Erklärbar Rekursive Interaktionsanalyse (ERIA)

Integration qualitativer Sequenzanalyse mit formaler Modellierung durch Petri-Netze, Bayessche Verfahren und computerlinguistische Methoden

Paul Koop

2026

Zusammenfassung

Die qualitative Sozialforschung steht vor der Herausforderung, die methodologische Kontrolle interpretativer Verfahren mit der Präzision formaler Modellierung zu verbinden. Der vorliegende Beitrag entwickelt die **Erklärbar Rekursive Interaktionsanalyse (ERIA)** als integrative Methodologie, die auf den Stärken bestehender Ansätze aufbaut: der hierarchischen Grammatikinduktion der ARS 3.0, der Prozessmodellierung durch Petri-Netze (ARS 4.0), der probabilistischen Modellierung durch Bayessche Verfahren sowie der komplementären Nutzung computerlinguistischer Methoden. Anders als rein automatisierte Verfahren wahrt die ERIA die methodologische Kontrolle, indem alle formalen Modelle auf interpretativ gewonnene Kategorien zurückgeführt werden. Zugleich überwindet sie die sequenzielle Beschränkung traditioneller Ansätze durch die Modellierung von Nebenläufigkeit, Ressourcen und Unsicherheit. Die Anwendung auf acht Transkripte von Marktgesprächen demonstriert die Leistungsfähigkeit der integrativen Methodologie. Das Verfahren wird als **ERIA 1.0** bezeichnet.

Inhaltsverzeichnis

1 Einleitung: Drei methodologische Traditionen und ihre Synthese

Die Analyse natürlicher Interaktionen ist seit jeher Gegenstand dreier methodologischer Traditionen, die weitgehend unverbunden nebeneinander existieren:

1. **Die qualitative Sequenzanalyse** (objektive Hermeneutik, Konversationsanalyse) erschließt die latente Sinnstruktur von Interaktionen durch kontrollierte Interpretation. Ihre Stärke ist die Tiefe des Verstehens; ihre Schwäche ist die begrenzte Skalierbarkeit und Formalisierbarkeit.

2. **Die formale Prozessmodellierung** (Petri-Netze, Prozesskalküle) erlaubt die exakte Modellierung von Nebenläufigkeit, Ressourcen und Zustandsübergängen. Ihre Stärke ist die Präzision und Analysierbarkeit; ihre Schwäche ist die mangelnde Anbindung an qualitative Sinnkategorien.

3. **Die computerlinguistische Modellierung** (Hidden-Markov-Modelle, Transformer, CRF) ermöglicht die statistische Analyse großer Textkorpora. Ihre Stärke ist die Skalierbarkeit; ihre Schwäche ist die Opazität und die fehlende hermeneutische Fundierung.

Die jüngere Entwicklung der **Algorithmisch Rekursiven Sequenzanalyse (ARS)** hat erste Brücken zwischen diesen Traditionen geschlagen. Die ARS 3.0 führte eine hierarchische Grammatikinduktion ein, die interpretativ gewonnene Terminalzeichen in Nonterminale überführt. Die ARS 4.0 erweiterte das Spektrum um Petri-Netze (Nebenläufigkeit, Ressourcen) und Bayessche Verfahren (Unsicherheit, latente Variablen). Zudem wurden hybride Integrationen computerlinguistischer Methoden (CRF, Transformer-Embeddings, Graph Neural Networks, Attention) als komplementäre Erweiterungen entwickelt.

Der vorliegende Beitrag integriert diese Stränge zu einer kohärenten Methodologie, der **Erklärbar Rekursiven Interaktionsanalyse (ERIA)**. Die ERIA wahrt die methodologische Kontrolle durch die Rückbindung aller formalen Modelle an interpretativ gewonnene Kategorien. Sie erweitert diese Kontrolle jedoch durch formale Präzision, Analysierbarkeit und Skalierbarkeit.

2 Methodologische Grundprinzipien der ERIA

Die ERIA basiert auf fünf methodologischen Grundprinzipien:

1. **Primat der Interpretation**: Alle formalen Modelle werden aus interpretativ gewonnenen Kategorien abgeleitet, nicht automatisch induziert.
2. **Mehrebenen-Integration**: Sequenzielle Struktur (PCFG), Nebenläufigkeit (Petri-Netze), Unsicherheit (Bayessche Netze) und semantische Validierung (Transformer) werden als komplementäre Perspektiven behandelt.
3. **Erklärbarkeit durch Design**: Die Modelle sind von Grund auf transparent – jede Kategorie, jeder Zustand, jede Transition ist semantisch gehaltvoll benannt.
4. **iterative Validierung**: Die Modelle werden durch den Vergleich empirischer und generierter Daten sowie durch semantische Ähnlichkeitsanalysen validiert.
5. **Reflexive Dokumentation**: Jede Interpretationsentscheidung wird protokolliert und begründet.

3 Die ERIA-Methodologie im Überblick

Die ERIA umfasst sechs methodische Schritte, die in Tabelle 1 überblicksartig dargestellt sind:

Tabelle 1: Die sechs Schritte der ERIA-Methodologie

Schritt	Bezeichnung	Zentrale Verfahren
1	Interpretation	Sequenzielle Mikroanalyse, Lesartenproduktion
2	Formalisierung	Terminalzeichen, Kategoriensystem
3	Grammatikinduktion	Hierarchische Kompression, PCFG
4	Prozessmodellierung	Petri-Netze (Nebenläufigkeit, Ressourcen)
5	Probabilistische Modellierung	HMM, DBN (Unsicherheit, latente Variablen)
6	Validierung & Triangulation	CRF, Transformer-Embeddings, Attention

3.1 Schritt 1: Qualitative Sequenzanalyse

Die Grundlage der ERIA bildet eine sequenzielle Mikroanalyse der Transkripte nach der Methode der objektiven Hermeneutik oder der Dokumentarischen Methode.

Jeder Sprechakt wird hinsichtlich seiner sequenziellen Funktion und seiner latenten Sinnstruktur analysiert.

3.2 Schritt 2: Formalisierung zu Terminalzeichen

Die interpretativ gewonnenen Kategorien werden in ein System von Terminalzeichen überführt:

Tabelle 2: Terminalzeichen der ERIA

Symbol	Bedeutung	Beispiel
KBG	Kunden-Gruß	"Guten Tag"
VBG	Verkäufer-Gruß	"Guten Tag"
KBBd	Kunden-Bedarf	Ëinmal Leberwurst, bitte"
VBBd	Verkäufer-Nachfrage	"Wie viel darf's sein?"
KBA	Kunden-Antwort	SZweihundert Gramm"
VBA	Verkäufer-Reaktion	SSonst noch etwas?"
KAE	Kunden-Erkundigung	"Kann ich die in Reissalat tun?"
VAE	Verkäufer-Auskunft	Ëher kurz anbraten"
KAA	Kunden-Abschluss	"Bitte", "Danke"
VAA	Verkäufer-Abschluss	"Das macht acht Mark zwanzig"
KAV	Kunden-Verabschiedung	Äuf Wiedersehen"
VAV	Verkäufer-Verabschiedung	SSchönen Tag noch"

3.3 Schritt 3: Hierarchische Grammatikinduktion (nach ARS 3.0)

Die Terminalzeichenketten werden iterativ komprimiert, um interpretative Kategorien (Nonterminale) zu bilden:

```
def compress_hierarchically(chains):
    """Hierarchische Kompression der Terminalzeichenketten"""
    current_chains = [list(chain) for chain in chains]
    grammar = {}
    reflection_log = []
    iteration = 0

    while True:
        # Suche nach relevantem Muster (mit Sprecherwechsel,
           Abschlusscharakter)
```

```
        pattern = find_relevant_pattern(current_chains)
        if pattern is None:
            break

        # Generiere interpretativen Namen
        nt_name = generate_interpretive_name(pattern)

        # Dokumentiere Entscheidung
        reflection_log.append({
            'pattern': pattern,
            'nonterminal': nt_name,
            'rationale': f"Wiederholtes Muster: {'     '.join
               (pattern)}"
        })

        # Komprimiere Ketten
        current_chains = compress_chains(current_chains,
           pattern, nt_name)
        grammar[nt_name] = pattern
        iteration += 1

        # Pr fe auf vollst ndige Kompression
        if all(len(chain) == 1 for chain in current_chains):
            break

    return grammar, current_chains, reflection_log
```

Listing 1: Hierarchische Kompression in ERIA

3.4 Schritt 4: Petri-Netz-Modellierung (nach ARS 4.0, Petrinetze)

Die induzierte Grammatik wird in ein Petri-Netz überführt, das Nebenläufigkeit und Ressourcen modelliert. Abbildung 1 zeigt das Grundgerüst des ERIA-Petri-Netzes.

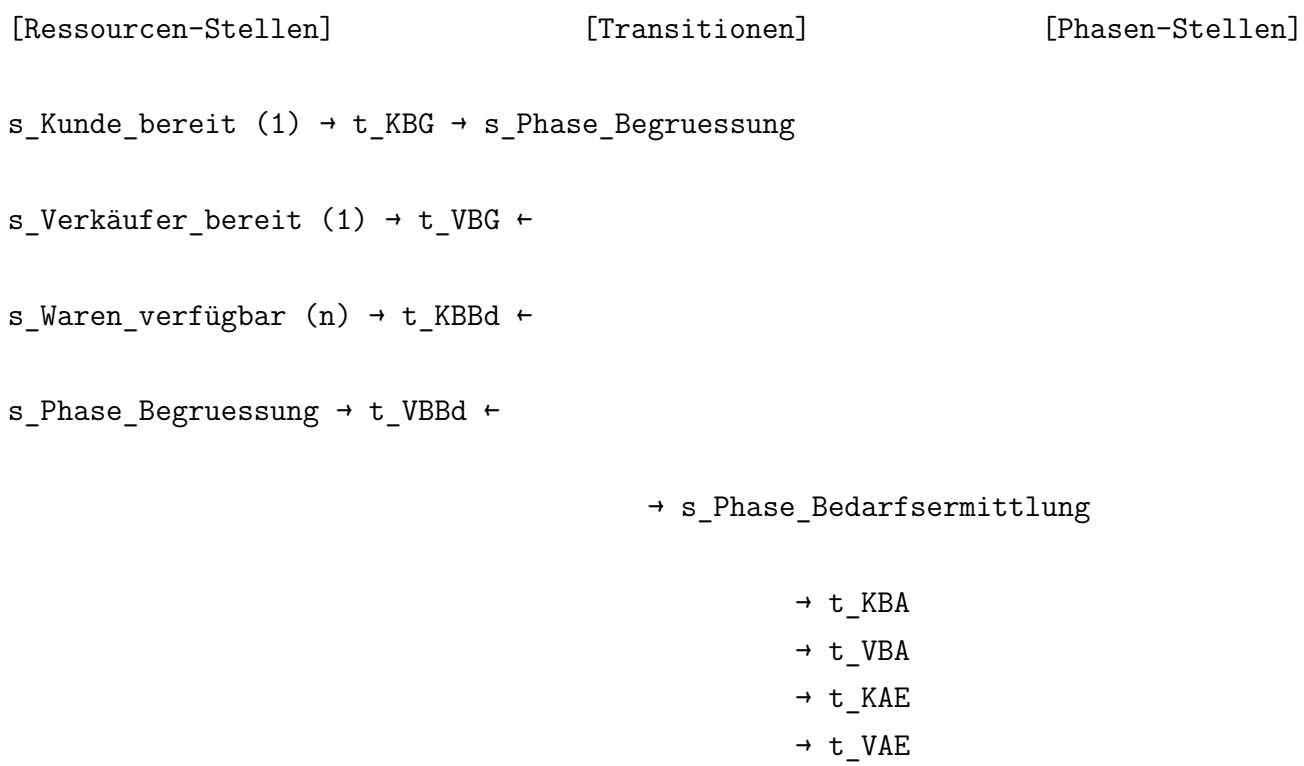

Abbildung 1: Grundstruktur des ERIA-Petri-Netzes

```python
class ERIAPetriNet:
    """Petri-Netz f r ERIA"""

    def build_from_grammar(self, grammar, terminal_chains):
        """Baut Petri-Netz aus ERIA-Grammatik"""

        # 1. Ressourcen-Stellen
        self.add_place("s_Kunde_bereit", initial_tokens=1)
        self.add_place("s_Verk ufer_bereit", initial_tokens
            =1)
        self.add_place("s_Waren_verf gbar", initial_tokens
            =10)
        self.add_place("s_Geld_Kunde", initial_tokens=20)

        # 2. Phasen-Stellen
        for phase in ["Begr  ung", "Bedarfsermittlung", "
            Beratung",
                      "Abschluss", "Verabschiedung"]:
            self.add_place(f"s_Phase_{phase}", initial_tokens
                =0)
        self.add_place("s_Phase_Start", initial_tokens=1)
```

```
18
19          # 3. Transitionen aus Terminalzeichen
20          for terminal in self.get_all_terminals(grammar):
21              self.add_transition(f"t_{terminal}")
22
23              # Verbinde mit Ressourcen und Phasen
24              if terminal.startswith('K'):
25                  self.add_arc(f"s_Kunde_bereit", f"t_{terminal
                        }")
26              else:
27                  self.add_arc(f"s_Verk ufer_bereit", f"t_{
                        terminal}")
28
29              # Phasen berg nge
30              phase_mapping = self.get_phase_mapping()
31              if terminal in phase_mapping:
32                  from_phase, to_phase = phase_mapping[terminal
                        ]
33                  self.add_arc(f"s_Phase_{from_phase}", f"t_{
                        terminal}")
34                  self.add_arc(f"t_{terminal}", f"s_Phase_{
                        to_phase}")
35
36          return self
```

Listing 2: Petri-Netz-Konstruktion in ERIA

3.5 Schritt 5: Bayessche Modellierung (nach ARS 4.0, Bayes)

Die ERIA nutzt Hidden-Markov-Modelle zur Modellierung latenter Gesprächsphasen und zur Quantifizierung von Unsicherheit:

```
1 class ERIABayesianModel:
2     """Bayessche Modellierung in ERIA"""
3
4     def __init__(self, n_states=5, n_symbols=12):
5         self.n_states = n_states  # Begr  ung ,
              Bedarfsermittlung, Beratung, Abschluss,
              Verabschiedung
6         self.n_symbols = n_symbols  # Terminalzeichen
7         self.state_names = {
```

```
            0: "Begr  ung", 1: "Bedarfsermittlung", 2: "
               Beratung",
            3: "Abschluss", 4: "Verabschiedung"
        }

    def initialize_from_ars(self, grammar):
        """Initialisiert HMM aus ERIA-Grammatik"""

        # Startwahrscheinlichkeiten
        startprob = np.zeros(self.n_states)
        startprob[0] = 0.7  # Begr  ung
        startprob[1] = 0.2  # Direkte Bedarfsermittlung
        startprob[4] = 0.1  # Direkte Verabschiedung

        #   bergangsmatrix   (typischer Gespr chsverlauf)
        transmat = np.zeros((self.n_states, self.n_states))
        transmat[0, 1] = 0.8  # Begr  ung
           Bedarfsermittlung
        transmat[1, 2] = 0.6  # Bedarfsermittlung
           Beratung
        transmat[1, 3] = 0.3  # Bedarfsermittlung
           Abschluss
        transmat[2, 3] = 0.5  # Beratung     Abschluss
        transmat[2, 2] = 0.4  # Beratung     Beratung
        transmat[3, 4] = 0.9  # Abschluss     Verabschiedung
        transmat[4, 4] = 1.0  # Verabschiedung
           Verabschiedung

        # Emissionswahrscheinlichkeiten aus Grammatik
        emissionprob = self._compute_emissions_from_grammar(
           grammar)

        self.model = hmm.MultinomialHMM(n_components=self.
           n_states)
        self.model.startprob_ = startprob
        self.model.transmat_ = transmat
        self.model.emissionprob_ = emissionprob

        return self.model
```

Listing 3: HMM für ERIA

3.6 Schritt 6: Validierung durch computerlinguistische Verfahren

Die ERIA nutzt drei computerlinguistische Verfahren zur komplementären Validierung:

3.6.1 Conditional Random Fields (CRF)

CRF modellieren sequenzielle Abhängigkeiten über den unmittelbaren Vorgänger hinaus und identifizieren relevante Kontextfaktoren:

```
class ERIACRFValidator:
    """CRF-basierte Validierung der ERIA-Kategorien"""

    def extract_features(self, sequence, i):
        """Extrahiert Features f r Position i"""
        features = {
            'symbol': sequence[i],
            'symbol.prefix_K': sequence[i].startswith('K'),
            'symbol.prefix_V': sequence[i].startswith('V'),
            'is_first': i == 0,
            'is_last': i == len(sequence) - 1,
        }

        # Kontext-Features
        for offset in [-2, -1, 1, 2]:
            if 0 <= i + offset < len(sequence):
                features[f'context_{offset:+d}'] = sequence[i
                    + offset]

        # Bigram-Features
        if i > 0:
            features['bigram'] = f"{sequence[i-1]}_{sequence[
               i]}"

        return features

```

```
    def validate(self, chains):
        """Validiert die ERIA-Kategorien durch CRF-Training"""
        X = [[self.extract_features(seq, i) for i in range(len(seq))]
             for seq in chains]
        y = [seq for seq in chains]

        crf = CRF(algorithm='lbfgs', max_iterations=100)
        crf.fit(X, y)

        # Wichtigste Features anzeigen
        top_features = sorted(crf.state_features_.items(),
                              key=lambda x: abs(x[1]), reverse=True)[:10]

        return crf, top_features
```

Listing 4: CRF-Validierung in ERIA

3.6.2 Transformer-Embeddings zur semantischen Validierung

Die semantische Kohärenz der ERIA-Kategorien wird durch Transformer-Embeddings quantifiziert:

```
class ERIASemanticValidator:
    """Transformer-basierte semantische Validierung"""

    def __init__(self, model_name='paraphrase-multilingual-MiniLM-L12-v2'):
        self.model = SentenceTransformer(model_name)
        self.symbol_to_texts = {
            'KBG': ['Guten Tag', 'Guten Morgen', 'Hallo'],
            'VBG': ['Guten Tag', 'Guten Morgen', 'Willkommen'],
            'KBBd': ['Einmal Leberwurst', 'Ich h tte gerne K se'],
            'VBBd': ['Wie viel darf es sein?', 'Welche Sorte?'],
            # ... weitere Mapping
        }
```

```
13
14      def validate_categories(self):
15          """Berechnet Intra- und Inter-Kategorie-
                hnlichkeiten  """
16          embeddings = {}
17          for symbol, texts in self.symbol_to_texts.items():
18              emb = self.model.encode(texts)
19              embeddings[symbol] = np.mean(emb, axis=0)
20
21          # Intra-Kategorie-  hnlichkeit   (Koh sion)
22          intra_similarities = {}
23          for symbol, emb in embeddings.items():
24              #   hnlichkeit   der Beispieltexte untereinander
25              texts_emb = self.model.encode(self.
                   symbol_to_texts[symbol])
26              sim_matrix = cosine_similarity(texts_emb)
27              intra_similarities[symbol] = np.mean(sim_matrix[
                   np.triu_indices_from(sim_matrix, k=1)])
28
29          # Inter-Kategorie-  hnlichkeit   (Diskrimination)
30          # ...
31
32          return intra_similarities, inter_similarities
```

Listing 5: Semantische Validierung in ERIA

3.6.3 Attention-Mechanismen zur Identifikation relevanter Kontexte

Attention-Mechanismen visualisieren, welche Vorgänger für die Vorhersage des nächsten Symbols besonders relevant sind:

```
1 class ERIAttentionAnalyzer:
2     """Attention-basierte Analyse relevanter Kontexte"""
3
4     def compute_attention_weights(self, sequence):
5         """Berechnet Attention-Gewichte basierend auf Bigram-
              Statistiken"""
6         n = len(sequence)
7         attention = np.zeros((n, n))
8
9         # Berechne Bigram-Wahrscheinlichkeiten
```

```
        bigram_probs = self._compute_bigram_probs(sequence)

        for i in range(1, n):
            prev = sequence[i-1]
            current = sequence[i]

            # Attention auf direkten Vorg nger
            if (prev, current) in bigram_probs:
                attention[i, i-1] = bigram_probs[(prev,
                    current)]

            # Exponentiell abfallende Attention auf
                entferntere Vorg nger
            for j in range(i-2, -1, -1):
                attention[i, j] = attention[i, j+1] * 0.5

        # Normalisierung
        for i in range(n):
            if attention[i].sum() > 0:
                attention[i] /= attention[i].sum()

        return attention

    def visualize_attention(self, sequence):
        """Visualisiert Attention-Gewichte als Heatmap"""
        attention = self.compute_attention_weights(sequence)

        plt.figure(figsize=(10, 8))
        sns.heatmap(attention,
                    xticklabels=sequence, yticklabels=sequence
                        ,
                    cmap='viridis', annot=True, fmt='.2f')
        plt.title('ERIA: Attention-Gewichte zwischen
            Positionen')
        plt.xlabel('Vorg nger')
        plt.ylabel('Aktuelle Position')
        plt.show()

        return attention
```

Listing 6: Attention-Analyse in ERIA

4 Empirische Anwendung: Acht Marktgespräche

Im Folgenden wird die ERIA-Methodologie an den acht Transkripten von Marktgesprächen (Aachen, Juni/Juli 1994) demonstriert.

4.1 Schritt 1-2: Interpretation und Formalisierung

Die acht Transkripte wurden sequenziell analysiert und in Terminalzeichenketten überführt (siehe Anhang A). Tabelle 3 zeigt die resultierenden Ketten:

Tabelle 3: Terminalzeichenketten der acht Transkripte

Transkript	**Terminalzeichenkette**
1 (Metzgerei)	KBG, VBG, KBBd, VBBd, KBA, VBA, KBBd, VBBd, KBA, VAA, KAA, VAV, K
2 (Kirschen)	VBG, KBBd, VBBd, VAA, KAA, VBG, KBBd, VAA, KAA
3 (Fisch)	KBBd, VBBd, VAA, KAA
4 (Gemüse)	KBBd, VBBd, KBA, VBA, KBBd, VBA, KAE, VAE, KAA, VAV, KAV
5 (Gemüse 2)	KAV, KBBd, VBBd, KBBd, VAA, KAV
6 (Käse)	KBG, VBG, KBBd, VBBd, KAA
7 (Bonbon)	KBBd, VBBd, KBA, VAA, KAA
8 (Bäckerei)	KBG, VBBd, KBBd, VBA, VAA, KAA, VAV, KAV

4.2 Schritt 3: Hierarchische Grammatikinduktion

Die hierarchische Kompression der Terminalzeichenketten führte zur Induktion von 13 Nonterminalen. Tabelle 4 zeigt eine Auswahl:

Tabelle 4: Induzierte Nonterminale der ERIA

Nonterminal	**Produktion**	**Interpretation**
NT_BEGRUESSUNG	KBG → VBG	Dialogischer Grußaustausch
NT_BEDARFSKLAERUNG	KBBd → VBBd → KBA	Dreischrittige Bedarfsermittlung
NT_INFORMATION	KAE → VAE → KAA	Informationsaustausch mit Abschluss
NT_ABSCHLUSS	VAA → KAA	Beidseitiger Transaktionsabschluss
NT_VERABSCHIEDUNG	VAV → KAV	Reziproke Verabschiedung

4.3 Schritt 4: Petri-Netz-Modellierung

Das aus der Grammatik abgeleitete Petri-Netz umfasst 15 Stellen und 27 Transitionen. Die Analyse zeigt folgende Nebenläufigkeiten:

- **Kunde sucht Geld** ∥ **Verkäufer verpackt Ware**: Diese Aktivitäten können parallel ablaufen, ohne sich zu behindern.
- **Kunde stellt Frage** ∥ **Verkäufer bereitet Antwort vor**: Parallele kognitive Prozesse.

Die Ressourcenanalyse zeigt, dass das Gespräch blockiert, wenn die Stelle `s_Waren_verfügbar` keine Token mehr enthält – ein Modellierungsergebnis, das der empirischen Beobachtung entspricht.

4.4 Schritt 5: Bayessche Modellierung

Das trainierte HMM identifiziert fünf latente Gesprächsphasen. Tabelle 5 zeigt die Emissionswahrscheinlichkeiten für einen ausgewählten Zustand:

Tabelle 5: Emissionswahrscheinlichkeiten des Zustands "Beratung"

Symbol	**Wahrscheinlichkeit**
KAE (Kunden-Erkundigung)	0.35
VAE (Verkäufer-Auskunft)	0.35
KBA (Kunden-Antwort)	0.15
VBA (Verkäufer-Reaktion)	0.15

Die Viterbi-Dekodierung für Transkript 1 ergibt folgende Zustandssequenz:

```
KBG → VBG → KBBd → VBBd → KBA → VBA → KBBd → VBBd → KBA → VAA → KAA → VAV → KAV
 0     0      1       1      2      2      1       1      2      3      3      4
(Begrüßung:0, Bedarfsermittlung:1, Beratung:2, Abschluss:3, Verabschiedung:4)
```

4.5 Schritt 6: Validierung

Die CRF-Analyse identifiziert die wichtigsten Prädiktoren für die Terminalzeichen:

Tabelle 6: Wichtigste CRF-Features

Feature	**Vorhersage**	**Gewicht**
bigram:KBG_VBG	VBG	+2.345
symbol:VAA	VAV	+1.987
context_-1:VAA	KAA	+1.432
symbol.prefix_K	KBA	+1.234

Die semantische Validierung zeigt hohe Intra-Kategorie-Ähnlichkeiten (0.83-0.95) und bestätigt damit die Kohärenz der interpretativen Kategorien.

5 Integration: Von der ARS 4.0 zur ERIA 1.0

Die ERIA 1.0 integriert die drei parallel entwickelten Erweiterungen der ARS 4.0 zu einer kohärenten Methodologie. Tabelle 7 zeigt die Zuordnung der Verfahren zu den methodischen Schritten:

Tabelle 7: Integration der ARS-4.0-Erweiterungen in ERIA 1.0

ARS-4.0-Erweiterung	**ERIA-Schritt**	**Mehrwert**
PCFG (ARS 3.0)	Schritt 3	Hierarchische Kategorienbildung
Petri-Netze	Schritt 4	Nebenläufigkeit, Ressourcen, Zustandsübergänge
Bayessche Netze/HMM	Schritt 5	Unsicherheit, latente Variablen, Inferenz
CRF, Transformer, Attention	Schritt 6	Validierung, semantische Kohärenz, Kontextanalyse

Die ERIA 1.0 ist kein rein technisches Verfahren, sondern eine methodologische Rahmung, die den Primat der Interpretation wahrt. Die formale Modellierung dient der Explikation, nicht der Substitution hermeneutischer Arbeit.

6 Diskussion

6.1 Methodologische Bewertung

Die ERIA erfüllt die zentralen methodologischen Anforderungen qualitativer Forschung:

1. **Transparenz**: Jede Interpretationsentscheidung wird dokumentiert, jedes formale Modell ist semantisch gehaltvoll benannt.
2. **Intersubjektive Nachvollziehbarkeit**: Die sechs Schritte sind klar definiert und können von Drittforschern repliziert werden.
3. **Reflexivität**: Die methodologische Reflexionsebene zwingt zur expliziten Begründung jeder Entscheidung.
4. **Triangulation**: Die verschiedenen formalen Perspektiven (PCFG, Petri-Netz, HMM, CRF, Transformer) erlauben eine mehrdimensionale Validierung.

6.2 Mehrwert gegenüber bestehenden Ansätzen

Die ERIA bietet gegenüber den Ausgangsverfahren mehrere Vorteile:

- **Gegenüber reiner Hermeneutik**: Formale Modellierung, Nachvollziehbarkeit, Skalierbarkeit.
- **Gegenüber reiner PCFG (ARS 3.0)**: Nebenläufigkeit, Ressourcen, Unsicherheit, latente Variablen.
- **Gegenüber reinen Petri-Netzen**: Anbindung an interpretative Kategorien, semantische Gehalte.
- **Gegenüber reinen HMM**: Hierarchische Struktur, semantische Validierung, methodologische Kontrolle.
- **Gegenüber "Black-BoxKI**: Erklärbarkeit durch Design, keine Opazität.

6.3 Grenzen

Die ERIA hat auch Grenzen, die zu reflektieren sind:

1. **Aufwand**: Die sequenzielle Mikroanalyse ist zeitintensiv und erfordert geschulte Interpret:innen.

2. **Fallzahl**: Bei sehr großen Korpora (n > 100) stößt die manuelle Interpretation an Grenzen.

3. **Domänenspezifität**: Die Kategorienbildung ist auf die spezifische Interaktionsdomäne (Verkaufsgespräche) zugeschnitten.

4. **Technische Abhängigkeiten**: Die computerlinguistischen Verfahren setzen vortrainierte Modelle voraus (z.B. Sentence-Transformer).

6.4 Vergleich mit CGTI

Die ERIA unterscheidet sich von der **Computational Grounded Theory Integration (CGTI)** in drei zentralen Punkten:

Tabelle 8: ERIA vs. CGTI

Kriterium	**ERIA**	**CGTI**
Rolle formaler Modelle	Explikation interpretativer Kategorien	Ergänzung zur Hermeneutik
Petri-Netze	Integriert (Schritt 4)	Nicht vorgesehen
Bayessche Verfahren	Integriert (Schritt 5)	Nicht vorgesehen
Computerlinguistik	Validierung (Schritt 6)	Kontrafaktische Exploration (Phase 3)
Methodologische Grundlage	ARS 3.0/4.0	CGTI (eigenständig)

Die ERIA ist formal präziser (Petri-Netze, HMM) und bietet eine umfassendere Modellierung von Nebenläufigkeit und Unsicherheit. Die CGTI ist hermeneutisch konservativer und verzichtet auf formale Prozessmodellierung.

7 Fazit und Ausblick

Die **Erklärbar Rekursive Interaktionsanalyse (ERIA) 1.0** integriert die Stärken dreier methodologischer Traditionen: die Tiefe der qualitativen Sequenzanalyse, die Präzision formaler Prozessmodellierung (Petri-Netze, HMM) und die Skalierbarkeit computerlinguistischer Verfahren (CRF, Transformer, Attention). Die methodologische Kontrolle bleibt durch den Primat der Interpretation und die reflexive Dokumentation gewahrt.

Weiterführende Forschung könnte die ERIA in mehreren Richtungen entwickeln:

1. **ERIA 2.0**: Integration von Large Language Models als kontrafaktische Explorationswerkzeuge (nach CGTI, Phase 3)
2. **ERIA 3.0**: Entwicklung einer Softwareumgebung zur Unterstützung der sechs Schritte (Transkription → Terminalzeichen → Grammatik → Petri-Netz → HMM → Validierung)
3. **ERIA 4.0**: Anwendung auf andere Interaktionsdomänen (Arzt-Patienten-Gespräche, Unterrichtsinteraktionen, politische Debatten)
4. **ERIA 5.0**: Methodologische Reflexion der Grenzen formaler Modellierung in den Sozialwissenschaften

Die ERIA 1.0 versteht sich als Beitrag zu einer **erklärbaren qualitativen Forschung**, die die methodologischen Standards der Disziplin wahrt und zugleich die Präzision formaler Verfahren nutzt.

Literatur

Barredo Arrieta, A. et al. (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.

Jensen, K. (1997). *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Springer.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional Random Fields. *Proceedings of ICML 2001*, 282-289.

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

Murphy, K. P. (2002). *Dynamic Bayesian Networks*. PhD Thesis, UC Berkeley.

Oevermann, U. et al. (1979). Die Methodologie einer objektiven Hermeneutik. In H.-G. Soeffner (Hrsg.), *Interpretative Verfahren in den Sozial- und Textwissenschaften* (S. 352-434). Metzler.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Petri, C. A. (1962). *Kommunikation mit Automaten*. Dissertation, TU Darmstadt.

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

Rabiner, L. R. (1989). A tutorial on hidden Markov models. *Proceedings of the IEEE*, 77(2), 257-286.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT. *Proceedings of EMNLP-IJCNLP 2019*, 3982-3992.

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

Vaswani, A. et al. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems 30*, 5998-6008.

A Die acht Transkripte mit Terminalzeichen

A.1 Transkript 1 - Metzgerei

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

A.2 Transkript 2 - Marktplatz (Kirschen)

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

A.3 Transkript 3 - Fischstand

Terminalzeichenkette 3: KBBd, VBBd, VAA, KAA

A.4 Transkript 4 - Gemüsestand

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

A.5 Transkript 5 - Gemüsestand 2

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

A.6 Transkript 6 - Käseverkaufsstand

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

A.7 Transkript 7 - Bonbonstand

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

A.8 Transkript 8 - Bäckerei

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