelt. Zum Schluss wird der Rest abhängig von den zuvor gesammelten Informationen transliteriert. Das funktioniert, wie ich erklärt habe mit einer einfachen Liste von Ersetzungsregeln, die mit regulären Ausdrücken ausgedrückt werden. Zuletzt wird der erste Buchstabe von Namen großgeschrieben und danach das Präfix angehängt und die Tokens werden wieder an ihren ursprünglichen Platz getan. Sonderzeichen sind nicht Teil von Tokens und werden gesondert behandelt.

Weiteres

Der Code ist bis auf ein paar unregelmäßige Kommentare kaum dokumentiert. Ich persönlich sehe darin erstmal kein Problem, weil ich nicht erwarte, dass sich irgendwer den Code dieser **Anwendungssoftware**, und natürlich vor allem die Dokumentation, anschauen wird und ich werde den Code auch in zehn Jahren noch verstehen können. Bei einer Bibliothek hätte ich eine Dokumentation für notwendig erachtet, da es zu erwarten ist, dass ein Nutzer diese benötigt.

Der Code ist auf GitHub und wird mit Abgabe dieser Bachelorarbeit veröffentlicht:

Repo mit Tag: https://github.com/theRealProHacker/dmg/tree/ba

Commithash: 201c302

VII. Evaluation

Wie schon ganz zu Beginn erwähnt, konnte ich nicht ausreichend Daten finden, um eine quantitative Analyse zu verantworten, wo am Ende ein Fehlerwert oder Ähnliches rauskommt. Ich möchte, aber betonen, dass ich durch die automatisierten Tests schon ein Indiz für die Richtigkeit des Algorithmusses habe. Wenn wir also davon ausgehen dürften, dass die Tests abdeckend und korrekt sind und es kein UI gäbe, dann bräuchten wir auch gar keine weiterführende Evaluation.

Wenn wir als Kosten für den Nutzer definieren, wie viele Zeichen er löschen, ändern oder hinzufügen muss, dann fallen insbesondere die Präfixe schwer ins Gewicht. Gerade bei längeren Präfixen stößt Qalsadi an seine Grenzen und dann muss der Nutzer sehr viel selbst eingreifen. Ein extremes Beispiel wäre *waliālāstišrāq* vs. *wa-li-l-istišrāq* mit Korrekturkosten von 5 für ein Wort. So ein Wort kommt jedoch sehr selten vor. Die aller meisten Wörter werden meiner Einschätzung nach, nur ein bis zwei Präfixe haben, auch die Kombination aus Artikel und *hamzat al-waṣl* ist nicht häufig, da das *hamzat al-waṣl* alleine schon selten ist, es sich dann um eine Verbalsubstantiv handeln muss und dann muss auch noch ein Artikel davorstehen, was im Arabischen schon seltener passiert als im Deutschen, weil Nomen auch auf anderen Wegen bestimmt werden können.

Die Vokalisation ist kein Problem, eher ein Staunfaktor. Sie braucht zwar etwa 10 Sekunden, aber das ist viel weniger als sie einem erspart. Ich musste sie bisher noch kein einziges Mal korrigieren.

Anstatt zu versuchen, eine großangelegte Umfrage unter den relativ unerfahrenen Studenten zu starten, hielt ich es für besser, einige Experten zu fragen. Generell gab es große Begeisterung, und die ersten Pläne wurden gemacht, die App bei den Professoren und Unis im deutschsprachigen Raum bekannt zu machen. Die App wurde als ansprechend, übersichtlich und allgemein nützlich bewertet.

> *“Das Programm scheint perfekt zu sein.” – Doga Akpinar*[6]

6 https://www.geschkult.fu-berlin.de/e/semiarab/arabistik/Seminar/Mitarbeiterinnen-und-Mitarbeiter/Studentische-Hilfskraefte/akpinar.html

VIII. Fazit und Aussicht

Aus der Evaluation und meinen Tests zu Large Language Models (LLMs) können wir schließen, dass sowohl eine Mischung aus einem intelligenten System zur Vokalisation und einem regelbasierten Ansatz zur weiteren Analyse als auch ein vollständig intelligenter Ansatz auf Basis von LLMs Potenzial zeigen. Es ist jedoch noch weitere Arbeit nötig:

> *"Für den Transliterierenden heißt das dasselbe wie für den, der einen arabischen Text laut vorlesen will: Beides ist ohne ein genaues Verständnis des Textes und der Ausspracheregeln unmöglich." — Thomas Hildebrandt* [1]

Trotzdem bietet die Software einen enormen Mehrwert im Vergleich zur Mühsal einer vollständig manuellen Transliteration. Die Korrektur der wenigen möglichen Fehler ist vergleichsweise einfach und bequem. Denn Transliteration wird für die absehbare Zukunft immer noch eine menschliche Aufgabe bleiben.

NLP ist ein interessanter Forschungsbereich, weil es nicht nur theoretisch (die Laufzeiten von Algorithmen vergleichen), aber auch nicht nur praktisch ist (einen Webserver mit einer Datenbank programmieren). Durch die Recherche treten kontinuierlich neue Anforderungen auf, die gelöst werden müssen. Je tiefer man in das Thema eintaucht, desto mehr lernt man die Feinheiten der natürlichen Sprache kennen. Eine Herausforderung war, eine Balance zwischen all den Quellen herzustellen und die vielen verschiedenen Ansichten gebührend zu berücksichtigen. Der letztlich gewählte Ansatz nimmt den Orientalisten jetzt und in näherer Zukunft eine große Last ab.

> *"The thing is, the problem space is almost unbounded. You could keep improving the program to account for more quirks of Arabic orthography, but there would always be the possibility that it would choke on some valid input that you haven't tested before. I think this will turn out to be a labor of love, if you stick with it. For a BA thesis project, you've done a lot already." — **Dr. Theodore S. Beers** eine Woche nach Anmeldung der Bachelorarbeit*

Ein wichtiges Anliegen ist mir, dass die Lösung vielseitig genutzt wird, dazu ist es wichtig an die Professoren und so in die Hörsääle, Seminarräume und Fachbereichsräte zu kommen. Denn die App kann als Werkzeug von Experten dienen, aber könnte laut Angaben einer Lehrerin auch als Lehrmethode angewandt werden.

Glücklicherweise ist der Arabistikbereich der FU gut vernetzt und so hoffe ich, dass Mundpropaganda die App auch in anderen deutschsprachige Universitäten bekannt machen wird. Dazu wäre es auch hilfreich, eine gute Anleitung mit hilfreichen Tipps zu schreiben. Eine weitere Möglichkeit wäre, verschiedene Profile für verschiedene Universitäten und Professoren zu erstellen, die dann in der URL genannt werden können, dadurch könnten die Professoren etwa einen Link zum Programm mit ihren persönlichen Einstellungen teilen.

Es bietet sich außerdem an, sowohl die Transliteration von Namen, als auch eine LLM-basierte Chat-Transliteration bereitzustellen. So kann ein Nutzer zwischen der Konsistenz einer regelbasierten Lösung und der Flexibilität einer LLM-basierten Lösung aussuchen.

Weitere Transliterationssysteme und Sprachen wären auch interessant. Ich wurde mehrmals gefragt, ob es auch möglich wäre, das Programm für die Transliteration des IJMES zu erweitern, weil es mittlerweile von den meisten englisch-sprachigen Wissenschaftlern verwendet wird. Konzeptionell ist es quasi das gleiche System, aber mit leichteren Regeln und ohne 1-zu-1-Transliteration. [47] Auch

das informelle Arabezi (auch Chat-Arabisch) wäre interessant; bei Sprachen arabische Dialekte, Persisch, Türkisch, Kurdisch, Urdu, etc.

Bei der Transliteration sind noch Probleme offen. Beispielsweise der Umgang mit Wörtern, die Qalsadi nicht erkennt. Hier könnten weitere Heuristiken angewandt werden, um dennoch die relevanten Informationen wie Prä- und Suffixe zu extrahieren. Außerdem wäre eine Erweiterung des arabischen Wörterbuches, das Qalsadi nutzt, vorteilhaft. Man könnte auch eine andere Bibliothek nehmen oder eine eigene Lösung zur Analyse von vokalisierten arabischen Texten entwickeln. Das würde in Aussicht stellen, eine aktuelle, vertrauenswürdige und verständliche Alternative zu haben.

Es gibt noch einige Spezialfälle, die so selten auftreten und so einfach zu korrigieren sind, dass sich ihre Lösung fast nicht lohnt. So etwa die klassischen präfigierten Partikel *a*, *ta*, *la,* die Behandlung von indeterminierten Präsenzpartizipien hinten schwacher Wurzeln, die Behandlung spezieller Wörter mit *hamzat al-waṣl* gefolgt von *l*, und die Zusammenschreibung einiger Namen.

Eine besonders schwierige Aufgabe stellt das Sammeln von Test- und Trainingsdaten zur automatischen Transliteration nach dem Standard der DMG dar. Dies liegt daran, dass solche Daten entweder nicht öffentlich zugänglich sind (Bücher und Publikationen) oder weit verstreut sind (Wikipedia). Dabei ist die Uneinheitlichkeit unter den deutschsprachigen Transliterierenden eine weitere große Hürde. Außerdem wird auch oft nur die Transliteration angegeben.

Referenzen

[1] T. Hildebrandt, “Die wissenschaftliche Umschrift der arabischen Sprache | Ein Leitfaden für die orientalistischen Fächer der Universität Bamberg,” Universität Bamberg, online: https://www.uni-bamberg.de/fileadmin/orientalistik/Dateien/Translit.pdf (zuletzt 13.06.24)

[2] H. Zirker, “KORAN-Transliteration.” 2013. online: https://www.uni-goettingen.de/de/document/download/43b887124901209d66d40a83d81bc83a.pdf/Koran_transliter.pdf (zuletzt 18.06.24)

[3] A. A. Ambros, “Probleme der Transliteration des Arabischen im EDV-Zeitalter.” Wiener Zeitschrift für die Kunde des Morgenlandes, vol. 81, pp. 9-19. 1991. online: https://www.jstor.org/stable/23865613

[4] “Buckwalter transliteration.” Wikipedia – The Free Encyclopedia. https://en.wikipedia.org/wiki/Buckwalter_transliteration (zuletzt 12.06.24)

[5] “DIN 316352.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/DIN_31635 (zuletzt 12.06.24)

[7] “Transliteration der arabischen Schrift.” Deutsche Morgenländische Gesellschaft. https://dmg-web.de/page/transliteration_de (zuletzt 12.06.24)

[8] “Hilfe:DMG.” Wiktionary – Das freie Wörterbuch. https://de.wiktionary.org/wiki/Hilfe:DMG (zuletzt 12.06.24)

[9] C. Brockelmann, A. Fischer, W. Heffening, F. Taeschner mit Beiträgen von Ph. S. von Ronkel und O. Spies, “Die Transliteration der arabischen Schrift in ihrer Anwendung auf die Hauptliteratursprachen der islamischen Welt | Denkschrift dem 19. internationalen Orientalistenkongreß in Rom,” Transkriptionskommission der Deutschen Morgenländischen Gesellschaft bei F. A. Brockhaus, Leipzig, 1935, online: https://www.aai.uni-hamburg.de/voror/medien/dmg.pdf

[10] “Regeln zur wissenschaftlichen Umschrift des Arabischen am Seminar für Orientalistik der Christian-Albrechts-Universität zu Kiel,” Islamwissenschaft, Christian-Albrechts-Universität zu Kiel, online: https://www.islam.uni-kiel.de/de/studium/materialien/materialien-1/umschrift-arabisch (zuletzt 13.06.24)

[11] “Transkriptionsregeln für die arabische Schrift,” Orientalisches Institut, MLU Halle, 2012, online: https://www.gkr.uni-leipzig.de/fileadmin/Fakult%C3%A4t_GKR/Orientalisches_Institut/Studium/Werkzeugkasten/Merkblatt_Transkription_plus_Tabelle.pdf (zuletzt 13.06.24)

[12] “Die Umschrift (Transkription) des arabisch-persischen Alphabets,” Seminar für Sprachen und Kulturen des Vorderen Orients, Islamwissenschaft, Ruprecht-Karls-Universität Heidelberg, online: https://www.uni-heidelberg.de/md/ori/islamwissenschaft/transkription.pdf (zuletzt: 13.06.24)

[13] “Arabic to English Transliteration Online.” Arabic Alphabet Academy. https://transliterate.arabicalphabet.net/ (zuletzt 12.06.24)

[14] “Ergänzungen zu den Transliterationsregeln der Deutschen Morgenländischen Gesellschaft für die arabische Schrift,” Deutsche Morgenländische Gesellschaft, online: https://dmg-web.de/page/transliteration_de/ergaenzungen_arabisch-transliteration_de.pdf (zuletzt 14.06.24)

[15] P. Reichmuth und S. Schielke, “Dī visenšaftliḫe 'umšrift des ʿarabišen nāḫ dēn rēgeln der doičen morgenlendišen gezelšaft cuzsamengeštelt fon Filip Rayḫmūṭ 'unt Ṣamūli Šīlke”, online: http://orientasia.de/cgi-data/downloads/arab_trans.pdf (zuletzt 15.06.24)

[16] “Taschdīd.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Taschd%C4%ABd (zuletzt 15.06.24)

[17] “Madda.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Madda (zuletzt 15.06.24)

[18] “ISO 233.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/ISO_233 (zuletzt 15.06.24)

[19] “ALA-LC Transkription.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/ALA-LC-Transkription (zuletzt 12.06.24)

[20] T. Zerrouki, “Towards An Open Platform For Arabic Language Processing,” Ph.D. dissertation, Ecole Nationale Supérieure d'informatique, Alger, Algérie, 2020. doi: 10.13140/RG.2.2.29882.82881

[21] E. J. Hu et al., LoRA: Low-Rank Adaptation of Large Language Models. 2021. doi: https://doi.org/10.48550/arXiv.2106.09685

[22] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, QLoRA: Efficient Finetuning of Quantized LLMs. 2023. https://doi.org/10.48550/arXiv.2305.14314

[23] “Prompt Engineering.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Prompt_Engineering (zuletzt 18.06.24)

[24] “Introduction to prompt engineering.” Microsoft Learn – Azure. https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/prompt-engineering (zuletzt 18.06.24)

[25] “Arabischer Name.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Arabischer_Name (zuletzt 18.06.24)

[26] “Arabische Namen für Jungen & Mädchen.” Baby Vornamen. https://www.baby-vornamen.de/Sprache_und_Herkunft/arabische_Vornamen.php (zuletzt 18.06.24)

[27] “Tā' marbūta.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/T%C4%81'_marb%C5%ABta (zuletzt 18.06.24)

[28] “Sonnenbuchstabe.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Sonnenbuchstabe (zuletzt 19.06.24)

[29] F. Alotaibi and M. Lee, "A Hybrid Approach to Features Representation for Fine-grained Arabic Named Entity Recognition", In Proceedings of COLING 2014, Dublin, Ireland, August 23-29. 2014. online: https://fsalotaibi.kau.edu.sa/Pages-Arabic-NE-Corpora.aspx

[30] A. Barqawi, T. Zerrouki, “Shakkala, Arabic text vocalization,” 2017. online: https://github.com/Barqawiz/Shakkala

[31] “Farasa.” QCRI. https://farasa.qcri.org/ (zuletzt 19.06.24)

[32] A. Fadel, I. Tuffaha, B. Al-Jawarneh, and M. Al-Ayyoub, “Arabic Text Diacritization Using Deep Neural Networks,” CoRR, vol. abs/1905.01965, 2019, [Online]. Available: http://arxiv.org/abs/1905.01965

[33] “Arabisch und Syrisch in Unicode.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Arabisch_und_Syrisch_in_Unicode (zuletzt 20.06.24)

[34] M. Hatmi, “Arabic Named Entity Recognition Model,” 2022, https://huggingface.co/hatmimoha/arabic-ner

[35] “Alif.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Alif (zuletzt 20.06.2024)

[36] “Buchārī.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Al-Buch%C4%81r%C4%AB (zuletzt 20.06.24)

[37] “Ibn Battūta.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Ibn_Batt%C5%ABta (zuletzt 20.06.24)

[38] “Hāddsch.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/H%C4%81ddsch (zuletzt 20.06.24)

[39] “Rumi.” Wikipedia – Die freie Enzyklopädie. https://de.wikipedia.org/wiki/Rumi_(Dichter) (zuletzt 20.06.24)

[40] T. Zerrouki, Qalsadi, Arabic mophological analyzer Library for python, 2012. v0.5, 2023. https://pypi.python.org/pypi/qalsadi/

[41] T. Zerrouki, Pyarabic, An Arabic language library for Python, 2010. v0.6.15, 2022. https://pypi.python.org/pypi/pyarabic/

[42] Zerrouki, T. (2024). Tashaphyne: A python package for arabic light stemming. Journal of Open Source Software, 9(93), 6063. doi: http://doi.org/10.21105/joss.06063

[43] T. Zerrouki, Tashaphyne, Arabic light stemmer, 2010. v0.3.6, 2022. https://pypi.python.org/pypi/Tashaphyne/

[44] T. Zerrouki, Naftawayh, Arabic Word Tagger, 2010. v0.4, 2020. https://pypi.python.org/pypi/naftawayh/

[45] T. Zerrouki, Qutrub, Arabic verb conjugation, 2012. v1.2.4.1, 2020. https://pypi.python.org/pypi/libqutrub/

[46] T. Zerrouki, Asmai, Arabic semantic analysis, 2020. v0.1, 2020. online: https://pypi.org/project/asmai/

[47] “IJMES Transliteration System for Arabic, Persian and Turkish.” Universität Cambridge. online: https://www.cambridge.org/core/services/aop-file-manager/file/57d83390f6ea5a022234b400/TransChart.pdf (zuletzt 25.06.24)

[48] ChatGPT. (GPT-3.5 & GPT-4o). OpenAI. online: https://chat.openai.com/chat (zuletzt 25.06.2024)

[49] O. Obeid et al., “CAMeL Tools: An Open Source Python Toolkit for Arabic Natural Language Processing,” in Proceedings of the 12th Language Resources and Evaluation Conference, May 2020, pp. 7022–7032. online: https://www.aclweb.org/anthology/2020.lrec-1.868