Between Interpretation and Computation

Algorithmic Recursive Sequence Analysis as a Bridge between Qualitative Hermeneutics and Formal Modeling

Paul Koop

June/July 1994 & 2024

Abstract

Qualitative social research currently faces a methodological dilemma: On one hand, generative AI systems promise an unprecedented scaling of interpretive work steps; on the other hand, due to their stochastic nature, they elude the classical validation logic of qualitative research. This paper argues that this dilemma can be resolved by revisiting formalizing approaches that were already present in the tradition of text analysis but were forgotten due to recent developments in generative AI. As a concrete solution, the paper develops **Algorithmic Recursive Sequence Analysis (ARS)**, a procedure that transforms interpretive processes into a formal grammar, making them transparent, reproducible, and intersubjectively verifiable. The connection to current discussions on **Explainable AI (XAI)** proves to be doubly fruitful: It provides a conceptual framework to reflect on the quality of qualitative interpretations and reminds us that explainability is not a luxury but a necessity—in technology as well as in science. The empirical application to eight transcripts of sales conversations demonstrates the effectiveness of the procedure.

Contents

1 Introduction: The Paradox of Qualitative Research in the Age of Generative AI

Qualitative social research currently faces a methodological dilemma. On one hand, generative AI systems promise an unprecedented scaling of interpretive work steps. On the other hand, due to their stochastic nature, these systems elude the classical validation logic of qualitative research. Where the latter traditionally relies on detailed disclosure of the coding process and intersubjective comprehensibility, there is now a blind reliance on the supposed "emergence" of neural networks.

This trend is problematic because it decouples computer-assisted text analysis from its methodological foundations. At the same time, however, it points to a deficit that affects qualitative research itself: It lacks a formalized vocabulary to make its interpretive processes accessible to algorithmic procedures. The result is a choice between two unsatisfactory options: either renouncing scaling or abandoning methodological control.

This paper argues that this dilemma can be resolved by revisiting formalizing approaches that were already present in the tradition of text analysis but were forgotten due to recent developments in generative AI. As a concrete solution, the paper develops **Algorithmic Recursive Sequence Analysis (ARS)**, a procedure that transforms interpretive processes into a formal grammar, making them transparent, reproducible, and intersubjectively verifiable.

The point of this approach lies in its connection to current discussions on **Explainable Artificial Intelligence (XAI)** . XAI has developed as a response to the opacity of neural networks (Samek & Müller, 2019; Barredo Arrieta et al., 2020). The central insight is: Those who cannot comprehend the decisions of complex AI systems cannot trust them—and should not use them in safety-critical areas (Weller, 2019). This insight, so the thesis of this paper, can be productively applied to qualitative research: It also needs procedures that make its interpretive processes explainable. ARS understands itself as such a procedure—as a contribution to an **explainable qualitative research** that preserves the methodological standards of the discipline while simultaneously opening up to algorithmic modeling.

The paper is structured as follows: Section 2 introduces the concept of Explainable AI and develops the analogy to qualitative research. Section 3 presents ARS in its methodological architecture. Section 4 documents the empirical application to eight transcripts of sales conversations. Section 5 reflects on the results in light of the XAI discussion. Section 6 draws a conclusion and outlines perspectives.

2 Explainable AI: Concept, Development, and Methodological Relevance

2.1 Origins and Fundamental Ideas of XAI

The development of Explainable Artificial Intelligence (XAI) is closely linked to the realization that the increasing performance of complex AI models comes with a loss of transparency. In particular, deep neural networks, which achieve impressive results in numerous application domains, operate as "black boxes": Their internal decision processes are not directly comprehensible to developers or users (Samek & Müller, 2019, p. 2).

This opacity becomes problematic when AI systems are used in safety-critical areas—in medical diagnostics, jurisprudence, finance, or autonomous control (Ortigossa et al., 2024, p. 80800). Wrong decisions can have serious consequences here. At the same time, the opacity of the models makes it difficult to identify bias and discrimination. A frequently cited case is the COMPAS system for recidivism prediction, which systematically disadvantaged African American defendants without this bias being recognizable from the model architecture (Barredo Arrieta et al., 2020, p. 84).

XAI research responds to this problem by developing methods to subsequently explain the decisions of complex models or to design interpretable models from the outset (Mersha et al., 2024). The term "Explainable AI" itself originates from an initiative of the US research agency DARPA, which from 2015 onwards specifically funded projects on the explainability of AI systems (Barredo Arrieta et al., 2020, p. 86). Since then, XAI has developed into an independent research field addressing both technical and ethical as well as legal questions.

An important legal driver of the XAI discussion was the European General Data Protection Regulation (GDPR). In particular, Recital 71 is often interpreted in research as the basis of a "right to explanation", even though the regulation does not formulate an explicit, enforceable right to complete algorithmic disclosure (Wachter et al., 2017). Nevertheless, the GDPR establishes binding requirements for transparency, comprehensibility, and information obligations in automated decisions, thereby reinforcing the normative pressure to develop explainable AI systems.

2.2 Central Concepts and Taxonomies

The XAI literature has developed a series of concepts and distinctions to structure the field. **Explainability** generally denotes the property of an AI system to present its decisions in a way that is understandable to humans (Barredo Arrieta et al., 2020, p. 89). **Interpretability** aims at enabling a human observer to comprehend the functioning of the system (Weller, 2019, p. 25). **Transparency** means the disclosure of systemic processes and design decisions (Weller, 2019, p. 27).

A fundamental taxonomic distinction concerns the timing of explainability: **Ad-hoc methods** (also "Explanation by Design") integrate explainability into the model architecture from the beginning. They design models that are principally interpretable due to their structure—such as decision trees or rule-based systems. **Post-hoc methods**, on the other hand, apply explanation techniques to already trained black-box models. They attempt to retrospectively reconstruct which input factors were decisive for a particular decision (Barredo Arrieta et al., 2020, p. 92).

A second distinction concerns the scope of explanation: **Global explanations** target the overall behavior of the model—they answer the question of how the model fundamentally functions. **Local explanations**, on the other hand, refer to individual decisions—they explain why a specific input led to a specific output (Ortigossa et al., 2024, p. 80805).

A third distinction concerns methodology: **Model-specific procedures** are only applicable to certain model architectures (e.g., neural networks). **Model-agnostic procedures**, on the other hand, can be used independently of the concrete model architecture (Mersha et al., 2024, p. 3).

Among the best-known XAI procedures are:

- **LIME (Local Interpretable Model-agnostic Explanations)**: A model-agnostic procedure that learns simple, interpretable local surrogate models to explain the decisions of complex black-box models (Barredo Arrieta et al., 2020, p. 102).
- **SHAP (SHapley Additive exPlanations)**: A procedure based on cooperative game theory that quantifies the contribution of each input feature to a prediction (Barredo Arrieta et al., 2020, p. 104).
- **Saliency Maps**: Visualizations that show for image classifiers which image regions were particularly relevant for a decision (Zhou et al., 2019).

- **Layer-wise Relevance Propagation (LRP)**: A procedure that propagates the prediction of a neural network backwards layer by layer, thus identifying relevant input regions (Montavon et al., 2019).

2.3 XAI as a Methodological Challenge

The XAI discussion is not limited to technical procedures. It touches on fundamental methodological questions: What does it mean to "explain" a decision? Who is the addressee of the explanation? What quality criteria apply to explanations?

NIST (National Institute of Standards and Technology) has formulated three fundamental properties of good explanations (Ortigossa et al., 2024, p. 80810):

1. **Meaningfulness**: Explanations must be understandable to the intended addressee. This requires adaptation to their prior knowledge and cognitive abilities.

2. **Accuracy**: Explanations must correctly represent the actual decision processes of the model. There is a potential conflict of goals with meaningfulness: An accurate but highly complex explanation may be incomprehensible; a comprehensible but inaccurate explanation may be misleading.

3. **Knowledge Limits**: Good explanations make clear under which conditions the model works reliably and where its limits lie.

These criteria are relevant not only for technical systems. They can, as this paper argues, be transferred to qualitative research. Qualitative interpretations must also be understandable (for the scientific community), accurate (in the sense of fidelity to the text), and state their limits (e.g., regarding the scope of interpretation). The XAI discussion thus provides a conceptual framework to reflect on the quality of qualitative interpretations—and to develop procedures that ensure this quality.

2.4 From XAI to Explainable Qualitative Research: An Analogy

The transfer of the XAI perspective to qualitative research is based on an analogy systematized in Table 1:

The point of this analogy lies in the reversal of perspective: While XAI asks how to explain the decisions of *technical* systems, explainable qualitative research asks how to make the interpretation processes of *human* researchers explainable. In both cases,

Table 1: Analogy between Technical XAI and Qualitative Research

Dimension	Technical XAI	Qualitative Research
Problem	Opaque decisions of neural networks	Opaque interpretation processes
Cause	Subsymbolic representations	Implicit rule knowledge
Consequence	Lack of trust, undetected bias	Lack of intersubjectivity
Solution	Explication of decision bases	Explication of interpretation rules
Methods	LIME, SHAP, Saliency Maps	ARS, explicit category formation
Criteria	Meaningfulness, Accuracy, Knowledge Limits	Comprehensibility, Text fidelity, Scope

it is about transforming implicit, opaque operations into explicit, comprehensible rules.

Algorithmic Recursive Sequence Analysis, presented in the following, understands itself as a procedure that accomplishes this transformation. It formalizes interpretation processes without automating them. It produces explicit, verifiable models without eliminating hermeneutic openness. And it thus creates the prerequisites for a qualitatively substantial but methodologically controlled use of algorithmic procedures.

3 Algorithmic Recursive Sequence Analysis: Methodological Architecture

3.1 Basic Operations: From Transcription to Terminal Symbol String

ARS operates on transcripts of natural interactions. The first step consists of a detailed sequential analysis following the logic of qualitative interpretation. Qualitative sequence analysis, as developed in objective hermeneutics (Oevermann et al., 1979) and conversation analysis (Sacks et al., 1974), aims to uncover the latent meaning structure of interactions through the systematic reconstruction of their sequential order. Each speech act is analyzed with regard to its sequential function and its intentional quality.

The analysis follows the principle of **interpretation production and falsification**

(Oevermann et al., 1979, p. 392): For each sequential step, alternative interpretation possibilities are generated and systematically tested against the further course of the interaction. This procedure of “controlled interpretation” (Flick, 2019, p. 158) ensures intersubjective comprehensibility and forces the explication of interpretation rules.

The result of this interpretive work is a **terminal symbol string**, in which each speech act is represented by a symbol from a previously developed category system. These terminal symbols function as a formalized equivalent of qualitative coding (Przyborski & Wohlrab-Sahr, 2021, p. 207). The following table illustrates this using an example from a transcript:

Table 2: Example of Terminal Symbol Assignment

Transcript Excerpt	**Terminal Symbol**	**Interpretation**
Customer: Good day	KBG	Customer greeting (initiation of interaction)
Salesperson: Good day	VBG	Salesperson greeting (reciprocal confirmation)
Customer: One portion of coarse liver sausage, please.	KBBd	Customer need (articulation of purchase desire)

3.2 Grammar Induction: From Individual Cases to Generative Models

Based on the terminal symbol strings, an individual grammar is induced for each transcript. This grammar specifies which sequence patterns are observable in the respective transcript and which transitions between terminal symbols are possible. Formally, it is a transition-based grammar operating at the level of terminal symbols, whose production rules are based on observed transition frequencies.

Unlike classical linguistic PCFGs (Manning & Schütze, 1999), ARS dispenses with explicit non-terminals and deep recursive derivations. Instead, the grammar models sequential regularities as probabilistic transitions between formalized speech act categories. The term grammar is used here in a methodological, not a strictly formal-linguistic sense: as an explicit, generative rule system for reconstructing observable sequence structures.

Induction is performed by simply counting observed transitions:

```python
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: Counting Transitions between Terminal Symbols

3.3 Unification and Optimization

The individual grammars are merged into a **unified grammar** covering the sequence structure of all transcripts. This is subjected to an iterative adjustment process that gradually increases the agreement of the transition probabilities with the empirically observed distribution structure. The procedure follows a heuristic scheme: It generates artificial strings, compares their frequency distribution with the empirical data, and iteratively adjusts the transition probabilities.

The definition of a start symbol represents a model-theoretic simplification. It serves to generate syntactically consistent sequences and does not claim to fully capture the empirical diversity of real conversation openings.

4 Empirical Application: Eight Transcripts of Sales Conversations

4.1 Hypothetical Initial Grammar

Based on the literature on sales conversations, the following hypothetical grammar was derived: A sales conversation (VKG) consists of greeting (BG), sales part (VT), and farewell (AV). The terminal symbols include KBG, VBG, KBBd, VBBd, KBA, VBA, KAE, VAE, KAA, VAA, KAV, VAV.

4.2 The Eight Transcripts

The complete transcripts can be found in Appendix A. They document interactions at various sales stands at Aachen market square in June/July 1994.

4.3 Terminal Symbol Strings

Since sales conversations can empirically begin with different speech acts, a uniform start symbol was defined for the generation of artificial sequences. This decision serves exclusively model consistency and does not affect the transition structure of the grammar.

The terminal symbol strings formed from the transcripts are fully documented in Appendix A.

4.4 Python Implementation

The complete Python program for grammar induction and optimization can be found in Appendix B. It implements the steps described in Section 3 and visualizes the optimization process.

4.5 Results of Iterative Adjustment

The optimized grammar exhibits the following structure:

Table 3: Optimized Transition Probabilities

Start Symbol	**Following Symbols with Probabilities**
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 the validation phase, where a larger number of artificial sequences ($n = 100$) were generated based on the optimized transition structure, there is a near-perfect agreement between empirical and generated frequencies ($r = 0.9999$; $p < 0.001$).

This high agreement is not to be understood as predictive performance or proof of generalization. Rather, it documents the structural reproducibility of the empirically observed transition patterns using the same grammar with an enlarged sample. At the same time, it must be methodologically reflected that the Pearson correlation coefficient for frequency vectors with constant sum (1.0) tends to yield high values. The correlation observed here therefore primarily confirms the internal consistency of the procedure, less an external validity in the sense of predictive power (Flick, 2019, p. 489).

During the iterative optimization phase, the correlation remains stable at about r 0.92, which already indicates a high structural fit of the induced grammar. The further increase in correlation during validation is due to the larger sample of generated sequences with unchanged transition structure.

5 Discussion: ARS as a Contribution to Explainable Qualitative Research

5.1 ARS and the XAI Criteria

ARS fulfills the three NIST criteria for good explanations in a form adapted to qualitative research:

Meaningfulness is ensured through explicit category formation. The terminal symbols are semantically meaningful (KBG = customer greeting) and remain tied to the interpretive exploration. A third researcher can comprehend which assignments were made. This corresponds to the principle of "communicative validation" central to qualitative research (Flick, 2019, p. 328).

Accuracy is operationalized here in the sense of structural fit, not in the sense of predictive validity. The high agreement between empirical and generated frequencies shows that the grammar precisely reproduces the observed distribution structure of the data. In the terminology of qualitative research, one could speak of "appropriateness to the subject matter" (Przyborski & Wohlrab-Sahr, 2021, p. 34).

Knowledge Limits are marked by documenting the production and falsification of interpretations. The grammar does not claim to capture the "true" structure of the interaction but reconstructs observable regularities based on interpretive decisions. It thus makes its own contingency visible—a methodological virtue discussed in qualitative research under the keyword "reflexivity" (Flick, 2019, p. 129).

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

In XAI terminology, ARS is to be classified as an **ad-hoc procedure** (Explanation by Design). It does not design the grammar as a subsequent explanation of an already existing model but integrates explainability into the modeling process from the beginning. The terminal symbols are not black boxes but explicate the interpretive decisions. The transition probabilities are not opaque weights but simple relative frequencies.

This fundamentally distinguishes ARS from post-hoc procedures that attempt to subsequently explain the decisions of neural networks. While these procedures can only provide approximate insights into a principally opaque architecture, ARS is designed to be transparent from the ground up.

5.3 Limits of the Analogy

The analogy between XAI and qualitative research has limits that must be reflected upon. **First**, XAI primarily aims at explaining *technical* systems, while qualitative research is about the explication of *human* interpretation processes. The causality is different: In XAI, we explain why an algorithm made a particular decision; in ARS, we explain how researchers arrived at a particular interpretation.

Second, XAI operates with a different concept of truth. The explanations are supposed to correctly represent the actual decision processes of the model. In ARS, on the other hand, there are no "actual" processes that exist independently of interpretation. The grammar is not a discovery but a construction—one that must, however, prove itself against empirical evidence (Flick, 2019, p. 80).

Third, the addressee is different. XAI explanations are directed at users, developers, or regulatory authorities. ARS explanations are directed at the scientific community of qualitative research. The criteria for meaningfulness must therefore be adapted to their specific discourse practice.

5.4 Methodological Implications

Despite these limits, the XAI perspective opens up productive questions for qualitative research: How can we explicate our interpretation processes so that they become comprehensible to others? What formats of explication are suitable? How can we not only claim but demonstrate the quality of our interpretations?

ARS provides a concrete answer to these questions. It formalizes interpretation processes without automating them. It makes interpretive decisions explicit without eliminating hermeneutic openness. It thus creates the prerequisites for a methodologically reflected use of algorithmic procedures in qualitative research.

6 Conclusion and Outlook

Qualitative social research faces the challenge of using the possibilities of algorithmic text analysis without sacrificing its methodological standards. Algorithmic Recursive Sequence Analysis offers a way to productively address this challenge. It formalizes interpretation processes without automating them. It produces explicit, verifiable models without eliminating hermeneutic openness.

The connection to the XAI discussion proves doubly fruitful: It provides a conceptual framework to reflect on the quality of qualitative interpretations. And it reminds us that explainability is not a luxury but a necessity—in technology as well as in science.

Further research could develop ARS in several directions: through the integration of additional formal modeling methods (Petri nets, Bayesian networks), through more systematic connection with computational linguistics methods, or through application to other types of interaction. What remains crucial is always methodological control: The formal procedures must respect the interpretive character of the analysis and must not lead to its automation.

References

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). *An Introduction to Qualitative Research* (7th ed.). Sage Publications.

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 (Eds.), *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (pp. 193-210). Springer.

Oevermann, U., Allert, T., Konau, E., & Krambeck, J. (1979). The methodology of objective hermeneutics and its general research-logical significance in the social sciences. In H.-G. Soeffner (Ed.), *Interpretive Procedures in the Social and Text Sciences* (pp. 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 Social Research: A Workbook* (5th ed.). De Gruyter Oldenbourg. [German original: *Qualitative Sozialforschung: Ein Arbeitsbuch*]

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 (Eds.), *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (pp. 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 (Eds.), *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (pp. 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 (Eds.), *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (pp. 239-252). Springer.

A The Eight Transcripts with Terminal Symbols

A.1 Transcript 1 - Butcher Shop

Date: June 28, 1994, **Location:** Butcher Shop, Aachen, 11:00 AM

Table 4: Transcript 1 - Terminal Symbols

Transcript Excerpt	Terminal Symbol
Customer: Good day	KBG
Salesperson: Good day	VBG
Customer: One portion of coarse liver sausage, please.	KBBd
Salesperson: How much would you like?	VBBd
Customer: Two hundred grams.	KBA
Salesperson: Two hundred grams. Anything else?	VBA
Customer: Yes, then a piece of Black Forest ham.	KBBd
Salesperson: How large should the piece be?	VBBd
Customer: Around three hundred grams.	KBA
Salesperson: That will be eight marks twenty.	VAA
Customer: Here you are.	KAA
Salesperson: Thank you and have a nice day!	VAV
Customer: Thanks, you too!	KAV

Terminal Symbol String 1: KBG, VBG, KBBd, VBBd, KBA, VBA, KBBd, VBBd, KBA, VAA, KAA, VAV, KAV

A.2 Transcript 2 - Marketplace (Cherries)

Date: June 28, 1994, **Location:** Marketplace, Aachen

Table 5: Transcript 2 - Terminal Symbols

Transcript Excerpt	Terminal Symbol
Salesperson: Everyone can try cherries here, everyone can try cherries here!	VBG
Customer 1: Half a kilo of cherries, please.	KBBd
Salesperson: Half a kilo? Or a kilo?	VBBd
Salesperson: Three marks, please.	VAA
Customer 1: Thank you!	KAA
Salesperson: Everyone can try cherries here!	VBG
Customer 2: Half a kilo, please.	KBBd
Salesperson: Three marks, please.	VAA
Customer 2: Thank you!	KAA

Terminal Symbol String 2: VBG, KBBd, VBBd, VAA, KAA, VBG, KBBd, VAA, KAA

A.3 Transcript 3 - Fish Stand

Date: June 28, 1994, **Location:** Fish Stand, Marketplace, Aachen

Table 6: Transcript 3 - Terminal Symbols

Transcript Excerpt	Terminal Symbol
Customer: One pound of saithe, please.	KBBd
Salesperson: Saithe, alright.	VBBd
Salesperson: Four marks nineteen, please.	VAA
Customer: Thank you!	KAA

Terminal Symbol String 3: KBBd, VBBd, VAA, KAA

A.4 Transcript 4 - Vegetable Stand (Detailed)

Date: June 28, 1994, **Location:** Vegetable Stand, Marketplace, Aachen, 11:00 AM

Table 7: Transcript 4 - Terminal Symbols

Transcript Excerpt	Terminal Symbol
Customer: Listen, I'll take some mushrooms.	KBBd
Salesperson: Brown or white?	VBBd
Customer: Let's take the white ones.	KBA
Salesperson: They're both fresh, don't worry.	VBA
Customer: What about chanterelles?	KBBd
Salesperson: Ah, they're great!	VBA
Customer: Can I put them in rice salad?	KAE
Salesperson: Better sauté them briefly in a pan.	VAE
Customer: Okay, I'll do that.	KAA
Salesperson: Have a nice day!	VAV
Customer: You too!	KAV

Terminal Symbol String 4: KBBd, VBBd, KBA, VBA, KBBd, VBA, KAE, VAE, KAA, VAV, KAV

A.5 Transcript 5 - Vegetable Stand (with KAV at beginning)

Date: June 26, 1994, **Location:** Vegetable Stand, Marketplace, Aachen, 11:00 AM

Table 8: Transcript 5 - Terminal Symbols

Transcript Excerpt	Terminal Symbol
Customer 1: Goodbye!	KAV
Customer 2: I'd like a kilo of Granny Smith apples here.	KBBd
Salesperson: Anything else?	VBBd
Customer 2: Yes, another kilo of onions.	KBBd
Salesperson: Six marks twenty-five, please.	VAA
Customer 2: Goodbye!	KAV

Terminal Symbol String 5: KAV, KBBd, VBBd, KBBd, VAA, KAV

A.6 Transcript 6 - Cheese Stand

Date: June 28, 1994, **Location:** Cheese Stand, Marketplace, Aachen

Table 9: Transcript 6 - Terminal Symbols

Transcript Excerpt	Terminal Symbol
Customer 1: Good morning!	KBG
Salesperson: Good morning!	VBG
Customer 1: I'd like five hundred grams of Dutch Gouda.	KBBd
Salesperson: As a piece?	VBBd
Customer 1: Yes, as a piece, please.	KAA

Terminal Symbol String 6: KBG, VBG, KBBd, VBBd, KAA

A.7 Transcript 7 - Candy Stand

Date: June 28, 1994, **Location:** Candy Stand, Marketplace, Aachen, 11:30 AM

Table 10: Transcript 7 - Terminal Symbols

Transcript Excerpt	Terminal Symbol
Customer: I'd like one hundred grams of the mixed ones.	KBBd
Salesperson: For home or to take away?	VBBd
Customer: To take away, please.	KBA
Salesperson: Fifty pfennigs, please.	VAA
Customer: Thanks!	KAA

Terminal Symbol String 7: KBBd, VBBd, KBA, VAA, KAA

A.8 Transcript 8 - Bakery

Date: July 9, 1994, **Location:** Bakery, Aachen, 12:00 PM

Table 11: Transcript 8 - Terminal Symbols

Transcript Excerpt	Terminal Symbol
Customer: Good day!	KBG
Salesperson: One portion of our best coffee, freshly ground, please.	VBBd
Customer: Yes, also two pieces of fruit salad and a small cup of cream.	KBBd
Salesperson: Alright!	VBA
Salesperson: That will be fourteen marks and nineteen pfennigs, please.	VAA
Customer: I'll pay in small change.	KAA
Salesperson: Thank you very much, have a nice Sunday!	VAV
Customer: Thanks, you too!	KAV

Terminal Symbol String 8: KBG, VBBd, KBBd, VBA, VAA, KAA, VAV, KAV

B Complete Python Implementation

```python
"""
Algorithmic Recursive Sequence Analysis 2.0
Grammar Induction from Eight Transcripts
Optimization through Iterative Comparison of Empirical and Generated Strings
"""

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

# ============================================================================
# 1. EMPIRICAL DATA: Terminal symbol strings from eight transcripts
# ============================================================================

empirical_chains = [
    # Transcript 1: Butcher Shop
    ['KBG', 'VBG', 'KBBd', 'VBBd', 'KBA', 'VBA', 'KBBd', 'VBBd', 'KBA', 'VAA', 'KAA', 'VAV', 'KAV'],
    # Transcript 2: Marketplace (Cherries)
    ['VBG', 'KBBd', 'VBBd', 'VAA', 'KAA', 'VBG', 'KBBd', 'VAA', 'KAA'],
    # Transcript 3: Fish Stand
    ['KBBd', 'VBBd', 'VAA', 'KAA'],
    # Transcript 4: Vegetable Stand (detailed)
    ['KBBd', 'VBBd', 'KBA', 'VBA', 'KBBd', 'VBA', 'KAE', 'VAE', 'KAA', 'VAV', 'KAV'],
    # Transcript 5: Vegetable Stand (with KAV at beginning)
    ['KAV', 'KBBd', 'VBBd', 'KBBd', 'VAA', 'KAV'],
    # Transcript 6: Cheese Stand
    ['KBG', 'VBG', 'KBBd', 'VBBd', 'KAA'],
    # Transcript 7: Candy Stand
```

```python
    ['KBBd', 'VBBd', 'KBA', 'VAA', 'KAA'],
    # Transcript 8: Bakery
    ['KBG', 'VBBd', 'KBBd', 'VBA', 'VAA', 'KAA', 'VAV', 'KAV']
]

# ========================================================================
# 2. TRANSITION COUNTING AND INITIAL PROBABILITIES
# ========================================================================

def count_transitions(chains):
    """Counts transitions between terminal symbols in all chains"""
    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):
    """Normalizes transition counts to probabilities"""
    probabilities = {}
    for start in transitions:
        total = sum(transitions[start].values())
        probabilities[start] = {end: count / total
                                for end, count in transitions[start].items()}
    return probabilities

# Initial calculations
initial_transitions = count_transitions(empirical_chains)
```

```
initial_probabilities = calculate_probabilities(initial_transitions)

print("=" * 70)
print("ALGORITHMIC RECURSIVE SEQUENCE ANALYSIS 2.0")
print("=" * 70)
print("\n1. INITIAL TRANSITION PROBABILITIES (FROM EMPIRICAL DATA)")
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. TERMINAL SYMBOLS AND START SYMBOL
# ========================================================================

terminal_symbols = sorted(list(set([item for sublist in empirical_chains
                                     for item in sublist])))
start_symbol = empirical_chains[0][0]  # KBG as start (can be adjusted)

print(f"\nTerminal symbols ({len(terminal_symbols)}): {terminal_symbols}")
print(f"Start symbol: {start_symbol}")

# ========================================================================
# 4. GENERATION OF ARTIFICIAL CHAINS
```

```
# ==============================================================================

def generate_chain(probabilities, start_symbol, max_length=20):
    """Generates a chain based on transition probabilities"""
    chain = [start_symbol]
    current = start_symbol

    for _ in range(max_length - 1):
        if current not in probabilities:
            break

        next_symbols = list(probabilities[current].keys())
        probs = list(probabilities[current].values())

        # If no following symbols exist, break
        if not next_symbols:
            break

        next_symbol = np.random.choice(next_symbols, p=probs)
        chain.append(next_symbol)
        current = next_symbol

        # Stop if we land at a terminal without further transitions
        if current not in probabilities:
            break

    return chain

def generate_multiple_chains(probabilities, start_symbol, n_chains=8, max_length=20):
    """Generates multiple chains"""
    return [generate_chain(probabilities, start_symbol, max_length)
            for _ in range(n_chains)]

```

```python
# ========================================================================
# 5. FREQUENCY ANALYSIS
# ========================================================================

def compute_frequencies(chains, terminals):
    """Computes relative frequencies of terminal symbols in chains"""
    frequency_array = np.zeros(len(terminals))
    terminal_index = {term: i for i, term in enumerate(terminals)}

    for chain in chains:
        for symbol in chain:
            if symbol in terminal_index:
                frequency_array[terminal_index[symbol]] += 1

    total = frequency_array.sum()
    if total > 0:
        frequency_array /= total  # Normalization

    return frequency_array

# Empirical frequencies as reference
empirical_frequencies = compute_frequencies(empirical_chains, terminal_symbols)

print("\n2. EMPIRICAL RELATIVE FREQUENCIES")
print("-" * 70)
for i, symbol in enumerate(terminal_symbols):
    print(f"{symbol}: {empirical_frequencies[i]:.4f}")

# ========================================================================
# 6. ITERATIVE GRAMMAR OPTIMIZATION
```

```python
# ==========================================================================

def optimize_grammar(empirical_chains, terminal_symbols, start_symbol,
                     max_iterations=1000, tolerance=0.01, target_correlation=0.9):
    """
    Optimizes the grammar through iterative comparison with generated chains.
    """

    # Initial probabilities from empirical data
    transitions = count_transitions(empirical_chains)
    probabilities = calculate_probabilities(transitions)

    # Empirical frequencies as target
    empirical_freqs = compute_frequencies(empirical_chains, terminal_symbols)

    best_correlation = 0
    best_significance = 1
    best_probabilities = None
    history = []

    print("\n3. ITERATIVE OPTIMIZATION")
    print("-" * 70)

    for iteration in range(max_iterations):
        # Generate 8 artificial chains
        generated_chains = generate_multiple_chains(probabilities, start_symbol, n_chains=8)

        # Compute frequencies of generated chains
        generated_freqs = compute_frequencies(generated_chains, terminal_symbols)

        # Correlation analysis
```

```
        correlation, p_value = pearsonr(empirical_freqs, generated_freqs)
        history.append((iteration, correlation, p_value))

        # Progress display every 50 iterations
        if iteration % 50 == 0:
            print(f"Iteration {iteration:4d}: Correlation = {correlation:.4f}, p = {p_value:.4f}")

        # Check termination criterion
        if correlation >= target_correlation and p_value < 0.05:
            best_correlation = correlation
            best_significance = p_value
            best_probabilities = {start: probs.copy()
                                  for start, probs in probabilities.items()}
            print(f"\nOptimum reached at iteration {iteration}:")
            print(f"  Correlation = {correlation:.4f}")
            print(f"  Significance = {p_value:.4f}")
            break

        # Adjust probabilities
        for start in probabilities:
            for end in probabilities[start]:
                # Error calculation
                empirical_prob = empirical_freqs[terminal_symbols.index(end)]
                generated_prob = generated_freqs[terminal_symbols.index(end)]
                error = empirical_prob - generated_prob

                # Adjustment with tolerance factor
                probabilities[start][end] += error * tolerance

                # Bound to [0,1]
                probabilities[start][end] = max(0.01, min(0.99, probabilities[start][end]))
```

```python

        # Renormalization
        for start in probabilities:
            total = sum(probabilities[start].values())
            if total > 0:
                probabilities[start] = {end: prob / total
                                        for end, prob in probabilities[start].items()}

    # If no optimum was reached, take the best iteration
    if best_probabilities is None:
        # Find iteration with highest correlation
        best_idx = max(range(len(history)), key=lambda i: history[i][1])
        best_iter, best_correlation, best_significance = history[best_idx]
        best_probabilities = calculate_probabilities(count_transitions(empirical_chains))
        print(f"\nNo optimum reached. Best correlation at iteration {best_iter}:")
        print(f"  Correlation = {best_correlation:.4f}")
        print(f"  Significance = {best_significance:.4f}")

    return best_probabilities, best_correlation, best_significance, history

# Perform optimization
optimized_probabilities, best_corr, best_sig, history = optimize_grammar(
    empirical_chains, terminal_symbols, start_symbol,
    max_iterations=500, tolerance=0.005, target_correlation=0.9
)

# ===========================================================================
# 7. OPTIMIZATION VISUALIZATION
```

```
# ==========================================================================

def plot_optimization_history(history):
    """Visualizes the optimization process"""
    iterations, correlations, p_values = zip(*history)

    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

    # Correlation development
    ax1.plot(iterations, correlations, 'b-', linewidth=1.5)
    ax1.set_xlabel('Iteration')
    ax1.set_ylabel('Correlation (Pearson r)')
    ax1.set_title('Optimization Process: Correlation between
       Empirical and Generated Frequencies')
    ax1.grid(True, alpha=0.3)
    ax1.axhline(y=0.9, color='r', linestyle='--', alpha=0.5,
       label='Target correlation (0.9)')
    ax1.legend()

    # p-value development (logarithmic)
    p_values = [max(p, 1e-10) for p in p_values]  # Avoid log
       (0)
    ax2.semilogy(iterations, p_values, 'g-', linewidth=1.5)
    ax2.set_xlabel('Iteration')
    ax2.set_ylabel('p-value (logarithmic)')
    ax2.set_title('Significance of Correlation')
    ax2.grid(True, alpha=0.3)
    ax2.axhline(y=0.05, color='r', linestyle='--', alpha=0.5,
        label='Significance level (0.05)')
    ax2.legend()

    plt.tight_layout()
    plt.savefig('optimization_history.png', dpi=150)
    plt.show()

# Optional: Visualization (if matplotlib available)
try:
    plot_optimization_history(history)
```

```python
    print("\nOptimization history saved as 'optimization_history.png'.")
except:
    print("\n(Note: matplotlib required for visualization)")

# ==============================================================================
# 8. OUTPUT OF OPTIMIZED GRAMMAR
# ==============================================================================

print("\n" + "=" * 70)
print("4. OPTIMIZED PROBABILISTIC GRAMMAR")
print("=" * 70)

# Output sorted by start symbol
for start in sorted(optimized_probabilities.keys()):
    transitions = optimized_probabilities[start]
    transitions_str = ", ".join([f"'{end}': {prob:.3f}"
                                 for end, prob in sorted(transitions.items())])
    print(f"\n{start} -> {transitions_str}")

# ==============================================================================
# 9. VALIDATION: COMPARISON OF EMPIRICAL AND GENERATED FREQUENCIES
# ==============================================================================

# Generate new chains with optimized grammar
validation_chains = generate_multiple_chains(
    optimized_probabilities, start_symbol, n_chains=100, max_length=20
)
```

```python
validation_frequencies = compute_frequencies(validation_chains, terminal_symbols)

print("\n" + "=" * 70)
print("5. VALIDATION: EMPIRICAL VS. GENERATED FREQUENCIES")
print("=" * 70)

table_data = []
for i, symbol in enumerate(terminal_symbols):
    table_data.append([
        symbol,
        f"{empirical_frequencies[i]:.4f}",
        f"{validation_frequencies[i]:.4f}",
        f"{abs(empirical_frequencies[i] - validation_frequencies[i]):.4f}"
    ])

print(tabulate(table_data,
               headers=["Symbol", "Empirical", "Generated", "Difference"],
               tablefmt="grid"))

# Overall correlation
final_corr, final_p = pearsonr(empirical_frequencies, validation_frequencies)
print(f"\nCorrelation (100 generated chains): r = {final_corr:.4f}, p = {final_p:.4f}")

# ============================================================================
# 10. EXAMPLE GENERATED CHAINS
# ============================================================================

print("\n" + "=" * 70)
print("6. EXAMPLE GENERATED TERMINAL SYMBOL CHAINS")
print("=" * 70)

```

```python
example_chains = generate_multiple_chains(
    optimized_probabilities, start_symbol, n_chains=5,
       max_length=15
)

for i, chain in enumerate(example_chains, 1):
    chain_str = " -> ".join(chain)
    print(f"\nChain {i} ({len(chain)} symbols):")
    print(f"  {chain_str}")

#
   ======================================================================
# 11. EXPORT GRAMMAR AS STRUCTURE
#
   ======================================================================

def export_grammar_as_pcfg(probabilities, filename="
   optimized_grammar.txt"):
    """Exports the grammar in PCFG format"""
    with open(filename, 'w', encoding='utf-8') as f:
        f.write("# Optimized probabilistic context-free
           grammar (PCFG)\n")
        f.write("# Generated by Algorithmic Recursive
           Sequence Analysis 2.0\n\n")

        for start in sorted(probabilities.keys()):
            transitions = probabilities[start]
            for end, prob in sorted(transitions.items()):
                f.write(f"{start} -> {end} [{prob:.3f}]\n")

    print(f"\nGrammar exported as '{filename}'.")

export_grammar_as_pcfg(optimized_probabilities)

print("\n" + "=" * 70)
print("ALGORITHMIC RECURSIVE SEQUENCE ANALYSIS COMPLETED")
print("=" * 70)
```

Listing 2: Algorithmic Recursive Sequence Analysis 2.0 - Complete Code