Zwischen Interpretation und Berechnung

Algorithmisch Rekursive Sequenzanalyse als Brücke zwischen qualitativer Hermeneutik und formaler Modellierung

Paul Koop

Juni/Juli 1994 & 2024

Zusammenfassung

Die qualitative Sozialforschung steht gegenwärtig vor einem methodologischen Dilemma: Einerseits versprechen generative KI-Systeme eine bislang unerreichte Skalierung interpretativer Arbeitsschritte, andererseits entziehen sie sich durch ihre stochastische Natur der klassischen Validierungslogik qualitativer Forschung. Der vorliegende Beitrag argumentiert, dass dieses Dilemma durch eine Rückbesinnung auf formalisierende Ansätze aufgelöst werden kann. Als konkreten Lösungsansatz entwickelt der Beitrag die **Algorithmisch Rekursive Sequenzanalyse (ARS)** , ein Verfahren, das Interpretationsprozesse in eine formale Grammatik überführt und damit transparent, reproduzierbar und intersubjektiv prüfbar macht. Die Verbindung zur aktuellen Diskussion um **Explainable AI (XAI)** erweist sich dabei als doppelt fruchtbar: Sie stellt ein begriffliches Instrumentarium bereit, um die Güte qualitativer Interpretationen zu reflektieren, und erinnert daran, dass Erklärbarkeit kein Luxus, sondern eine Notwendigkeit ist – in der Technik wie in der Wissenschaft. Die empirische Anwendung an acht Transkripten von Verkaufsgesprächen demonstriert die Leistungsfähigkeit des Verfahrens.

Inhaltsverzeichnis

1 Einleitung: Das Paradoxon qualitativer Forschung im Zeitalter generativer KI

Die qualitative Sozialforschung steht gegenwärtig vor einem methodologischen Dilemma. Einerseits versprechen generative KI-Systeme eine bislang unerreichte Skalierung interpretativer Arbeitsschritte. Andererseits entziehen sich eben diese Systeme durch ihre stochastische Natur der klassischen Validierungslogik qualitativer Forschung. Wo diese traditionell auf die detaillierte Offenlegung des Codierprozesses und die intersubjektive Nachvollziehbarkeit setzt, tritt nun ein blinder Verlass auf die vermeintliche „Emergenz“ neuronaler Netze.

Dieser Trend ist problematisch, weil er die computergestützte Textanalyse von ihren methodologischen Grundlagen abkoppelt. Zugleich aber verweist er auf ein Defizit, das die qualitative Forschung selbst betrifft: Sie verfügt über kein formalisiertes Vokabular, um ihre Interpretationsprozesse für algorithmische Verfahren anschlussfähig zu machen. Die Folge ist eine Wahl zwischen zwei unbefriedigenden Optionen: entweder Verzicht auf Skalierung oder Preisgabe methodologischer Kontrolle.

Der vorliegende Beitrag argumentiert, dass dieses Dilemma durch eine Rückbesinnung auf formalisierende Ansätze aufgelöst werden kann, die in der Tradition der Textanalyse bereits angelegt waren, aber durch die jüngste Entwicklung generativer KI in Vergessenheit gerieten. Als konkreten Lösungsansatz entwickelt der Beitrag die **Algorithmisch Rekursive Sequenzanalyse (ARS)** , ein Verfahren, das Interpretationsprozesse in eine formale Grammatik überführt und damit transparent, reproduzierbar und intersubjektiv prüfbar macht.

Die Pointe dieses Ansatzes liegt in seiner Verbindung zu aktuellen Diskussionen um **Explainable Artificial Intelligence (XAI)** . XAI hat sich als Antwort auf die Opazität neuronaler Netze entwickelt (Samek & Müller, 2019; Barredo Arrieta et al., 2020). Die zentrale Einsicht lautet: Wer die Entscheidungen komplexer KI-Systeme nicht nachvollziehen kann, kann ihnen nicht vertrauen – und darf sie in sicherheitskritischen Bereichen nicht einsetzen (Weller, 2019). Diese Einsicht, so die These des Beitrags, lässt sich produktiv auf die qualitative Forschung wenden: Auch sie benötigt Verfahren, die ihre Interpretationsprozesse erklärbar machen. Die ARS versteht sich als ein solches Verfahren – als Beitrag zu einer **erklärbaren qualitativen Forschung**, die die methodologischen Standards der Disziplin wahrt und zugleich für algorithmische Modellierung öffnet.

Der Beitrag ist wie folgt aufgebaut: Abschnitt 2 führt in das Konzept der Explainable

AI ein und entwickelt die Analogie zur qualitativen Forschung. Abschnitt 3 stellt die ARS in ihrer methodischen Architektur dar. Abschnitt 4 dokumentiert die empirische Anwendung an acht Transkripten von Verkaufsgesprächen. Abschnitt 5 reflektiert die Ergebnisse im Lichte der XAI-Diskussion. Abschnitt 6 zieht ein Fazit und zeigt Perspektiven auf.

2 Explainable AI: Begriff, Entwicklung und methodologische Relevanz

2.1 Entstehung und Grundgedanken der XAI

Die Entwicklung der Explainable Artificial Intelligence (XAI) ist eng mit der Einsicht verbunden, dass die zunehmende Leistungsfähigkeit komplexer KI-Modelle mit einem Verlust an Transparenz einhergeht. Insbesondere tiefe neuronale Netze, die in zahlreichen Anwendungsdomänen beeindruckende Ergebnisse erzielen, operieren als „Black Boxes“: Ihre inneren Entscheidungsprozesse sind weder für Entwickler noch für Nutzer unmittelbar nachvollziehbar (Samek & Müller, 2019, S. 2).

Diese Opazität wird dann problematisch, wenn KI-Systeme in sicherheitskritischen Bereichen eingesetzt werden – in der medizinischen Diagnostik, der Rechtsprechung, der Finanzwirtschaft oder der autonomen Steuerung (Ortigossa et al., 2024, S. 80800). Fehlentscheidungen können hier gravierende Folgen haben. Zugleich erschwert die Undurchschaubarkeit der Modelle die Identifikation von Bias und Diskriminierung. Ein vielzitierter Fall ist das COMPAS-System zur Rückfallprognose von Straftätern, das afroamerikanische Angeklagte systematisch benachteiligte, ohne dass diese Verzerrung aus der Modellarchitektur erkennbar gewesen wäre (Barredo Arrieta et al., 2020, S. 84).

Die XAI-Forschung reagiert auf dieses Problem, indem sie Methoden entwickelt, um die Entscheidungen komplexer Modelle nachträglich zu erklären oder von vornherein interpretierbare Modelle zu entwerfen (Mersha et al., 2024). Der Begriff „Explainable AI“ selbst geht auf eine Initiative der US-amerikanischen Forschungsagentur DARPA zurück, die ab 2015 gezielt Projekte zur Erklärbarkeit von KI-Systemen förderte (Barredo Arrieta et al., 2020, S. 86). Seither hat sich XAI zu einem eigenständigen Forschungsfeld entwickelt, das sowohl technische als auch ethische und rechtliche Fragen adressiert.

Eine wichtige rechtliche Triebkraft der XAI-Diskussion war die europäische Datenschutz-

Grundverordnung. Insbesondere Erwägungsgrund 71 wird in der Forschung häufig als Grundlage eines „Rechts auf Erklärung“ interpretiert, auch wenn die Verordnung kein explizites, einklagbares Recht auf vollständige algorithmische Offenlegung formuliert (Wachter et al., 2017). Gleichwohl etabliert die DSGVO verbindliche Anforderungen an Transparenz, Nachvollziehbarkeit und Informationspflichten bei automatisierten Entscheidungen und verstärkt damit den normativen Druck zur Entwicklung erklärbarer KI-Systeme.

2.2 Zentrale Begriffe und Taxonomien

Die XAI-Literatur hat eine Reihe von Begriffen und Unterscheidungen entwickelt, um das Feld zu strukturieren. **Erklärbarkeit (Explainability)** bezeichnet allgemein die Eigenschaft eines KI-Systems, seine Entscheidungen in für Menschen verständlicher Weise darlegen zu können (Barredo Arrieta et al., 2020, S. 89). **Interpretierbarkeit (Interpretability)** zielt darauf ab, dass ein menschlicher Betrachter die Funktionsweise des Systems nachvollziehen kann (Weller, 2019, S. 25). **Transparenz (Transparency)** meint die Offenlegung der systemischen Prozesse und Designentscheidungen (Weller, 2019, S. 27).

Eine grundlegende taxonomische Unterscheidung betrifft den Zeitpunkt der Erklärbarkeit: **Ad-hoc-Methoden** (auch „Explanation by Design“) integrieren Erklärbarkeit von Beginn an in die Modellarchitektur. Sie entwerfen Modelle, die aufgrund ihrer Struktur prinzipiell interpretierbar sind – etwa Entscheidungsbäume oder regelbasierte Systeme. **Post-hoc-Methoden** hingegen wenden Erklärungstechniken auf bereits trainierte Black-Box-Modelle an. Sie versuchen, nachträglich zu rekonstruieren, welche Input-Faktoren für eine bestimmte Entscheidung ausschlaggebend waren (Barredo Arrieta et al., 2020, S. 92).

Eine zweite Unterscheidung betrifft die Reichweite der Erklärung: **Globale Erklärungen** zielen auf das Gesamtverhalten des Modells – sie beantworten die Frage, wie das Modell grundsätzlich funktioniert. **Lokale Erklärungen** hingegen beziehen sich auf einzelne Entscheidungen – sie erklären, warum ein bestimmter Input zu einem bestimmten Output geführt hat (Ortigossa et al., 2024, S. 80805).

Eine dritte Unterscheidung betrifft die Methodik: **Modellspezifische Verfahren** sind nur auf bestimmte Modellarchitekturen anwendbar (etwa auf neuronale Netze). **Modellagnostische Verfahren** hingegen können unabhängig von der konkreten Modellarchitektur eingesetzt werden (Mersha et al., 2024, S. 3).

Zu den bekanntesten XAI-Verfahren zählen:

- **LIME (Local Interpretable Model-agnostic Explanations)**: Ein modellagnostisches Verfahren, das lokal einfache, interpretierbare Ersatzmodelle lernt, um die Entscheidungen komplexer Black-Box-Modelle zu erklären (Barredo Arrieta et al., 2020, S. 102).
- **SHAP (SHapley Additive exPlanations)**: Ein auf kooperativer Spieltheorie basierendes Verfahren, das den Beitrag jedes Input-Features zu einer Vorhersage quantifiziert (Barredo Arrieta et al., 2020, S. 104).
- **Salienz-Maps**: Visualisierungen, die für Bildklassifikatoren anzeigen, welche Bildregionen für eine Entscheidung besonders relevant waren (Zhou et al., 2019).
- **Layer-wise Relevance Propagation (LRP)**: Ein Verfahren, das die Vorhersage eines neuronalen Netzes schichtweise rückwärts durch das Netz propagiert und so relevante Input-Regionen identifiziert (Montavon et al., 2019).

2.3 XAI als methodologische Herausforderung

Die XAI-Diskussion beschränkt sich nicht auf technische Verfahren. Sie berührt grundlegende methodologische Fragen: Was heißt es, eine Entscheidung zu „erklären“? Wer ist die Adressatin der Erklärung? Welche Qualitätskriterien gelten für Erklärungen?

Das NIST (National Institute of Standards and Technology) hat hierzu drei fundamentale Eigenschaften guter Erklärungen formuliert (Ortigossa et al., 2024, S. 80810):

1. **Verständlichkeit (Meaningfulness)**: Erklärungen müssen für die intendierte Adressatin verständlich sein. Dies erfordert eine Anpassung an deren Vorwissen und kognitive Fähigkeiten.
2. **Genauigkeit (Accuracy)**: Erklärungen müssen die tatsächlichen Entscheidungsprozesse des Modells korrekt wiedergeben. Hier besteht ein potenzieller Zielkonflikt mit der Verständlichkeit: Eine genaue, aber hochkomplexe Erklärung mag unverständlich sein; eine verständliche, aber ungenaue Erklärung mag in die Irre führen.
3. **Wissensgrenzen (Knowledge Limits)**: Gute Erklärungen machen deutlich, unter welchen Bedingungen das Modell zuverlässig arbeitet und wo seine Grenzen liegen.

Diese Kriterien sind nicht nur für technische Systeme relevant. Sie lassen sich, so die

These dieses Beitrags, auf die qualitative Forschung übertragen. Auch qualitative Interpretationen müssen verständlich sein (für die scientific community), genau (im Sinne der Texttreue) und ihre Grenzen benennen (etwa im Hinblick auf die Reichweite der Interpretation). Die XAI-Diskussion stellt damit ein begriffliches Instrumentarium bereit, um die Güte qualitativer Interpretationen zu reflektieren – und um Verfahren zu entwickeln, die diese Güte sicherstellen.

2.4 Von der XAI zur erklärbaren qualitativen Forschung: Eine Analogie

Die Übertragung der XAI-Perspektive auf die qualitative Forschung beruht auf einer Analogie, die in Tabelle 1 systematisiert ist:

Tabelle 1: Analogie zwischen technischer XAI und qualitativer Forschung

Dimension	**Technische XAI**	**Qualitative Forschung**
Problem	Opake Entscheidungen neuronaler Netze	Opake Interpretationsprozesse
Ursache	Subsymbolische Repräsentationen	Implizites Regelwissen
Folge	Fehlendes Vertrauen, unentdeckter Bias	Fehlende Intersubjektivität
Lösung	Explikation der Entscheidungsgrundlagen	Explikation der Interpretationsregeln
Verfahren	LIME, SHAP, Salienz-Maps	ARS, explizite Kategorienbildung
Kriterien	Verständlichkeit, Genauigkeit, Wissensgrenzen	Nachvollziehbarkeit, Texttreue, Reichweite

Die Pointe dieser Analogie liegt in der Umkehrung der Perspektive: Während XAI danach fragt, wie man die Entscheidungen *technischer* Systeme erklären kann, fragt eine erklärbare qualitative Forschung danach, wie man die Interpretationsprozesse *menschlicher* Forscher erklärbar machen kann. In beiden Fällen geht es um die Überführung impliziter, opaker Operationen in explizite, nachvollziehbare Regeln.

Die Algorithmisch Rekursive Sequenzanalyse, die im Folgenden dargestellt wird, versteht sich als ein Verfahren, das diese Überführung leistet. Sie formalisiert Interpretationsprozesse, ohne sie zu automatisieren. Sie produziert explizite, überprüfbare Modelle, ohne die hermeneutische Offenheit zu eliminieren. Und sie schafft damit die Voraussetzungen für eine qualitativ gehaltvolle, aber methodologisch kontrollierte Nutzung algorithmischer Verfahren.

3 Algorithmisch Rekursive Sequenzanalyse: Methodische Architektur

3.1 Grundoperationen: Von der Transkription zur Terminalzeichenkette

Die ARS operiert auf Transkripten natürlicher Interaktionen. Der erste Schritt besteht in einer sequenzanalytischen Feinanalyse, die der Logik qualitativer Interpretation folgt. Die qualitative Sequenzanalyse, wie sie in der objektiven Hermeneutik (Oevermann et al., 1979) und der Konversationsanalyse (Sacks et al., 1974) entwickelt wurde, zielt darauf ab, die latente Sinnstruktur von Interaktionen durch die systematische Rekonstruktion ihrer sequenziellen Ordnung zu erschließen. Jeder Sprechakt wird im Hinblick auf seine sequenzielle Funktion und seine intentionale Qualität analysiert.

Die Analyse folgt dem Prinzip der **Lesartenproduktion und -falsifikation** (Oevermann et al., 1979, S. 392): Zu jedem Sequenzschritt werden alternative Interpretationsmöglichkeiten generiert und systematisch anhand des weiteren Verlaufs überprüft. Dieses Verfahren der „ kontrollierten Interpretation“ (Flick, 2019, S. 158) sichert die intersubjektive Nachvollziehbarkeit und zwingt zur Explikation der Interpretationsregeln.

Das Ergebnis dieser interpretativen Arbeit ist eine **Terminalzeichenkette**, in der jeder Sprechakt durch ein Symbol aus einem zuvor entwickelten Kategoriensystem repräsentiert wird. Diese Terminalzeichen fungieren als formalisiertes Äquivalent qualitativer Codierungen (Przyborski & Wohlrab-Sahr, 2021, S. 207). Die folgende Tabelle illustriert dies am Beispiel eines Transkripts:

Tabelle 2: Beispiel für die Zuordnung von Terminalzeichen

Transkriptausschnitt	**Terminalzeichen**	**Interpretation**
Kunde: Guten Tag	KBG	Kunden-Gruß (Initiation der Interaktion)
Verkäuferin: Guten Tag	VBG	Verkäufer-Gruß (reziproke Bestätigung)
Kunde: Einmal von der groben Leberwurst, bitte.	KBBd	Kunden-Bedarf (Artikulation eines Kaufwunsches)

3.2 Grammatikinduktion: Von Einzelfällen zu generativen Modellen

Auf der Grundlage der Terminalzeichenketten wird für jedes Transkript eine individuelle Grammatik induziert. Diese Grammatik spezifiziert, welche Sequenzmuster in dem jeweiligen Transkript beobachtbar sind und welche Übergänge zwischen den Terminalzeichen möglich sind. Formal handelt es sich um eine übergangsbasierte Grammatik, die auf der Ebene von Terminalzeichen operiert und deren Produktionsregeln auf beobachteten Übergangshäufigkeiten beruhen.

Im Unterschied zu klassischen linguistischen PCFGs (Manning & Schütze, 1999) verzichtet die ARS auf explizite Nichtterminale und tiefenrekursive Ableitungen. Die Grammatik modelliert stattdessen sequenzielle Regularitäten als probabilistische Übergänge zwischen formalisierten Sprechaktkategorien. Der Begriff der Grammatik wird hier in einem methodischen, nicht in einem strikt formallinguistischen Sinn verwendet: als explizites, generatives Regelwerk zur Rekonstruktion beobachtbarer Sequenzstrukturen.

Die Induktion erfolgt durch einfache Zählung der beobachteten Übergänge:

```
transitions = {}
for chain in empirical_chains:
    for i in range(len(chain) - 1):
        start, end = chain[i], chain[i + 1]
        if start not in transitions:
            transitions[start] = {}
        if end not in transitions[start]:
            transitions[start][end] = 0
        transitions[start][end] += 1
```

Listing 1: Zählung der Übergänge zwischen Terminalzeichen

3.3 Vereinigung und Optimierung

Die individuellen Grammatiken werden zu einer **vereinigten Grammatik** zusammengeführt, die die Sequenzstruktur aller Transkripte abdeckt. Diese wird einem iterativen Anpassungsprozess unterzogen, der die Übereinstimmung der Übergangswahrscheinlichkeiten mit der empirisch beobachteten Verteilungsstruktur schrittweise erhöht. Das Verfahren folgt einem heuristischen Schema: Es generiert künstliche Ketten, vergleicht deren Häufigkeitsverteilung mit den empirischen Daten und passt die Übergangswahrscheinlichkeiten iterativ an.

Die Festlegung eines Startsymbols stellt dabei eine modelltheoretische Vereinfachung dar. Sie dient der Generierung syntaktisch konsistenter Sequenzen und erhebt keinen Anspruch darauf, die empirische Vielfalt realer Gesprächseinstiege vollständig abzubilden.

4 Empirische Anwendung: Acht Transkripte von Verkaufsgesprächen

4.1 Hypothetische Ausgangsgrammatik

Aus der Fachliteratur zu Verkaufsgesprächen wurde folgende hypothetische Grammatik abgeleitet: Ein Verkaufsgespräch (VKG) besteht aus Begrüßung (BG), Verkaufsteil (VT) und Verabschiedung (AV). Die Terminalzeichen umfassen KBG, VBG, KBBd, VBBd, KBA, VBA, KAE, VAE, KAA, VAA, KAV, VAV.

4.2 Die acht Transkripte

Die vollständigen Transkripte finden sich in Anhang A. Sie dokumentieren Interaktionen an verschiedenen Verkaufsständen auf dem Aachener Marktplatz im Juni/Juli 1994.

4.3 Terminalzeichenketten

Da Verkaufsgespräche empirisch mit unterschiedlichen Sprechakten beginnen können, wurde für die Generierung künstlicher Sequenzen ein einheitliches Startsymbol definiert. Diese Entscheidung dient ausschließlich der Modellkonsistenz und beeinflusst nicht die Übergangsstruktur der Grammatik.

Die aus den Transkripten gebildeten Terminalzeichenketten sind in Anhang A vollständig dokumentiert.

4.4 Python-Implementierung

Das vollständige Python-Programm zur Grammatikinduktion und -optimierung findet sich in Anhang B. Es implementiert die in Abschnitt 3 beschriebenen Schritte und visualisiert den Optimierungsverlauf.

4.5 Ergebnisse der iterativen Anpassung

Die optimierte Grammatik weist folgende Struktur auf:

Tabelle 3: Optimierte Übergangswahrscheinlichkeiten

Ausgangssymbol	**Folgesymbole mit Wahrscheinlichkeiten**
KBG	VBG (0.67), VBBd (0.33)
VBG	KBBd (1.0)
KBBd	VBBd (0.67), VAA (0.17), VBA (0.17)
VBBd	KBA (0.44), VAA (0.22), KBBd (0.22), KAA (0.11)
KBA	VBA (0.5), VAA (0.5)
VBA	KBBd (0.5), KAE (0.25), VAA (0.25)
VAA	KAA (0.86), KAV (0.14)
KAA	VAV (0.75), VBG (0.25)
VAV	KAV (1.0)
KAE	VAE (1.0)
VAE	KAA (1.0)
KAV	KBBd (1.0)

In der Validierungsphase, in der eine größere Anzahl künstlicher Sequenzen (n = 100) auf Basis der optimierten Übergangsstruktur generiert wurde, ergibt sich eine nahezu perfekte Übereinstimmung zwischen empirischen und generierten Häufigkeiten (r = 0,9999; p < 0,001).

Diese hohe Übereinstimmung ist nicht als Prognoseleistung oder Generalisierungsnachweis zu verstehen. Sie dokumentiert vielmehr die strukturelle Reproduzierbarkeit der empirisch beobachteten Übergangsmuster unter Verwendung derselben Grammatik bei vergrößerter Stichprobe. Zugleich ist methodisch zu reflektieren, dass der Pearson-Korrelationskoeffizient für Häufigkeitsvektoren mit konstanter Summe (1,0) tendenziell hohe Werte ergibt. Die hier beobachtete Korrelation bestätigt daher primär die interne Konsistenz des Verfahrens, weniger eine externe Validität im Sinne von Vorhersagekraft (Flick, 2019, S. 489).

Während der iterativen Optimierungsphase liegt die Korrelation stabil bei etwa r 0,92, was bereits auf eine hohe strukturelle Passung der induzierten Grammatik hinweist. Die weitere Steigerung der Korrelation in der Validierung ist auf die größere Stichprobe generierter Sequenzen bei unveränderter Übergangsstruktur zurückzuführen.

5 Diskussion: ARS als Beitrag zu einer erklärbaren qualitativen Forschung

5.1 ARS und die XAI-Kriterien

Die ARS erfüllt die drei vom NIST formulierten Kriterien guter Erklärungen in einer für die qualitative Forschung adaptierten Form:

Verständlichkeit wird durch die explizite Kategorienbildung gesichert. Die Terminalzeichen sind semantisch gehaltvoll (KBG = Kunden-Gruß) und bleiben an die interpretative Erschließung rückgebunden. Ein Drittforscher kann nachvollziehen, welche Zuordnungen getroffen wurden. Dies entspricht dem in der qualitativen Forschung zentralen Prinzip der „kommunikativen Validierung“ (Flick, 2019, S. 328).

Genauigkeit wird hier im Sinne struktureller Passung operationalisiert, nicht im Sinne prädiktiver Validität. Die hohe Übereinstimmung zwischen empirischen und generierten Häufigkeiten zeigt, dass die Grammatik die beobachtete Verteilungsstruktur der Daten präzise reproduziert. In der Terminologie der qualitativen Forschung ließe sich von „Gegenstandsangemessenheit“ sprechen (Przyborski & Wohlrab-Sahr, 2021, S. 34).

Wissensgrenzen werden durch die Dokumentation der Lesartenproduktion und -falsifikation markiert. Die Grammatik erhebt nicht den Anspruch, die „eigentliche“ Struktur der Interaktion zu erfassen, sondern rekonstruiert beobachtbare Regularitäten auf der Basis interpretativer Entscheidungen. Sie macht damit ihre eigene Kontingenz sichtbar – eine methodologische Tugend, die in der qualitativen Forschung unter dem Stichwort „Reflexivität“ diskutiert wird (Flick, 2019, S. 129).

5.2 Ad-hoc vs. Post-hoc: ARS als Explanation by Design

In der XAI-Terminologie ist die ARS als **Ad-hoc-Verfahren** (Explanation by Design) zu klassifizieren. Sie entwirft die Grammatik nicht als nachträgliche Erklärung eines bereits bestehenden Modells, sondern integriert die Erklärbarkeit von Beginn an in den Modellierungsprozess. Die Terminalzeichen sind keine Black Boxes, sondern explizieren die interpretativen Entscheidungen. Die Übergangswahrscheinlichkeiten sind keine undurchschaubaren Gewichte, sondern einfache relative Häufigkeiten.

Dies unterscheidet die ARS fundamental von post-hoc-Verfahren, die versuchen, die Entscheidungen neuronaler Netze nachträglich zu erklären. Während diese Verfahren immer nur approximative Einblicke in eine prinzipiell opake Architektur geben

können, ist die ARS von Grund auf transparent angelegt.

5.3 Grenzen der Analogie

Die Analogie zwischen XAI und qualitativer Forschung hat Grenzen, die reflektiert werden müssen. **Erstens** zielt XAI primär auf die Erklärung *technischer* Systeme, während es in der qualitativen Forschung um die Explikation *menschlicher* Interpretationsprozesse geht. Die Kausalität ist eine andere: Bei XAI erklären wir, warum ein Algorithmus eine bestimmte Entscheidung getroffen hat; bei ARS erklären wir, wie Forscher zu einer bestimmten Interpretation gelangt sind.

Zweitens operiert XAI mit einem anderen Wahrheitsbegriff. Die Erklärungen sollen die tatsächlichen Entscheidungsprozesse des Modells korrekt wiedergeben. Bei ARS hingegen gibt es keine „tatsächlichen" Prozesse, die unabhängig von der Interpretation existieren. Die Grammatik ist keine Entdeckung, sondern eine Konstruktion – eine, die sich allerdings an der empirischen Evidenz bewähren muss (Flick, 2019, S. 80).

Drittens ist die Adressatin eine andere. XAI-Erklärungen richten sich an Nutzer, Entwickler oder Regulierungsbehörden. ARS-Erklärungen richten sich an die scientific community der qualitativen Forschung. Die Kriterien der Verständlichkeit müssen daher an deren spezifische Diskurspraxis angepasst werden.

5.4 Methodologische Implikationen

Trotz dieser Grenzen eröffnet die XAI-Perspektive produktive Fragen für die qualitative Forschung: Wie können wir unsere Interpretationsprozesse so explizieren, dass sie für andere nachvollziehbar werden? Welche Formate der Explikation sind geeignet? Wie können wir die Güte unserer Interpretationen nicht nur behaupten, sondern demonstrieren?

Die ARS gibt auf diese Fragen eine konkrete Antwort. Sie formalisiert Interpretationsprozesse, ohne sie zu automatisieren. Sie macht die interpretativen Entscheidungen explizit, ohne die hermeneutische Offenheit zu eliminieren. Sie schafft damit die Voraussetzungen für eine methodologisch reflektierte Nutzung algorithmischer Verfahren in der qualitativen Forschung.

6 Fazit und Ausblick

Die qualitative Sozialforschung steht vor der Herausforderung, die Möglichkeiten algorithmischer Textanalyse zu nutzen, ohne ihre methodologischen Standards preis-

zugeben. Die Algorithmisch Rekursive Sequenzanalyse bietet einen Weg, diese Herausforderung produktiv zu wenden. Sie formalisiert Interpretationsprozesse, ohne sie zu automatisieren. Sie produziert explizite, überprüfbare Modelle, ohne die hermeneutische Offenheit zu eliminieren.

Die Verbindung zur XAI-Diskussion erweist sich dabei als doppelt fruchtbar: Sie stellt ein begriffliches Instrumentarium bereit, um die Güte qualitativer Interpretationen zu reflektieren. Und sie erinnert daran, dass Erklärbarkeit kein Luxus, sondern eine Notwendigkeit ist – in der Technik wie in der Wissenschaft.

Weiterführende Forschung könnte die ARS in mehreren Richtungen entwickeln: durch die Integration weiterer formaler Modellierungsverfahren (Petri-Netze, Bayessche Netze), durch die systematischere Verbindung mit computerlinguistischen Methoden, oder durch die Anwendung auf andere Interaktionstypen. Entscheidend bleibt dabei stets die methodologische Kontrolle: Die formalen Verfahren müssen den interpretativen Charakter der Analyse respektieren und dürfen nicht zu dessen Automatisierung führen.

Literatur

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82-115.

Flick, U. (2019). *Qualitative Sozialforschung: Eine Einführung* (9. Aufl.). Rowohlt.

Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Mersha, M., et al. (2024). Explainable Artificial Intelligence: A Survey of Needs, Techniques, Applications, and Future Direction. *Neurocomputing*, 599, 128111.

Montavon, G., Binder, A., Lapuschkin, S., Samek, W., & Müller, K.-R. (2019). Layer-Wise Relevance Propagation: An Overview. In W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, & K.-R. Müller (Hrsg.), *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (S. 193-210). Springer.

Oevermann, U., Allert, T., Konau, E., & Krambeck, J. (1979). Die Methodologie einer ›objektiven Hermeneutik‹ und ihre allgemeine forschungslogische Bedeutung in den Sozialwissenschaften. In H.-G. Soeffner (Hrsg.), *Interpretative Verfahren in den Sozial- und Textwissenschaften* (S. 352-434). Metzler.

Ortigossa, E. S., Gonçalves, T., & Nonato, L. G. (2024). EXplainable Artificial Intelligence (XAI)—From Theory to Methods and Applications. *IEEE Access*, 12, 80799-80846.

Przyborski, A., & Wohlrab-Sahr, M. (2021). *Qualitative Sozialforschung: Ein Arbeitsbuch* (5. Aufl.). De Gruyter Oldenbourg.

Sacks, H., Schegloff, E. A., & Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4), 696-735.

Samek, W., & Müller, K.-R. (2019). Towards Explainable Artificial Intelligence. In W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, & K.-R. Müller (Hrsg.), *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (S. 1-10). Springer.

Wachter, S., Mittelstadt, B., & Floridi, L. (2017). Why a right to explanation of

automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2), 76-99.

Weller, A. (2019). Transparency: Motivations and Challenges. In W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, & K.-R. Müller (Hrsg.), *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (S. 23-40). Springer.

Zhou, B., Bau, D., Oliva, A., & Torralba, A. (2019). Comparing the Interpretability of Deep Networks via Network Dissection. In W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, & K.-R. Müller (Hrsg.), *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (S. 239-252). Springer.

A Die acht Transkripte mit Terminalzeichen

A.1 Transkript 1 - Metzgerei

Datum: 28. Juni 1994, **Ort:** Metzgerei, Aachen, 11:00 Uhr

Tabelle 4: Transkript 1 - Terminalzeichen

Transkriptausschnitt	**Terminalzeichen**
Kunde: Guten Tag	KBG
Verkäuferin: Guten Tag	VBG
Kunde: Einmal von der groben Leberwurst, bitte.	KBBd
Verkäuferin: Wie viel darf's denn sein?	VBBd
Kunde: Zwei hundert Gramm.	KBA
Verkäuferin: Sonst noch etwas?	VBA
Kunde: Ja, dann noch ein Stück von dem Schwarzwälder Schinken.	KBBd
Verkäuferin: Wie groß soll das Stück sein?	VBBd
Kunde: So um die dreihundert Gramm.	KBA
Verkäuferin: Das macht dann acht Mark zwanzig.	VAA
Kunde: Bitte.	KAA
Verkäuferin: Danke und einen schönen Tag noch!	VAV
Kunde: Danke, ebenfalls!	KAV

Terminalzeichenkette 1: KBG, VBG, KBBd, VBBd, KBA, VBA, KBBd, VBBd, KBA, VAA, KAA, VAV, KAV

A.2 Transkript 2 - Marktplatz (Kirschen)

Datum: 28. Juni 1994, **Ort:** Marktplatz, Aachen

Tabelle 5: Transkript 2 - Terminalzeichen

Transkriptausschnitt	**Terminalzeichen**
Verkäufer: Kirschen kann jeder probieren hier!	VBG
Kunde 1: Ein halbes Kilo Kirschen, bitte.	KBBd
Verkäufer: Ein halbes Kilo? Oder ein Kilo?	VBBd
Verkäufer: Drei Mark, bitte.	VAA
Kunde 1: Danke schön!	KAA
Verkäufer: Kirschen kann jeder probieren hier!	VBG
Kunde 2: Ein halbes Kilo, bitte.	KBBd
Verkäufer: Drei Mark, bitte.	VAA
Kunde 2: Danke schön!	KAA

Terminalzeichenkette 2: VBG, KBBd, VBBd, VAA, KAA, VBG, KBBd, VAA, KAA

A.3 Transkript 3 - Fischstand

Datum: 28. Juni 1994, **Ort:** Fischstand, Marktplatz, Aachen

Tabelle 6: Transkript 3 - Terminalzeichen

Transkriptausschnitt	**Terminalzeichen**
Kunde: Ein Pfund Seelachs, bitte.	KBBd
Verkäufer: Seelachs, alles klar.	VBBd
Verkäufer: Vier Mark neunzehn, bitte.	VAA
Kunde: Danke schön!	KAA

Terminalzeichenkette 3: KBBd, VBBd, VAA, KAA

A.4 Transkript 4 - Gemüsestand (ausführlich)

Datum: 28. Juni 1994, **Ort:** Gemüsestand, Aachen, Marktplatz, 11:00 Uhr

Tabelle 7: Transkript 4 - Terminalzeichen

Transkriptausschnitt	**Terminalzeichen**
Kunde: Hören Sie, ich nehme ein paar Champignons mit.	KBBd
Verkäufer: Braune oder helle?	VBBd
Kunde: Nehmen wir die hellen.	KBA
Verkäufer: Die sind beide frisch, keine Sorge.	VBA
Kunde: Wie ist es mit Pfifferlingen?	KBBd
Verkäufer: Ah, die sind super!	VBA
Kunde: Kann ich die in Reissalat tun?	KAE
Verkäufer: Eher kurz anbraten in der Pfanne.	VAE
Kunde: Okay, mache ich.	KAA
Verkäufer: Schönen Tag noch!	VAV
Kunde: Gleichfalls!	KAV

Terminalzeichenkette 4: KBBd, VBBd, KBA, VBA, KBBd, VBA, KAE, VAE, KAA, VAV, KAV

A.5 Transkript 5 - Gemüsestand (mit KAV zu Beginn)

Datum: 26. Juni 1994, **Ort:** Gemüsestand, Aachen, Marktplatz, 11:00 Uhr

Tabelle 8: Transkript 5 - Terminalzeichen

Transkriptausschnitt	**Terminalzeichen**
Kunde 1: Auf Wiedersehen!	KAV
Kunde 2: Ich hätte gern ein Kilo von den Granny Smith Äpfeln hier.	KBBd
Verkäufer: Sonst noch etwas?	VBBd
Kunde 2: Ja, noch ein Kilo Zwiebeln.	KBBd
Verkäufer: Sechs Mark fünfundzwanzig, bitte.	VAA
Kunde 2: Auf Wiedersehen!	KAV

Terminalzeichenkette 5: KAV, KBBd, VBBd, KBBd, VAA, KAV

A.6 Transkript 6 - Käseverkaufsstand

Datum: 28. Juni 1994, **Ort:** Käseverkaufsstand, Aachen, Marktplatz

Tabelle 9: Transkript 6 - Terminalzeichen

Transkriptausschnitt	**Terminalzeichen**
Kunde 1: Guten Morgen!	KBG
Verkäufer: Guten Morgen!	VBG
Kunde 1: Ich hätte gerne fünfhundert Gramm holländischen Gouda.	KBBd
Verkäufer: Am Stück?	VBBd
Kunde 1: Ja, am Stück, bitte.	KAA

Terminalzeichenkette 6: KBG, VBG, KBBd, VBBd, KAA

A.7 Transkript 7 - Bonbonstand

Datum: 28. Juni 1994, **Ort:** Bonbonstand, Aachen, Marktplatz, 11:30 Uhr

Tabelle 10: Transkript 7 - Terminalzeichen

Transkriptausschnitt	**Terminalzeichen**
Kunde: Von den gemischten hätte ich gerne hundert Gramm.	KBBd
Verkäufer: Für zu Hause oder zum Mitnehmen?	VBBd
Kunde: Zum Mitnehmen, bitte.	KBA
Verkäufer: Fünfzig Pfennig, bitte.	VAA
Kunde: Danke!	KAA

Terminalzeichenkette 7: KBBd, VBBd, KBA, VAA, KAA

A.8 Transkript 8 - Bäckerei

Datum: 9. Juli 1994, **Ort:** Bäckerei, Aachen, 12:00 Uhr

Tabelle 11: Transkript 8 - Terminalzeichen

Transkriptausschnitt	**Terminalzeichen**
Kunde: Guten Tag!	KBG
Verkäuferin: Einmal unser bester Kaffee, frisch gemahlen, bitte.	VBBd
Kunde: Ja, noch zwei Stück Obstsalat und ein Schälchen Sahne.	KBBd
Verkäuferin: In Ordnung!	VBA
Verkäuferin: Das macht vierzehn Mark und neunzehn Pfennig, bitte.	VAA
Kunde: Ich zahle in Kleingeld.	KAA
Verkäuferin: Vielen Dank, schönen Sonntag noch!	VAV
Kunde: Danke, Ihnen auch!	KAV

Terminalzeichenkette 8: KBG, VBBd, KBBd, VBA, VAA, KAA, VAV, KAV

B Vollständige Python-Implementierung

```
"""
Algorithmisch Rekursive Sequenzanalyse 2.0
Grammatikinduktion aus acht Transkripten
Optimierung durch iterativen Vergleich empirischer und
    generierter Ketten
"""

import numpy as np
from scipy.stats import pearsonr
import matplotlib.pyplot as plt
from tabulate import tabulate

#
    ================================================================================

# 1. EMPIRISCHE DATEN: Terminalzeichenketten aus acht
    Transkripten
#
    ================================================================================


empirical_chains = [
    # Transkript 1: Metzgerei
    ['KBG', 'VBG', 'KBBd', 'VBBd', 'KBA', 'VBA', 'KBBd', '
        VBBd', 'KBA', 'VAA', 'KAA', 'VAV', 'KAV'],
    # Transkript 2: Marktplatz (Kirschen)
    ['VBG', 'KBBd', 'VBBd', 'VAA', 'KAA', 'VBG', 'KBBd', 'VAA
        ', 'KAA'],
    # Transkript 3: Fischstand
    ['KBBd', 'VBBd', 'VAA', 'KAA'],
    # Transkript 4: Gem sestand (ausfuehrlich)
    ['KBBd', 'VBBd', 'KBA', 'VBA', 'KBBd', 'VBA', 'KAE', 'VAE
        ', 'KAA', 'VAV', 'KAV'],
    # Transkript 5: Gem sestand (mit KAV zu Beginn)
    ['KAV', 'KBBd', 'VBBd', 'KBBd', 'VAA', 'KAV'],
    # Transkript 6: K severkaufsstand
    ['KBG', 'VBG', 'KBBd', 'VBBd', 'KAA'],
    # Transkript 7: Bonbonstand
```

```
    ['KBBd', 'VBBd', 'KBA', 'VAA', 'KAA'],
    # Transkript 8: Baeckerei
    ['KBG', 'VBBd', 'KBBd', 'VBA', 'VAA', 'KAA', 'VAV', 'KAV']
]

# ============================================================================
# 2. UeBERGANGSZAEHLUNG UND INITIALE WAHRSCHEINLICHKEITEN
# ============================================================================

def count_transitions(chains):
    """Zaehlt Uebergaenge zwischen Terminalzeichen in allen Ketten"""
    transitions = {}
    for chain in chains:
        for i in range(len(chain) - 1):
            start, end = chain[i], chain[i + 1]
            if start not in transitions:
                transitions[start] = {}
            if end not in transitions[start]:
                transitions[start][end] = 0
            transitions[start][end] += 1
    return transitions

def calculate_probabilities(transitions):
    """Normalisiert Uebergangszaehlungen zu Wahrscheinlichkeiten"""
    probabilities = {}
    for start in transitions:
        total = sum(transitions[start].values())
        probabilities[start] = {end: count / total
                                for end, count in transitions[start].items()}
    return probabilities

# Initiale Berechnungen
```

```
initial_transitions = count_transitions(empirical_chains)
initial_probabilities = calculate_probabilities(initial_transitions)

print("=" * 70)
print("ALGORITHMISCH REKURSIVE SEQUENZANALYSE 2.0")
print("=" * 70)
print("\n1. INITIALE UeBERGANGSWAHRSCHEINLICHKEITEN (AUS EMPIRISCHEN DATEN)")
print("-" * 70)

for start in sorted(initial_probabilities.keys()):
    transitions_str = ", ".join([f"{end}: {prob:.3f}"
                                 for end, prob in
                                    initial_probabilities[
                                    start].items()])
    print(f"{start} -> {transitions_str}")

#
    ================================================================================

# 3. TERMINALZEICHEN UND STARTZEICHEN
#
    ================================================================================


terminal_symbols = sorted(list(set([item for sublist in empirical_chains
                                    for item in sublist])))
start_symbol = empirical_chains[0][0]  # KBG als Start (kann angepasst werden)

print(f"\nTerminalzeichen ({len(terminal_symbols)}): {terminal_symbols}")
print(f"Startzeichen: {start_symbol}")

#
    ================================================================================

# 4. GENERIERUNG KUeNSTLICHER KETTEN
```

```
89 #
       ==========================================================================
90
91 def generate_chain(probabilities, start_symbol, max_length
       =20):
92     """Generiert eine Kette basierend auf den
           Uebergangswahrscheinlichkeiten"""
93     chain = [start_symbol]
94     current = start_symbol
95
96     for _ in range(max_length - 1):
97         if current not in probabilities:
98             break
99
100        next_symbols = list(probabilities[current].keys())
101        probs = list(probabilities[current].values())
102
103        # Falls keine Folgesymbole vorhanden, abbrechen
104        if not next_symbols:
105            break
106
107        next_symbol = np.random.choice(next_symbols, p=probs)
108        chain.append(next_symbol)
109        current = next_symbol
110
111        # Stopp, wenn wir bei einem Terminal ohne weitere
               Uebergaenge landen
112        if current not in probabilities:
113            break
114
115    return chain
116
117 def generate_multiple_chains(probabilities, start_symbol,
       n_chains=8, max_length=20):
118    """Generiert mehrere Ketten"""
119    return [generate_chain(probabilities, start_symbol,
           max_length)
120            for _ in range(n_chains)]
121
```

```
122 #
    ================================================================================

123 # 5. HAEUFIGKEITSANALYSE
124 #
    ================================================================================


125
126 def compute_frequencies(chains, terminals):
127     """Berechnet relative Haeufigkeiten der Terminalzeichen
        in Ketten"""
128     frequency_array = np.zeros(len(terminals))
129     terminal_index = {term: i for i, term in enumerate(
        terminals)}
130
131     for chain in chains:
132         for symbol in chain:
133             if symbol in terminal_index:
134                 frequency_array[terminal_index[symbol]] += 1
135
136     total = frequency_array.sum()
137     if total > 0:
138         frequency_array /= total  # Normierung
139
140     return frequency_array
141
142 # Empirische Haeufigkeiten als Referenz
143 empirical_frequencies = compute_frequencies(empirical_chains,
    terminal_symbols)
144
145 print("\n2. EMPIRISCHE RELATIVE HAEUFIGKEITEN")
146 print("-" * 70)
147 for i, symbol in enumerate(terminal_symbols):
148     print(f"{symbol}: {empirical_frequencies[i]:.4f}")
149
150 #
    ================================================================================

151 # 6. ITERATIVE OPTIMIERUNG DER GRAMMATIK
```

```
152 #
    ================================================================================
153
154 def optimize_grammar(empirical_chains, terminal_symbols,
    start_symbol,
155                      max_iterations=1000, tolerance=0.01,
                            target_correlation=0.9):
156     """
157     Optimiert die Grammatik durch iterativen Vergleich mit
            generierten Ketten.
158     """
159
160     # Initiale Wahrscheinlichkeiten aus empirischen Daten
161     transitions = count_transitions(empirical_chains)
162     probabilities = calculate_probabilities(transitions)
163
164     # Empirische Haeufigkeiten als Zielgroesse
165     empirical_freqs = compute_frequencies(empirical_chains,
            terminal_symbols)
166
167     best_correlation = 0
168     best_significance = 1
169     best_probabilities = None
170     history = []
171
172     print("\n3. ITERATIVE OPTIMIERUNG")
173     print("-" * 70)
174
175     for iteration in range(max_iterations):
176         # Generiere 8 kuenstliche Ketten
177         generated_chains = generate_multiple_chains(
                probabilities, start_symbol, n_chains=8)
178
179         # Berechne Haeufigkeiten der generierten Ketten
180         generated_freqs = compute_frequencies(
                generated_chains, terminal_symbols)
181
182         # Korrelationsanalyse
```

```
183         correlation, p_value = pearsonr(empirical_freqs,
                generated_freqs)
184         history.append((iteration, correlation, p_value))
185 
186         # Fortschrittsanzeige alle 50 Iterationen
187         if iteration % 50 == 0:
188             print(f"Iteration {iteration:4d}: Korrelation = {
                    correlation:.4f}, p = {p_value:.4f}")
189 
190         # Pruefe Abbruchkriterium
191         if correlation >= target_correlation and p_value <
                0.05:
192             best_correlation = correlation
193             best_significance = p_value
194             best_probabilities = {start: probs.copy()
195                                   for start, probs in
                                          probabilities.items()}
196             print(f"\nOptimum erreicht bei Iteration {
                    iteration}:")
197             print(f"  Korrelation = {correlation:.4f}")
198             print(f"  Signifikanz = {p_value:.4f}")
199             break
200 
201         # Anpassung der Wahrscheinlichkeiten
202         for start in probabilities:
203             for end in probabilities[start]:
204                 # Fehlerberechnung
205                 empirical_prob = empirical_freqs[
                        terminal_symbols.index(end)]
206                 generated_prob = generated_freqs[
                        terminal_symbols.index(end)]
207                 error = empirical_prob - generated_prob
208 
209                 # Anpassung mit Toleranzfaktor
210                 probabilities[start][end] += error *
                        tolerance
211 
212                 # Begrenzung auf [0,1]
213                 probabilities[start][end] = max(0.01, min
                        (0.99, probabilities[start][end]))
```

```
214
215         # Renormalisierung
216         for start in probabilities:
217             total = sum(probabilities[start].values())
218             if total > 0:
219                 probabilities[start] = {end: prob / total
220                                         for end, prob in
                                            probabilities[start
                                            ].items()}
221
222     # Falls kein Optimum erreicht wurde, nimm die beste
        Iteration
223     if best_probabilities is None:
224         # Finde Iteration mit hoechster Korrelation
225         best_idx = max(range(len(history)), key=lambda i:
            history[i][1])
226         best_iter, best_correlation, best_significance =
            history[best_idx]
227         best_probabilities = calculate_probabilities(
            count_transitions(empirical_chains))
228         print(f"\nKein Optimum erreicht. Beste Korrelation
            bei Iteration {best_iter}:")
229         print(f"  Korrelation = {best_correlation:.4f}")
230         print(f"  Signifikanz = {best_significance:.4f}")
231
232     return best_probabilities, best_correlation,
        best_significance, history
233
234 # Optimierung durchfuehren
235 optimized_probabilities, best_corr, best_sig, history =
    optimize_grammar(
236     empirical_chains, terminal_symbols, start_symbol,
237     max_iterations=500, tolerance=0.005, target_correlation
        =0.9
238 )
239
240 #
    ================================================================

241 # 7. VISUALISIERUNG DER OPTIMIERUNG
```

```
242 #
    ==========================================================================
243
244 def plot_optimization_history(history):
245     """Visualisiert den Optimierungsverlauf"""
246     iterations, correlations, p_values = zip(*history)
247
248     fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))
249
250     # Korrelationsverlauf
251     ax1.plot(iterations, correlations, 'b-', linewidth=1.5)
252     ax1.set_xlabel('Iteration')
253     ax1.set_ylabel('Korrelation (Pearson r)')
254     ax1.set_title('Optimierungsverlauf: Korrelation zwischen
        empirischen und generierten Haeufigkeiten')
255     ax1.grid(True, alpha=0.3)
256     ax1.axhline(y=0.9, color='r', linestyle='--', alpha=0.5,
        label='Zielkorrelation (0.9)')
257     ax1.legend()
258
259     # p-Wert-Verlauf (logarithmisch)
260     p_values = [max(p, 1e-10) for p in p_values]  #
        Vermeidung von log(0)
261     ax2.semilogy(iterations, p_values, 'g-', linewidth=1.5)
262     ax2.set_xlabel('Iteration')
263     ax2.set_ylabel('p-Wert (logarithmisch)')
264     ax2.set_title('Signifikanz der Korrelation')
265     ax2.grid(True, alpha=0.3)
266     ax2.axhline(y=0.05, color='r', linestyle='--', alpha=0.5,
         label='Signifikanzniveau (0.05)')
267     ax2.legend()
268
269     plt.tight_layout()
270     plt.savefig('optimierungsverlauf.png', dpi=150)
271     plt.show()
272
273 # Optional: Visualisierung (wenn matplotlib verfuegbar)
274 try:
275     plot_optimization_history(history)
```

```
276     print("\nOptimierungsverlauf wurde als '
            optimierungsverlauf.png' gespeichert.")
277 except:
278     print("\n(Hinweis: Fuer Visualisierung ist matplotlib
            erforderlich)")
279
280 #
    ================================================================

281 # 8. AUSGABE DER OPTIMIERTEN GRAMMATIK
282 #
    ================================================================


283
284 print("\n" + "=" * 70)
285 print("4. OPTIMIERTE PROBABILISTISCHE GRAMMATIK")
286 print("=" * 70)
287
288 # Nach Startzeichen sortierte Ausgabe
289 for start in sorted(optimized_probabilities.keys()):
290     transitions = optimized_probabilities[start]
291     transitions_str = ", ".join([f"'{end}': {prob:.3f}"
292                                  for end, prob in sorted(
                                        transitions.items())])
293     print(f"\n{start} -> {transitions_str}")
294
295 #
    ================================================================

296 # 9. VALIDIERUNG: VERGLEICH EMPIRISCHER UND GENERIERTER
    HAEUFIGKEITEN
297 #
    ================================================================


298
299 # Generiere neue Ketten mit optimierter Grammatik
300 validation_chains = generate_multiple_chains(
301     optimized_probabilities, start_symbol, n_chains=100,
            max_length=20
302 )
```

```
303 validation_frequencies = compute_frequencies(
        validation_chains, terminal_symbols)
304
305 print("\n" + "=" * 70)
306 print("5. VALIDIERUNG: EMPIRISCHE VS. GENERIERTE
        HAEUFIGKEITEN")
307 print("=" * 70)
308
309 table_data = []
310 for i, symbol in enumerate(terminal_symbols):
311     table_data.append([
312         symbol,
313         f"{empirical_frequencies[i]:.4f}",
314         f"{validation_frequencies[i]:.4f}",
315         f"{abs(empirical_frequencies[i] -
               validation_frequencies[i]):.4f}"
316     ])
317
318 print(tabulate(table_data,
319                headers=["Symbol", "Empirisch", "Generiert", "
                      Differenz"],
320                tablefmt="grid"))
321
322 # Gesamtkorrelation
323 final_corr, final_p = pearsonr(empirical_frequencies,
        validation_frequencies)
324 print(f"\nKorrelation (100 generierte Ketten): r = {
        final_corr:.4f}, p = {final_p:.4f}")
325
326 #
        ============================================================================
327 # 10. BEISPIEL-GENERIERTE KETTEN
328 #
        ============================================================================
329
330 print("\n" + "=" * 70)
331 print("6. BEISPIEL GENERIERTER TERMINALZEICHENKETTEN")
332 print("=" * 70)
```

```
333
334 example_chains = generate_multiple_chains(
335     optimized_probabilities, start_symbol, n_chains=5,
        max_length=15
336 )
337
338 for i, chain in enumerate(example_chains, 1):
339     chain_str = " -> ".join(chain)
340     print(f"\nKette {i} ({len(chain)} Symbole):")
341     print(f"  {chain_str}")
342
343 #
    ================================================================================

344 # 11. EXPORT DER GRAMMATIK ALS STRUKTUR
345 #
    ================================================================================

346
347 def export_grammar_as_pcfg(probabilities, filename="
    optimierte_grammatik.txt"):
348     """Exportiert die Grammatik im PCFG-Format"""
349     with open(filename, 'w', encoding='utf-8') as f:
350         f.write("# Optimierte probabilistische kontextfreie
            Grammatik (PCFG)\n")
351         f.write("# Generiert durch Algorithmisch Rekursive
            Sequenzanalyse 2.0\n\n")
352
353         for start in sorted(probabilities.keys()):
354             transitions = probabilities[start]
355             for end, prob in sorted(transitions.items()):
356                 f.write(f"{start} -> {end} [{prob:.3f}]\n")
357
358     print(f"\nGrammatik wurde als '{filename}' exportiert.")
359
360 export_grammar_as_pcfg(optimized_probabilities)
361
362 print("\n" + "=" * 70)
363 print("ALGORITHMISCH REKURSIVE SEQUENZANALYSE ABGESCHLOSSEN")
364 print("=" * 70)
```

Listing 2: Algorithmisch Rekursive Sequenzanalyse 2.0 - Vollständiger Code