INFORMATION MODELS FOR PREDICTION

PROJECT 1 - ALGORITHMIC THEORY OF INFORMATION COURSE

Alexandre Ribeiro (108122)
Maria Sardinha (108756)
Miguel Pinto (107449)

Index

- Introduction
- Key Programs Developed
- Workflow Diagram
- Model Implementation
- Implementation Steps
- Standard FCM vs Recursive FCM
- Model Export/Import
- Editor and Auxiliary Features
- Generated Text Samples
- Experimental Results
- Main Findings
- Future Work
- Live Demo + Questions

Introduction

- Implemented Finite-Context Models (FCMs) for text analysis and predictive text generation.

- **Aim:** Estimate symbol distributions in text for statistical analysis and predictive text generation.

- Developed two main C++ programs:
  - **fcm** (statistical analysis)
  - **generator** (text generation)

- Evaluated models performance with varying parameters (context order *k*, smoothing *alpha*).

- Implemented and compared JSON/binary model export formats.

Key Programs Developed

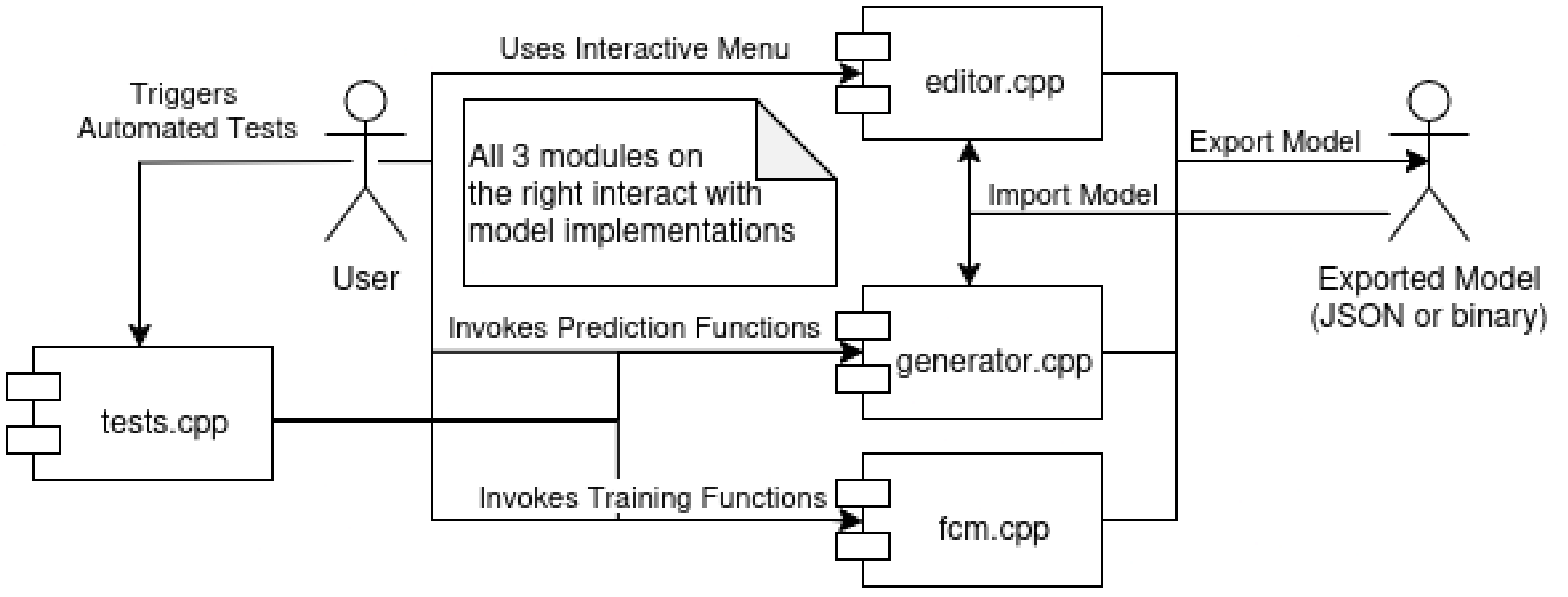
fcm:
Reads input text, constructs FCM by updating frequency tables, computes probabilities with Laplace smoothing, calculates Average Information Count.

generator:
Generates new text based on a trained model using weighted random selection and can also do all operations **fcm** does.

editor:
User-friendly interactive text-based interface for model manipulation.

tests:
Automated testing, with multiple combination of parameters, for models validation.

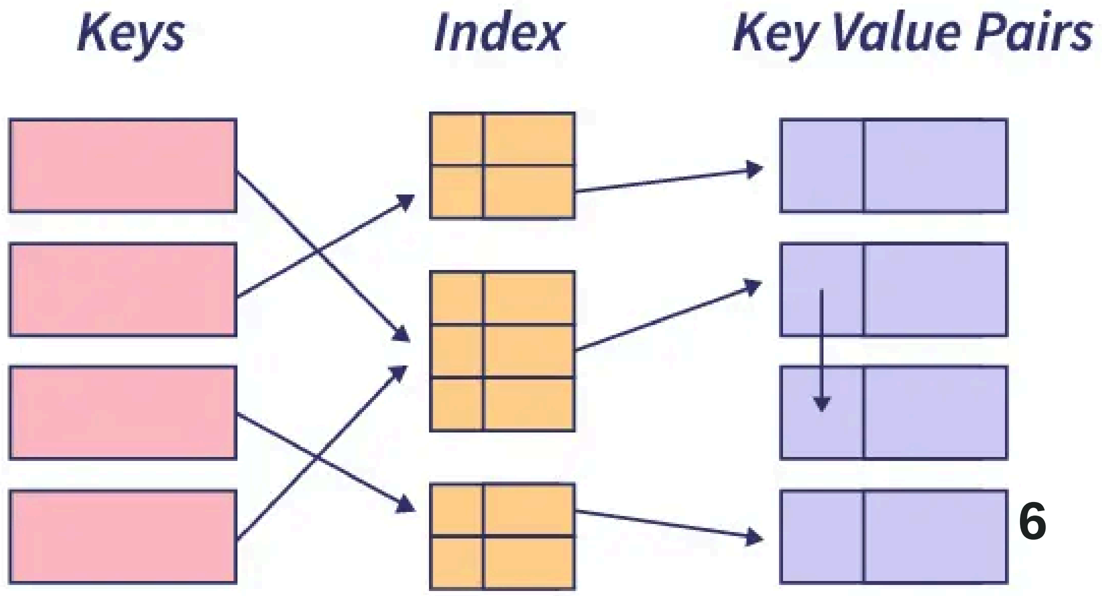
Workflow Diagram



Model Implementation

Data Structures

- **Hash Tables** (*unordered_map*) for efficient O(1) lookups.
- **Frequency Table:** counts occurrences of symbols after contexts.
- **Context Count Table:** tracks context occurrences.
- **Probability Table:** stores computed probabilities.
- **Alphabet Set:** stores unique symbols.



Implementation Steps

Learning Phase

- UTF-8 character splitting.
- Sliding window processing.
- Dynamic alphabet update.
- Context frequency update.
- Frequency table update.

Average Information Content Calculation

- Quantifies the model quality (lower == better)
- Measured in bits per symbol (bps)

Text Generation

- Context-based sampling.
- Weighted random selection via cumulative distribution, probabilities calculated with Laplace smoothing.
- Rolling context updates.

Standard FCM vs Recursive FCM

Standard FCM

Uses fixed context length k

Recursive FCM

Incorporates multiple context lengths (*k* down to 1).

RFCM features:

- Recursive learning for all context lengths.
- Separate frequency tables for each context length.
- Recursive probability retrieval with fallback mechanism.

{JSON}

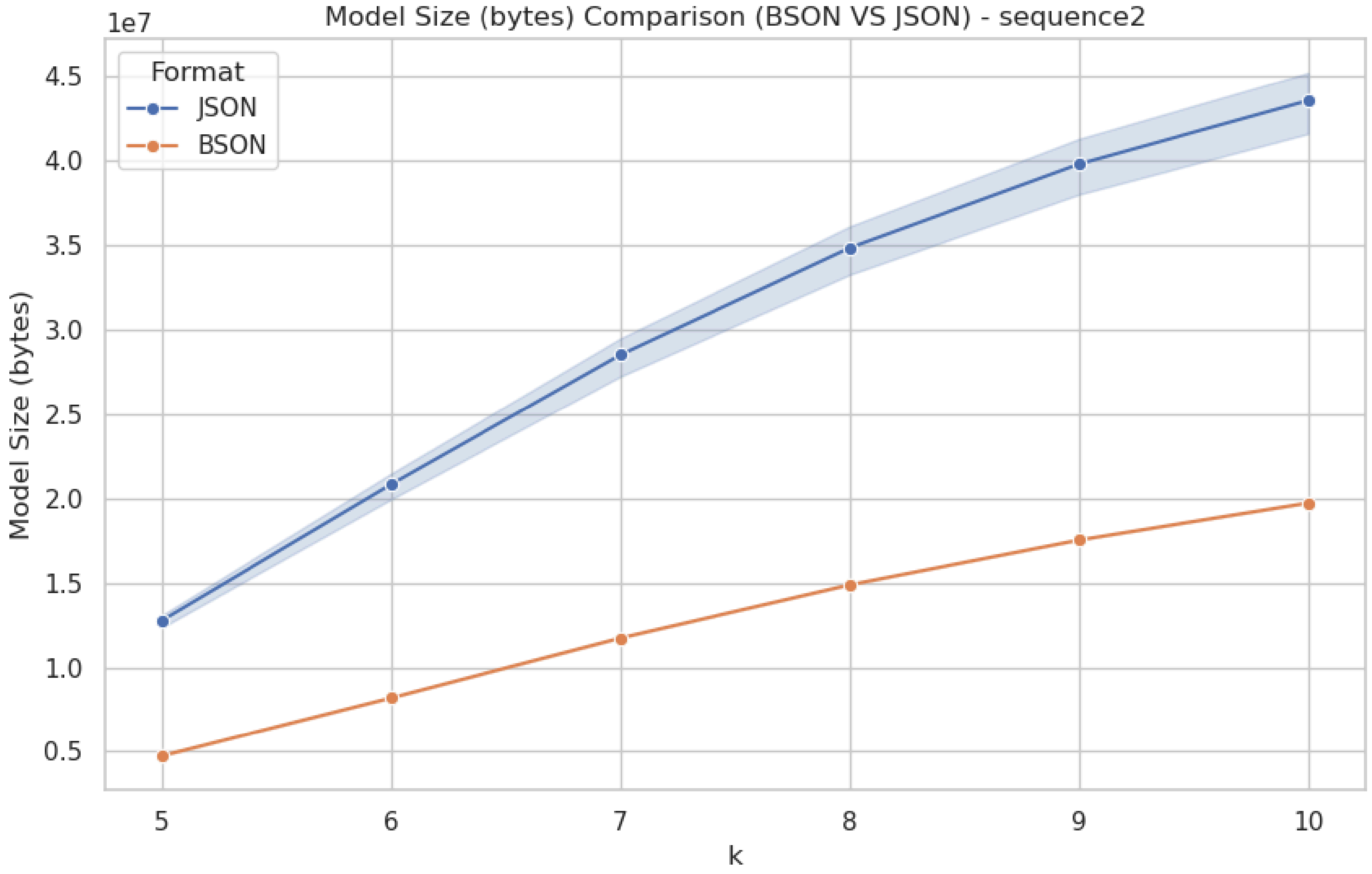
VS

{ 01010100
11101011
10101110
01010101 }
BSON

Model Export/Import

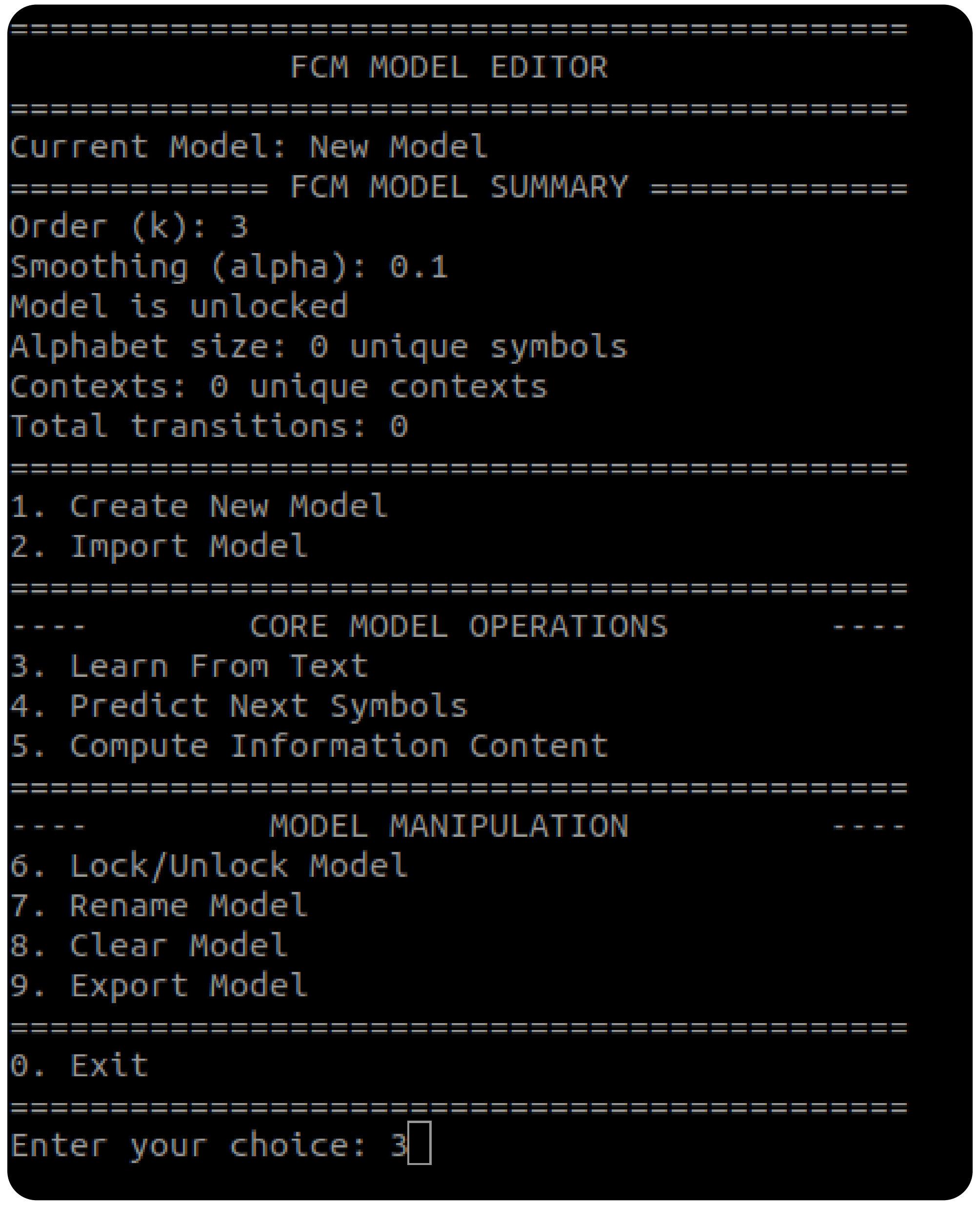
Formats

- **JSON:**
  - Human-readable.
  - Larger file size.
- **Binary (BSON):**
  - Compact & Efficient.
  - Non-human readable.

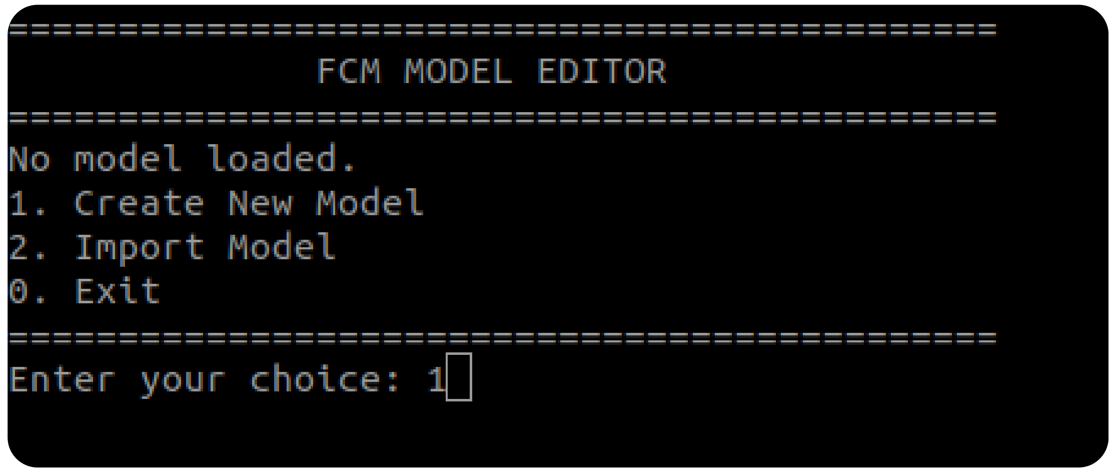


Editor and Auxiliary Features

- Interactive menu for model creation and manipulation.
- Models properties display.
- Continuous refinement.
- Syntactic analysis via word list comparison.



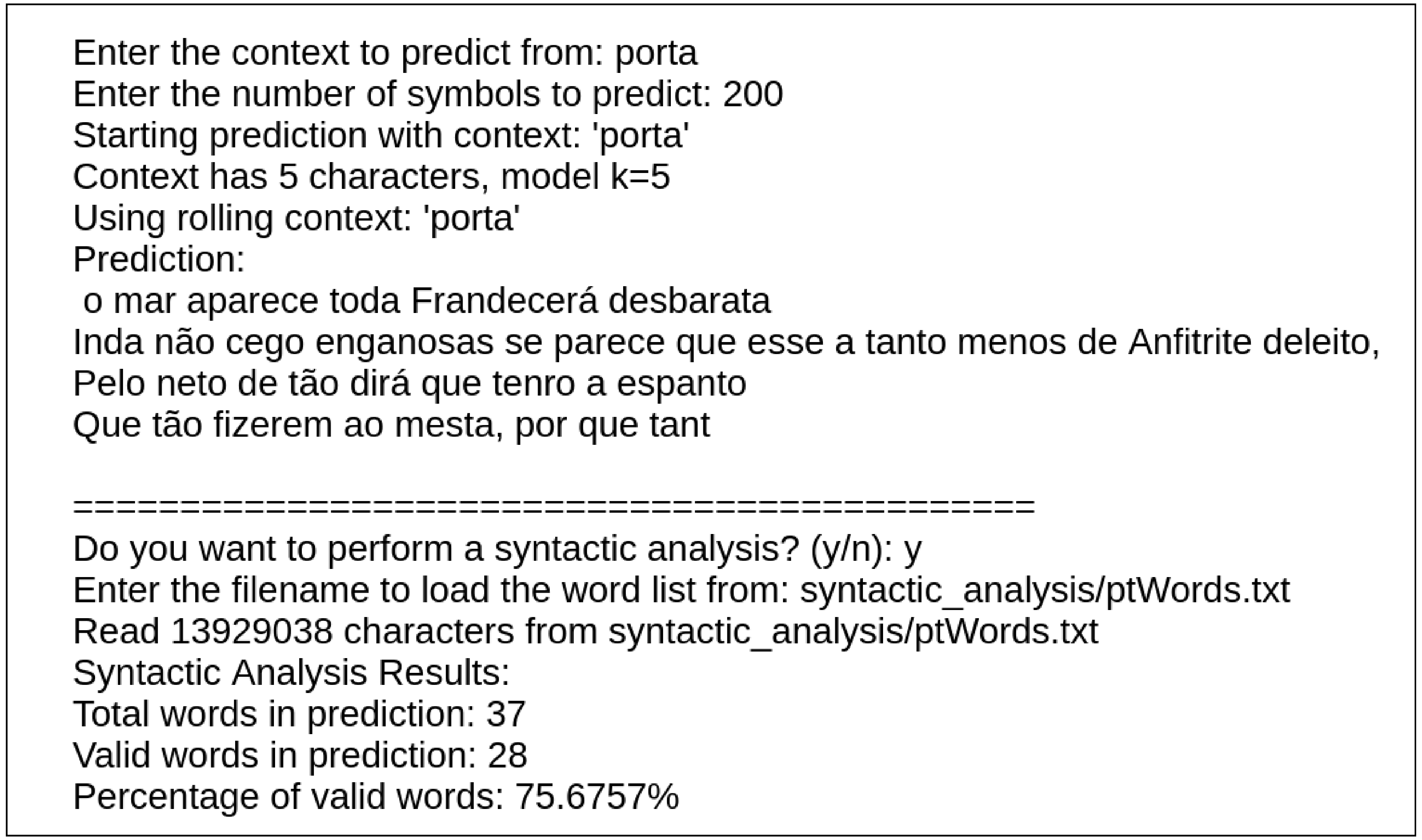
```
=============================================
              FCM MODEL EDITOR
=============================================
Current Model: New Model
============= FCM MODEL SUMMARY =============
Order (k): 3
Smoothing (alpha): 0.1
Model is unlocked
Alphabet size: 0 unique symbols
Contexts: 0 unique contexts
Total transitions: 0
=============================================
1. Create New Model
2. Import Model
=============================================
----        CORE MODEL OPERATIONS        ----
3. Learn From Text
4. Predict Next Symbols
5. Compute Information Content
=============================================
----         MODEL MANIPULATION          ----
6. Lock/Unlock Model
7. Rename Model
8. Clear Model
9. Export Model
=============================================
0. Exit
=============================================
Enter your choice: 3
```



```
=============================================
              FCM MODEL EDITOR
=============================================
No model loaded.
1. Create New Model
2. Import Model
0. Exit
=============================================
Enter your choice: 1
```

Generated Text Samples

- Example outputs with different parameters.
- Syntactic validation evaluation (in this example, ~75% of the words were recognized).

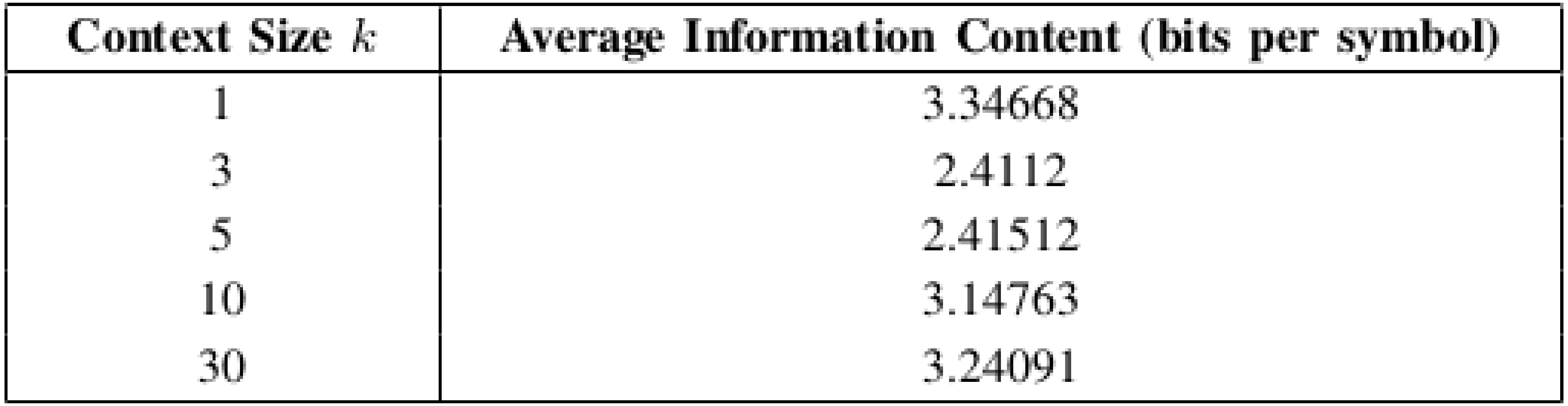


```
Enter the context to predict from: porta
Enter the number of symbols to predict: 200
Starting prediction with context: 'porta'
Context has 5 characters, model k=5
Using rolling context: 'porta'
Prediction:
 o mar aparece toda Frandecerá desbarata
Inda não cego enganosas se parece que esse a tanto menos de Anfitrite deleito,
Pelo neto de tão dirá que tenro a espanto
Que tão fizerem ao mesta, por que tant

=============================================
Do you want to perform a syntactic analysis? (y/n): y
Enter the filename to load the word list from: syntactic_analysis/ptWords.txt
Read 13929038 characters from syntactic_analysis/ptWords.txt
Syntactic Analysis Results:
Total words in prediction: 37
Valid words in prediction: 28
Percentage of valid words: 75.6757%
```

Experimental Results

Effect of Context Size



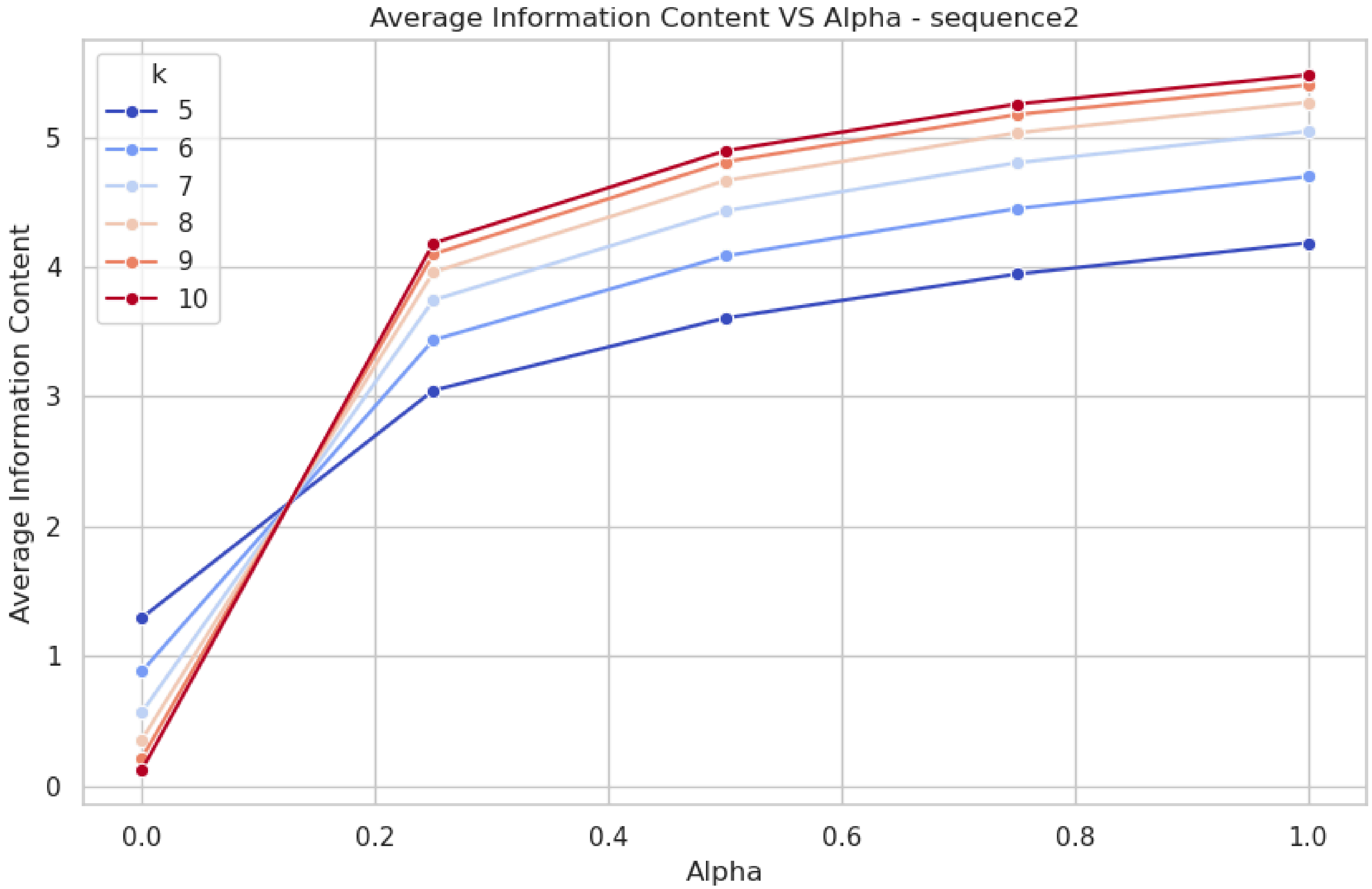
Context Size k	Average Information Content (bits per symbol)
1	3.34668
3	2.4112
5	2.41512
10	3.14763
30	3.24091

Average Information Content, based on *k* - sequence2.txt

Experimental Results

Impact of Smoothing Parameter

- Moderate *alpha* values generally produce more reliable models.
- Need to balance over-smoothing and under-smoothing.

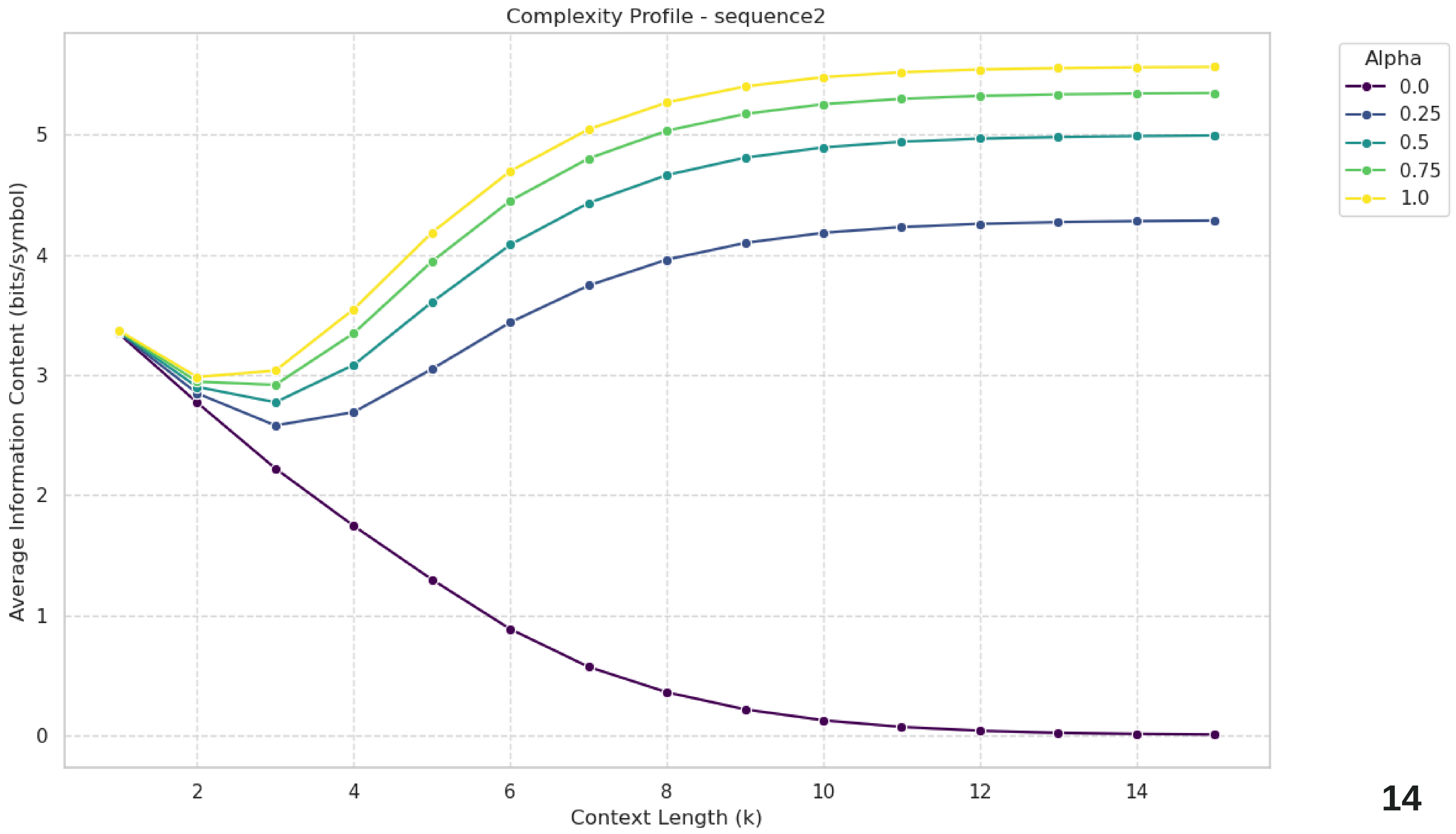


Average Information Content, based on alpha - sequence2.txt

Experimental Results

Complexity Profiles

- Lower-order contexts contribute significantly to prediction accuracy.
- Sharp decreases at low *alpha* values indicate highly structured sequences.
- Higher *alpha* values show more randomness and long-range dependencies



Complexity Profile - sequence2.txt

Main Findings

- Optimal context size (*k*) between 3-5 for tested datasets.
- Moderate smoothing parameters (~0.1) produce more reliable models.
- Binary serialization reduced storage by 40-60% compared to JSON.
- Generated text exhibited recognizable stylistic elements.
- Syntactic analysis showed moderate-high rate of valid words.

Future Work

- Adaptive context selection based on multiple tables (RFCM).
- Interactive text generation with real-time suggestions.
- Improved dictionary-based correction mechanisms.

DEMO

+

QUESTIONS