Implementation of an Anti-Plagiarism Constraint Model for Sequence Generation Systems

Janita Aamir, Paul Bodily
Computer Science Department
Idaho State University
Pocatello, ID 83209 USA
aamijani@isu.edu, bodipaul@isu.edu

Abstract

Sequence generation models are heavily used in computational creative systems in natural language, music composition, and other creative domains. One of the biggest challenges that come with sequence generation models is that, because they learn from existing resources, products from these models often exhibit varying degrees of plagiarism. Papadopoulos, Roy, and Pachet (2014) have, in previous work, presented a max-order Markov automaton to avoid plagiarism in generative sequence models. However, the original publication presented only the algorithmic pseudocode without providing a working implementation. In this replication study, we present a working implementation of the max-order Markov automaton designed to be integrated into sequence generation models for avoiding plagiarism. We use our working implementation to generate new results that verify the efficacy of this approach to avoiding plagiarism. We illustrate how the max-order Markov automaton can be integrated effectively to avoid plagiarism in CC systems using a lyrical music composition system, *Pop**, as an example. Source code:
`https://github.com/aamijani/Anti-Plagiarism_Constraint_Model`

Introduction

Research into the development of generative sequence models have been foundation to much of the advancements in computational creativity (CC) across the domains of music and natural language. As these computationally creative systems gain more attention from the wider population, it becomes crucial for these systems to be aware of and avoid plagiarism (i.e., generation of subsequences longer than some pre-specified length that are copied verbatim from a training corpus). Most of the computationally creative models learn from existing resources to produce new outputs. It is therefore not uncommon for these models to occasionally exhibit plagiarism. Besides the obvious negative impacts that plagiarism can have on the novelty of generative CC systems, the problem of plagiarism also raises an ethical dilemma. The problem of plagiarism in sequence generation models is an imminent problem that must be addressed if CC is to broaden its appeal and relevance beyond being merely an academic pursuit.

Our interest in this study is to replicate the results of *Avoiding Plagiarism in Markov Sequences Generation* (Papadopoulos, Roy, and Pachet 2014). The approach presented in this paper is an effective way to avoid plagiarism. To our knowledge, no one, including the original author, has published an open-source implementation of the model that is available for use. The implementation we present here has been made publicly available. It is implemented using generic type variables allowing for new types to be specified later without need to modify the original codebase. This facilitates integration with Markov generative systems in a variety of domains. A strength of Markov generative systems is that, when combined with constraints (e.g., anti-plagiarism constraints), they are capable of guaranteeing the strict enforcement of those constraints.

Much state-of-the-art sequence generation is currently done both in and out of CC with transformer and LSTM models. For example, ChordAL (Tan 2019) is a system built using Bi-LSTMs that composes melodies. DeepJ (Mao, Shin, and Cottrell 2018) is a generative model that uses LSTMs and is capable of composing music conditioned on a specific mixture of composer styles. GLACNet (Kim et al. 2018) generates visual stories by making use of bi-directional LSTMs. These models have been found to be particularly difficult to constrain. One of the more successful attempts has the Anticipation-RNN model (Hadjeres, Pachet, and Nielsen 2017). However, even this model allows a percentage of generated sequences that do not satisfy constraints and thus still does not make guarantees (Hadjeres and Nielsen 2020).

There have been several Markov generation systems presented in the CC field. For example, *Pop** (Bodily and Ventura 2022) is a music generation Markov model that uses Twitter as an inspiration to produce music. *SMUG* (Scirea et al. 2015) is a system which utilizes Markov chains and works by using academic papers as an inspiration to compose lyrics and melodies. *EMILY* (Shihadeh and Ackerman 2020) is a system that aims to create original poems in the style of renowned poet Emily Dickinson. It makes use of Markov Chains to produce these poems. *LyricJam* (Vechtomova, Sahu, and Kumar 2021) is another generative system that uses live instrumental music to generate lyrics. In order for these and other systems to gain traction beyond merely academic exercises, they need to avoid plagiarism. The suc-

cess of these and other generative systems depends on their ability to avoid plagiarism.

The study done by Papadopoulos, Roy, and Pachet is important because of how many systems there are that produce music. Moreover, our published model is generalized and is able to not only avoid plagiarism in music generation systems, but also other systems like short-story writing, slogans, etc. In the paper, (2014) introduce a max-order Markov automaton in the framework of constraints satisfaction (CSP). This automaton ensures that sequences generated by a Markov model do not contain subsequences longer than a specified maximum order. Besides its use for avoiding plagiarism, this model can also be used to *detect* plagiarism in existing artifacts (e.g., rhythms, lyrics, etc.).

Replication

The approach outlined by Papadopoulos, Roy, and Pachet (2014) is broken into two algorithms which we refer to as Algorithms 1 and 2. We give a high-level overview of these algorithms below. Our publicly available implementation of these two algorithms can be readily applied to generate sequences of natural language, music, rhythms, etc. In the following sections, we illustrate results of our working implementation in two natural language domains.

Automaton

The base model underlying both a Markov and a max-order Markov model is the finite automaton. A finite automaton $A = \{Q, \Sigma, \delta, q_0, F\}$ is a 5-tuple with elements defined as follows:

- Q is a finite non-empty set of states;
- Σ, the alphabet, is a finite non-empty set of symbols;
- $q_0 \in Q$ is the initial state of the automaton;
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function which maps a state to its successors for a given symbol;
- $F \subseteq Q$ is the set of final or accepting states.

Markov Automaton (Algorithm 1)

The Markov Property states that only the present state (independent of how this state was reached) is the determining factor for the probability of future states. The output of the Markov automaton algorithm is an automaton that recognizes all valid Markovian sequences; i.e., sequences where any two successive N-grams correspond to a $(N+1)$-gram of the training corpus (for a Markov order of N) (Papadopoulos, Roy, and Pachet 2014). A Markov automaton maintains the property that for each $a \in \Sigma$ there exists a unique $q_a \in Q$ and all transitions in δ transitioning via a map to the state q_a.

Fig. 1 shows the 1st-order Markov automaton constructed using 'KALIKIMAKA' as its input dataset. The strength of this (intermediate) model is that it accepts valid Markov strings such as 'MALI' and 'LIMA'. The weakeness of this model is that it also accepts the full original string 'KALIKIMAKA'. For the purposes of eliminating plagiarism, we need to modify the automaton to disallow substrings above a defined length that, albeit valid Markov strings, are also exact substrings of the training set. Thus our Markov automaton is the input to our second algorithm.

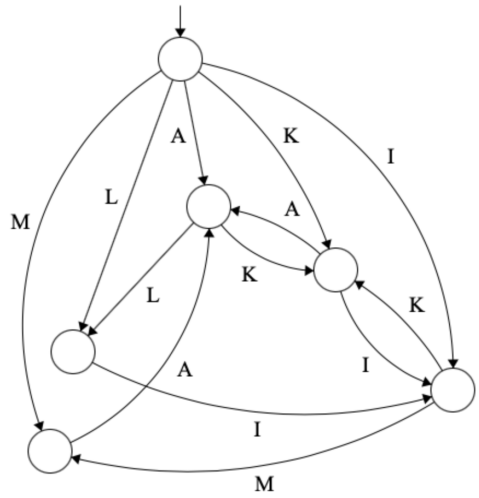

Figure 1: *A Markov automaton for letters*. This automaton accepts all valid Markov strings that can be generated from a 1st-order Markov model trained on 'KALIKIMAKA'. All nodes are accept nodes.

Max-Order Markov Automaton (Algorithm 2)

Algorithm 2 modifies the Markov automaton to remove from the set of accepted strings any sequence containing a 'no-good' subsequence, i.e., a sequence above some length L that appears verbatim in the corpus. This is accomplished by first creating a trie of all no-goods in which all states but the ones corresponding to a full no-good are accept states. This guarantees that a no-good cannot be accepted by the model. Next edges are added for overlapping prefixes. For example, if *ABCD* and *BCEF* are no-goods, then the prefixes *ABC* and *BCEF* share an overlapping prefix (i.e., *BC*). Adding edges for overlapping prefixes ensures that the automaton will not only reject *ABCD* and *BCEF* but also that is will reject *ABCEF*, as well. Algorithm 2 uses an adaptation of the Aho and Corasick (1975) string-matching algorithm to form these cross-prefix transitions.

Fig. 2 shows the resulting max-order Markov automaton derived from the Markov automaton in Fig. 1 with $L = 4$.

Easy reuse in new domains

Our implementation of the max-order Markov automaton uses generics to allow anti-plagiarism constraints to be readily applied to sequence generation models in any domain. Whereas our previous examples demonstrated the construction a max-order Markov automaton for constraining sequences of *letters*, we demonstrate here the application of our implemented model to constrain sequences of *words*. Fig. 3 shows the Markov automaton derived from the training sequence 'can you can a can as a canner can can a can'. A expected, the model accepts valid Markov strings such as

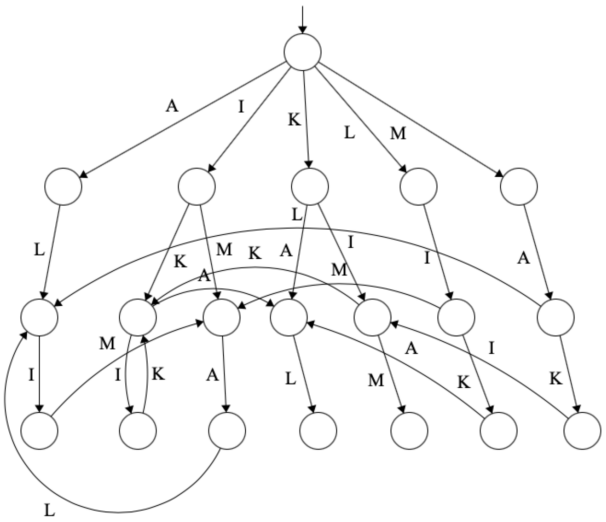

Figure 2: *A max-order Markov automaton for letters*. This automaton accepts the same set of strings as the automaton in Fig. 1 minus strings of length ≥ 4 that contain exact substrings of (i.e., plagiarize) the training sequence 'KALIKIMAKA'. All nodes are accept nodes.

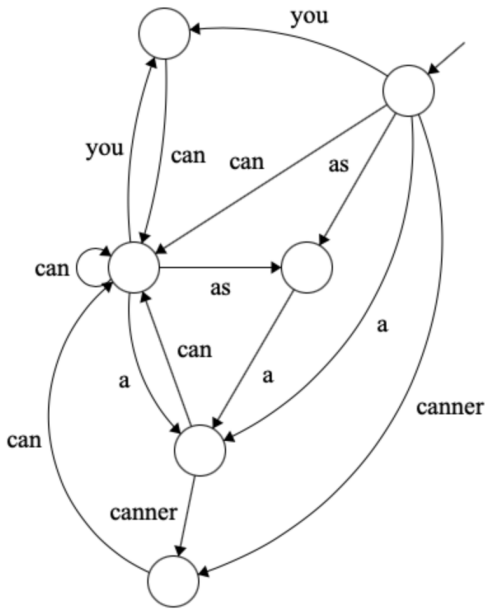

Figure 3: *A Markov automaton for words*. This automaton accepts all valid Markov sequences that can be generated from a 1st-order Markov model trained on 'can you can a can as a canner can can a can'. All nodes are accept nodes.

'you can can a canner' and 'can a canner can you' as well as the full original sequence.

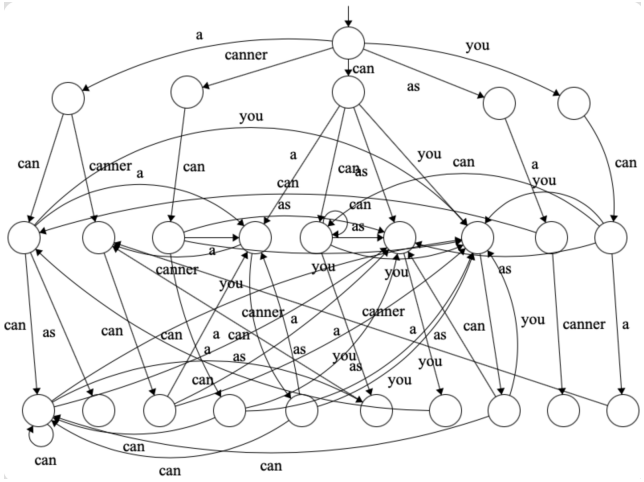

Figure 4: *A max-order Markov automaton for words*. This automaton accepts the same set of sentences as the automaton in Fig. 3 minus sentences of ≥ 4 words that contain exact phrases from (i.e., plagiarize) the training sequence 'can you can a can as a canner can can a can'. All nodes are accept nodes.

Integrated Visualization Feature

Our implementation of the algorithms for constructing max-order Markov automata includes a feature allowing graphs of the finite automata to be visualized at each intermediate algorithmic step and/or in their final form. This enables users to better see and understand the process of how the automata are built and to verify the model's results. The feature saves graphs in .dot format.

Applications in CC Systems

Our primary motivation for being able to generate max-order Markov automata is to incorporate anti-plagiarism constraints into *Pop**, a CC lyrical music composition system built using constrained Markov models (Bodily and Ventura 2022). The model uses Markov models to generate interdependent harmonic, melodic, rhythmic, and lyrical sequences. Like several other generative models (cf. (Fargier and Vilarem 2004; Papadopoulos et al. 2015)), *Pop** defines and integrates constraints in the form of finite automata. For example, Fig. 5) illustrates a finite automaton constructed to enforce a rhyming constraint between the first and fourth words in a four-word lyrical sequence. Automata such as these are then compiled with Markov models to probabilistically generate sequences that adhere to constraints.

Computational theory informs us that regular languages are closed under intersection, and indeed algorithms have been presented that, given two automata A and B, create a third automata C, such that the set of sequences accepted by C is the intersection of the sets accepted by A and B (Sipser 1996). By combining max-order Markov automata with the automata already in use to constrain the generation of *Pop**, we immediately inherit the ability to constrain our compositions against plagiarism—across all aspects of the

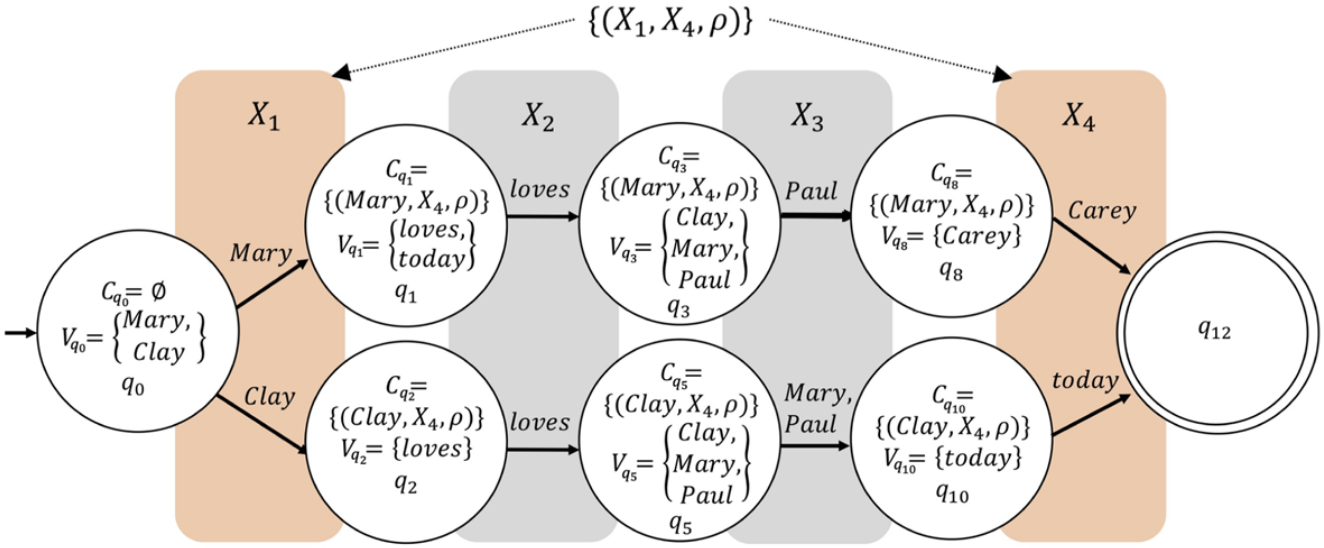

Figure 5: Shown is an automaton designed to enforce a rhyming constraint, ρ, between the first and last positions, X_1 and X_4, in the Markov generation of a four-word sentence in the CC music composition system *Pop**. Generative systems like *Pop** that define constraints using automata are particularly well-suited for easy integration of max-order Markov automata for constraining against plagiarism. Figure originally from (Bodily and Ventura 2022).

composition.

Conclusion

We have presented an implementation of an anti-plagiarism model first presented by Papadopoulos, Roy, and Pachet (2014). The model works by utilizing a max-order Markov automaton that only accepts non-plagiaristic sequences based on a specified corpus. We illustrated through examples how, through the use of generics, this model can be applied with constrained sequence generation in novel CC domains, and highlighted, in particular, its envisioned integration into a lyrical music composition system. Whether the goal be to achieve greater novelty or to show increased respect to the ethics of avoiding plagiarism, the implemented model we have presented will serve to aid CC practitioners to achieve greater and more ambitious milestones in the pursuit of computational creativity.

Author Contributions

Both authors contributed to all aspects of the work, including ideation, narrative/position development and writing.

Acknowledgments

The authors have no acknowledgments.

References

Aho, A. V., and Corasick, M. J. 1975. Efficient string matching: an aid to bibliographic search. *Communications of the ACM* 18(6):333–340.

Bodily, P., and Ventura, D. 2022. Steerable music generation which satisfies long-range dependency constraints. *Transactions of the International Society for Music Information Retrieval* 5(1).

Fargier, H., and Vilarem, M.-C. 2004. Compiling csps into tree-driven automata for interactive solving. *Constraints* 9(4):263–287.

Hadjeres, G., and Nielsen, F. 2020. Anticipation-rnn: Enforcing unary constraints in sequence generation, with application to interactive music generation. *Neural Computing and Applications* 32(4):995–1005.

Hadjeres, G.; Pachet, F.; and Nielsen, F. 2017. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning*, 1362–1371. PMLR.

Kim, T.; Heo, M.; Son, S.; Park, K.; and Zhang, B. 2018. GLAC net: Glocal attention cascading networks for multi-image cued story generation. *CoRR* abs/1805.10973.

Mao, H. H.; Shin, T.; and Cottrell, G. 2018. Deepj: Style-specific music generation. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, 377–382.

Papadopoulos, A.; Pachet, F.; Roy, P.; and Sakellariou, J. 2015. Exact sampling for regular and markov constraints with belief propagation. In *International Conference on Principles and Practice of Constraint Programming*, 341–350. Springer.

Papadopoulos, A.; Roy, P.; and Pachet, F. 2014. Avoiding plagiarism in markov sequence generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Scirea, M.; Barros, G. A.; Shaker, N.; and Togelius, J. 2015. Smug: Scientific music generator. In *ICCC*, 204–211.

Shihadeh, J., and Ackerman, M. 2020. Emily: An emily dickinson machine. In *ICCC*, 243–246.

Sipser, M. 1996. Introduction to the theory of computation. *ACM Sigact News* 27(1):27–29.

Tan, H. H. 2019. Chordal: A chord-based approach for music generation using bi-lstms. In *ICCC*, 364–365.

Vechtomova, O.; Sahu, G.; and Kumar, D. 2021. Lyricjam: A system for generating lyrics for live instrumental music. *arXiv preprint arXiv:2106.01960*.