“She Offered No Argument”: Constrained Probabilistic Modeling for Mnemonic Device Generation

Paul M. Bodily, Porter Glines, and **Brandon Biggs**
Department of Computer Science
Idaho State University
921 S. 8th Ave, Pocatello, ID 83209 USA
bodipaul@isu.edu, glinport@isu.edu, biggbran@isu.edu

Abstract

A common aspect to creativity as described by creative theorists is the juxtaposition and balance of two opposing qualities, namely novelty and typicality. Practical models of computational creativity are needed that effectively leverage the contributions of each of these qualities in a synchronous manner. We discuss the effectiveness of constrained probabilistic models in representing this duality in generative models of creativity. We illustrate constrained Markov models as an example of a constrained probabilistic model and demonstrate its application to computational creativity in the elaboration of a system called NhMMonic for generating mnemonic devices. We demonstrate the effectiveness of the system[1] using a qualitative survey. Our findings suggest that the constrained Markov model is particularly effective at generating mnemonics that exhibit novelty and typicality in grammatical and semantic flow with the overall result of more effective mnemonics for the purpose of memorization. Source code as well as our mnemonic device generator are both freely accessible online.

Introduction

Computational creativity (CC) has been defined as “the philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative” (Colton and Wiggins 2012). The plural focus on the philosophy, science and engineering of computational systems has yielded valuable theoretical contributions as well as a number of functional creative systems. Emergent from this plural focus is the challenge of maintaining harmony between theory and practice. To be sure the abstract philosophy and concrete engineering can and should work to challenge one another in their mutual growth and evolution; however, the goal ultimately is to develop systems that accurately reflect the philosophical moorings and to advance theories whose tenets agree with what is observed about creativity in practice. Thus the role of *practical models* of creativity becomes significant—models that, by virtue of their ability to implement principles deriving from the philosophy, can be generalized beyond any single creative system with great effect, while maintaining ready applicability and implementability. As described by Jordanous (2016), these models define the *creative process* of a system, namely “what the creative individual does to be creative.”

Several examples of practical models of creativity have been demonstrated. Evolutionary models represent a practical implementation of the widely-accepted theory that creativity is a self-evaluative, iterative process as discussed by Csíkszentmihályi (1996) (e.g., see Morris et al. (2012)). Related is the model of a dynamic knowledge base (Pérez y Pérez and Sharples 2004) in which novel artefacts that have been evaluated as belonging to the domain are added to a system's set of exemplars, possibly altering the definition of the domain itself (e.g., as discussed by Boden (2003)). Generate-and-check is another model that has been suggested as being representative of the creative process (Pease, Guhe, and Smaill 2010).

In considering the modeling of theoretical aspects of creativity, one particularly intriguing aspect that is often discussed is the tenuous balance that a creative system must maintain between novelty and typicality—the adherence to structural domain-defining rules combined with an exploratory discovery of new, valuable artefacts. These two characteristics can sometimes seem at odds with one another; a creative system must both obey norms at some level and break them entirely at other levels. It is the juxtaposition of these qualities that evokes the perception of creativity: the observer recognizes and appreciates an artefact relative to its contextual domain while at the same time being challenged and surprised as a result of the artefact's unique traits and value. Csíkszentmihályi (1996) emphasizes that creativity stems from a person learning the rules of and basic procedures of a domain and then channeling thinking based on those rules in new directions. Saunders and Gero (2001) puts novelty and typicality on a spectrum called the Wundt curve or “hedonic function” and frames successful creativity in terms of finding the correct balance of typicality and novelty (see Figure 1). Margaret Boden (2003), in her seminal work *The Creative Mind: Myths and Mechanisms*, compares (exploratory) creativity to navigating a “structured conceptual space” to find “things you'd never noticed before.” Wiggins (2006) elaborates a formal mechanism of Boden's concept of creativity by defining two rule sets, $\mathscr{R}$ and $\mathscr{T}$. Of

[1] An interactive demo can be viewed at `https://ccil.cs.isu.edu/projects/mnemonic/`

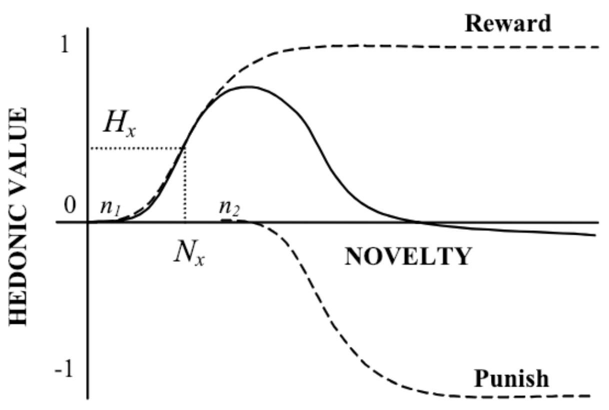


Figure 1: The Wundt curve models value as the sum of two nonlinear functions: H_x which rewards novelty, and N_x which punishes novelty beyond some threshold of typicality, from Saunders and Gero (2001).

these two sets $\mathscr{R}$ is a set of rules which "constrain the space" to a representation of "the agreed nature of what the artefact is, in the abstract"; $\mathscr{T}$, by contrast, is a set of traversal rules which, when constructed effectively, is designed to find concepts that have not been previously discovered. Ritchie (2007), in defining empirical criteria for attributing creativity to a computer program, defines three essential properties, two of which are novelty and typicality (the third is quality, which Boden also emphasizes and which we will discuss below).

Many existing abstract frameworks for building creative systems have been described, several of which explicitly model the components of novelty and typicality (e.g., (Ventura 2017)). Our purpose is not to present a new framework or pattern for creative systems; rather our purpose is to discuss from an *implementation* standpoint how typicality and novelty can be modeled so as to explicitly leverage their unique contributions and simultaneously ensure that both are effectively achieved. In what follows we examine the suitability of a previously unexplored model in CC—a *constrained probabilistic model*—for this purpose. We describe how the dual nature of this model mirrors the dual properties of typicality and novelty and how the model strikes an appropriate balance between them. As a concrete example of the effective application of these models to generate novelty and typicality, we describe an implementation of a constrained Markov model, NhMMonic, for generating mnemonic devices. We show using evaluative surveys that the system generates mnemonics that demonstrate typicality, novelty, and value (as measured by how well the mnemonic facilitates memorization and learning).

Parallels Between Computational Creativity and Constrained Probabilistic Modeling

Computational creativity can be thought of as a generative act in which, for some particular domain, the set of possible artefacts $\mathcal{D} = \{x_1, \ldots, x_n\}$ is represented as a random variable X that with probability $P(x_i)$ takes on the value x_i. The primary strength of probabilistic models is that they generalize well from a set of training examples to be able to generate novel artefacts. Inasmuch as this generalization is accomplished independent of the biases of the system designer, it lends strength to the argument that probabilistic systems possess some degree of autonomy beyond manually-crafted rule-based systems. In practice, implementing a creative system in this manner presents two challenges.

One challenge is determining the probability distribution $P(X)$: with what probability should the model generate a particular x_i? This challenge can be solved explicitly—as in the case of systems that manually encode a generative process—or implicitly—as in the case of systems that attempt to learn abstract statistical properties from a set of training examples.

Prior to or in the course of resolving the first challenge, we face a second, more formidable challenge: defining the domain $\mathcal{D}$ itself. Decisions about whether a particular artefact x_j belongs or does not belong to $\mathcal{D}$ can vary from one individual to the next (Koren 2010). For now let us assume that D exists as a "fuzzy" subset of some larger domain, which we shall call $U_{\mathcal{D}}$ and which represents the universal set of all artefacts that can be represented using the same language with which artefacts in $\mathcal{D}$ are represented. For example, the domain of haiku exists as a subdomain of natural language generally. The domain of musical chorales exists as a subdomain of musical compositions generally. The fuzziness of the set $\mathcal{D}$ can derive from a variety of issues such as the difficulty in precisely defining $\mathcal{D}$ or the willingness of domain experts to accept artefacts that (to varying extents) break the rules typical of an artefact in $\mathcal{D}$.

Any particular creative system defines a set that more or less approximates $\mathcal{D}$ and possibly includes some artefacts that are less commonly agreed upon as belonging to $\mathcal{D}$ (see Figure 2). How this set is implemented is important in designing creative systems that efficiently generate artefacts in $\mathcal{D}$. For rule-based systems, the rules by which an artefact belongs within the set are hard-coded; logic is designed to prevent consideration of artefacts that break rules of the domain beyond some threshold. For evolutionary models, this set can be defined by designing a fitness function that penalizes artefacts outside of this domain. The set can also be defined as a set of constraints given as input to a constraint satisfaction solver, but with limited sense of how good one solution is with respect to another (Onarheim and Biskjaer 2017).

In the process of generalization, probabilistic models trained with artefacts from $\mathcal{D}$ are typically capable of generating artefacts that do not belong in $\mathcal{D}$. Increased expressive power in these models (i.e., the ability to generalize novel solutions) derives from maximizing independence relationships between elements of an artefact (e.g., being able to model rhythm and pitch separately in a music composition). This process can, however, lead to the generation of artefacts whose combined elements produce artefacts that most would agree do not belong in $\mathcal{D}$.

Suboptimal solutions exist to ensure that a probabilistic

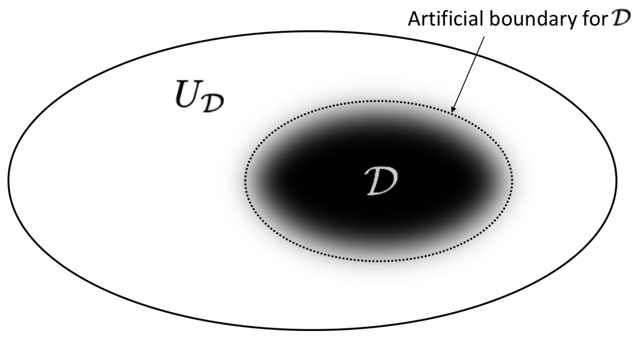


Figure 2: In many forms of creativity, the set of domain artefacts $\mathcal{D}$ exists as a structured subset of a larger domain $U_{\mathcal{D}}$ of all artefacts that can be represented using the same language as is used to describe artefacts in $\mathcal{D}$. Due to the inherent difficulty of defining belonging to a particular domain for a general audience, the set of artefacts included in $\mathcal{D}$ is in reality somewhat vague. In practice creative systems define a set that approximates $\mathcal{D}$ which defines the expressive range of the model. The extent to which this set includes or excludes artefacts that are commonly accepted as belonging to $\mathcal{D}$ controls how conservative or liberal the model will be in judging whether or not an artefact is representative of the domain.

model generates artefacts within the domain D of interest. Probabilistic models *could* ensure their output by minimizing independence assumptions (i.e., forcing the model to generate solutions more similar to the training data). This solution significantly decreases the model's ability to discover novelty from the training data. This solution also requires training on data that is more precisely representative of $\mathcal{D}$. A second suboptimal solution is the generate-and-check or rejection sampling model: probabilistically generate artefacts using the over-generalized model and then filter results to those within the D (Pease, Guhe, and Smaill 2010). This solution not only creates inefficiencies, but often assigns low probability to artefacts belonging to $\mathcal{D}$ (Ventura 2017). In such cases it becomes improbable that the system generates valid artefacts in reasonable time (Pachet, Roy, and Barbieri 2011).

A better solution to the problem of enforcing the model's domain of artefacts is the incorporation of constraints into a model that maintains probabilistic reasoning. The "fundamental entwinement of constraints and creativity" has been noted as an area of recent interest for creativity research, "with skillful and innovative handling of constraints seen as a prerequisite for apt creative performance" (Onarheim and Biskjaer 2017).

A constrained probabilistic model defines a set of rules for belonging in $\mathcal{D}$ as a set of constraints $\mathcal{C}$. Given $\mathcal{C}$ and a probability distribution $P_{U_{\mathcal{D}}}(x_j)$ for all artefacts in $x_j \in U_{\mathcal{D}}$, a constrained probabilistic model defines the probability of generating an artefact x_i as

$$P(x_i) \propto \begin{cases} P_{U_{\mathcal{D}}}(x_i) & \text{if } x_i \text{ satisfies } \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

By defining constraints explicitly, the model can be trained on artefacts from $U_{\mathcal{D}}$ generally, maintain independence assumptions that maximize expressivity, and ensure probability within the generative model is only assigned to artefacts which belong to $\mathcal{D}$.

There are several types of constrained probabilistic models including multi-valued decision diagrams (MDDs) for sequential domains (Perez and Régin 2017); MDDs that enforce constraints on non-discretized temporal sequences (Roy et al. 2016); factor graphs for imposing constraints represented as regular languages Papadopoulos et al.; and non-homogeneous Markov models (Pachet, Roy, and Barbieri 2011). Each model incorporates a probabilistic element designed to imitate statistical properties of a corpus—with model parameters (e.g., Markov order or context length) that control the degree of similarity to the corpus—and constraints to guarantee specifiable characteristics of the application domain. Previous work has also shown how constraints can be used avoid plagiarism (i.e., limit the model's output domain to $\mathcal{D}$ less the artefacts used for training) (Papadopoulos and Roy 2014). It is of interest to note that much of the language used to describe the implementation of these models mirrors closely the language used to by creative theorists to describe the relationship between novelty and typicality. For example, Perez and Régin (2017) describe the process by which the model generates new phrases as a "sampling of the solution set while respecting probabilities," specifying that the solution set "incorporate[s] some side constraints defining the type of phrases we would like to obtain."

Quality Assurance

We have discussed how constrained probabilistic models are well-suited for explicitly modeling typicality and novelty, but what about quality? As Boden (2003) puts it, "a computer could merrily produce novel combinations till kingdom come. But would they be of any interest?" How well are constrained probabilistic models able to produce or evaluate *quality*?

To the extent that quality can be represented in either the system's probabilistic model *and/or* the system's constraint set, a constrained probabilistic model is naturally endowed with a function for evaluating the quality of the artefacts. By structuring the system's probabilistic model such that high quality artefacts (by some definition of quality) are assigned higher probability, the system will naturally gravitate towards stochastically generating artefacts of value (as will be shown in our demonstrative example). In cases where quality is a function of the presence or absence of certain characteristics (consider, for example, assessing quality based on the presence of satisfactory rhymes), the system's constraints can ensure that only artefacts of some minimum quality threshold are generated.

A constrained probabilistic model thus does not define its own function for evaluating quality, but *does* inherently encode one in the forms of probabilistic models and sets of constraints (both of which could be explicitly defined or themselves learned from some training data, as demonstrated in (Bodily, Bay, and Ventura 2017)).

Non-Homogeneous Markov Models

We describe a computational creative system for generating mnemonic devices using a non-homogeneous Markov model (NHMM), a constrained probabilistic model that is also called a constrained Markov model (Pachet, Roy, and Barbieri 2011).

A Markov model $\mathcal{M}$ is a stochastic, probabilistic model defined over a finite state space that strictly adheres to the Markov property, meaning $\mathcal{M}$ is memory-less beyond a finite window. The set of all sequences $s = s_1, \ldots, s_n$ of length n generated by $\mathcal{M}$ is represented by S (in our current example this can be thought of as being equivalent to $U_{\mathcal{D}}$ from above). Every sequence $s \in S$ has a non-zero probability equivalent to

$$P_{\mathcal{M}}(s) = P_{\mathcal{M}}(s_1) \cdot P_{\mathcal{M}}(s_2|s_1) \cdots P_{\mathcal{M}}(s_l|s_{n-1})$$

$\mathcal{M}$ is constructed by computing the probability matrix $P_{\mathcal{M}}$ from training examples.

A non-homogeneous Markov model $\mathcal{N}$ is constructed from a Markov model $\mathcal{M}$, a sequence length l, and a finite sequence of unary constraints $\{C_1, \ldots, C_l\}$. The set of solutions for $\mathcal{N}$ is represented by S_C (equivalent to bounded $\mathcal{D}$ from above). With the constraints applied to $\mathcal{N}$, the probabilities of sequences generated by $\mathcal{N}$ must equal the probability of the same sequence generated by $\mathcal{M}$:

$$P_{\mathcal{N}}(s) = \begin{cases} P_{\mathcal{M}}(s) & \text{if } s \in S_C \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{N}$ initially constructs $l - 1$ probability matrices identical to $P_{\mathcal{M}}$ in $\mathcal{M}$, one for each transition in the sequence to be generated. States or transitions that violate a constraint are removed. Arc consistency is then enforced on the probability matrices, meaning that states or transitions that do not lead to a solution $s \in S_C$ are removed (see Figure 3b). Because the probability matrices in the NHMM are arc consistent and therefore non-zero probabilities are guaranteed to lead to a solution $s \in S_C$. This guarantee of solutions avoids the inefficiency generate-and-check where nearly all samples are rejected when the probability of a solution is small. Finally, the model is re-normalized such that probabilities $P_{\mathcal{N}}(s) = P_{\mathcal{M}}(s|s \in S_C)$ (Pachet, Roy, and Barbieri 2011).

NHMMs have been applied to model music generation, generating melodies constrained to begin and end on the same note (Pachet, Roy, and Barbieri 2011). Barbieri et al. (2012) apply NHMMs to generate lyrics matching rhyme, syllable stress, part-of-speech, and semantic constraints.

NhMMonic

Here we demonstrate the application of constrained probabilistic modeling to computational creativity through *non-homogeneous Markov modeling of mnemonics* (abbreviated as NhMMonic). We define a *mnemonic task* as a sequence of words $s = s_1, \ldots, s_l$ to be memorized. A *mnemonic device* then is a sequence of words $m = m_1, \ldots, m_l$ of the same length generated such that for all $1 \leq i \leq l$ the first letters in the words s_i and m_i are constrained to be the same (see Figure 3). The primary purpose of a mnemonic device is to aid in memorization of the order and/or identity of a s by finding a more memorable sequence m that through its constrained similarity to s can serve as a reminder of s. The value of an artefact in this domain is heavily predicated on its effectiveness in facilitating memory.

To our knowledge no mnemonic device generation models have been formally presented. We find that most available Mnemonic generation tools online use what we will call a *template* method. The template method for mnemonic generation first determines a sequence of part of speech constraints as a function of the length l of the sequence to be generated. Words matching these constraints and the aforementioned first-letter constraints are randomly selected from a word bank to fit into the specific sentence structure. The shortcoming to most template-based methods is that they do not model transitions between words, resulting in phrases that exhibit grammatical cohesion, but not semantic cohesion.

Because NHMMs explicitly model transitions between words while allowing for constraints, we consider this model a good candidate for the mnemonic problem. Although NHMMs can and have been used to impose part-of-speech constraints or templates, we chose not to include these constraints in our NHMM implementations preferring to demonstrate that even a relatively simple NHMM can provide good results. While we expect both models to be capable of generating novelty (or *uniqueness* as it is labeled in our survey), we expect NHMMs to outperform other mnemonic device models when it comes to the aspects of typicality relating to grammatical/semantic cohesion and ease of memorization.

Methods

In assessing the NhMMonic system we applied two variants of NHMMs. NHMM-1 has a Markov order of 1 and NHMM-2 has a Markov order of 2 (essentially treating each *pair* of words as a single state token). A higher Markov order allows the mnemonic output to more closely resemble the sample text, increasing the model's cohesion and typicality. A drawback of having a higher Markov order is that fewer solutions $s \in S_C$ are found and in some cases no solutions are found given finite training sentences. NHMM-1, with its lower Markov order, allows our system to find solutions when NHMM-2 does not.

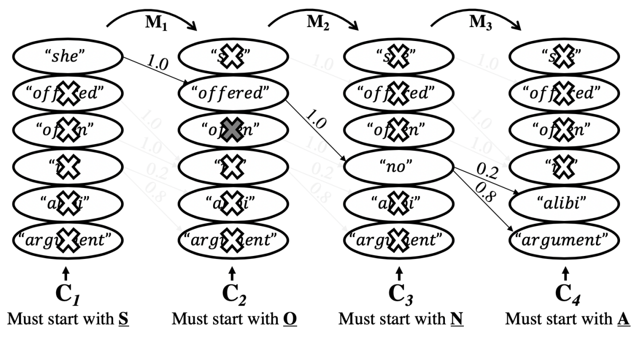
For a mnemonic task $s = s_1, \ldots, s_l$, we derive a unary constraint oat position i to ensure that the first character of the sequence variable m_i matches the first given letter of s_i. For the purposes of improved readability of generated mnemonics we impose a few additional constraints. For NHMM-1, we constrain each sequence variables m_i to be at least 4 letters long and the last variable m_l to have ended a sentence in the training set. For NHMM-2 the only added constraint is to ensure that the last variable m_l is not a pronoun, preposition, conjunction, or determiner.

The code for the NHMMs used by the NhMMonic system are available in both a C++ implementation[2] (used for

[2] `https://github.com/po-gl/ConstrainedMarkovModel`

Stream-enterer, **O**nce-returner, **N**on-returner, **A**rahant

(a) A Mnemonic Task



(b) Constrained Probabilistic Model (NHMM)

"***S**he **o**ffered **n**o **a**rgument*"

(c) Mnemonic Device Generation

Figure 3: *The NhMMonic model.* (a) A mnemonic task (i.e., the four stages of enlightenment) to be memorized. (b) A non-homogeneous Markov model built to solve the mnemonic task. M_1, M_2, and M_3 represent Markov constraints; C_1, C_2, C_3, and C_4 denote unary constraints derived from the task. Nodes marked with white X's are removed due to violation of unary constraints while the node marked with a grey X is removed to keep the model arc consistent. Edge labels indicate transition probabilities. (c) A possible mnemonic generated by the model.

NHMM-1) and a Java implementation[3] (used for NHMM-2) online

Results

To evaluate the use of constrained Markov models for generating mnemonic devices, we devised an online survey to compare four different mnemonic device generation models:

- **Template**—a third-party model[4] that selects a part-of-speech template to match the desired sequence length and then randomly selects words matching part of speech and initial letter constraints from a hand-crafted word bank.
- **NHMM-0**—a model which randomly selects words matching initial letter constraints with probability derived from word frequencies in the training corpus.
- **NHMM-1**—a first-order NHMM as described above.
- **NHMM-2**—a second-order NHMM as described above.

The latter three models were trained on the COCA dataset (Davies 2009). NHMM-0 and NHMM-1 were trained on 6.8 million sentences from fictional works written between the years 1995 and 2015 while NHMM-2 trained on 3 million sentences from the same works.

Each model was used to generate 4 mnemonic devices for each of 19 different memorization tasks[5] (Figure 6 shows some examples of tasks included in the experiment). NHMM-2 was able to find satisfying solutions to 12 of the tasks.

To evaluate the generated mnemonics, we designed a survey in which each evaluation consisted of four parts:

1. The respondent was shown one of the 19 memorization tasks for 10 seconds.
2. The respondent was then shown a mnemonic device for the memorization task for 10 seconds (selected randomly from those generated by the four models).
3. The respondent was then given the (unordered) words from the original memorization task and asked to put them in the correct order based on his/her memory of the task and the mnemonic.
4. Lastly the respondent was asked to evaluate the mnemonic device (using Likert scales from 1 to 5) for
   (a) *memory*—ease of memorization
   (b) *flow*—grammatical/semantic coherence
   (c) *creativity*—overall creative value
   (d) *uniqueness*—degree of novelty

Each respondent completed four evaluations in this manner.

A total of 80 individuals completed the survey for a total of 320 mnemonic device evaluations. The survey was distributed to different social media websites, such as Reddit, Facebook, and Twitter. No personal information was gathered before or after the survey was taken. Figure 4 shows average scores for the four evaluated characteristics by model. The NHMM-2 model made notable improvements over other models in the categories of ease of memorization (*memory*) and grammatical/semantic cohesion (*flow*). Although the NHMM-0 model performed relatively poorly on *memory*, *flow*, and *creativity*, this model was considered equally capable of generating novelty (i.e., *uniqueness*).

Figure 5 shows the impact of task length on ease of memorization, showing generally that the longer a mnemonic task is, the more difficult mnemonics generated for the task are to remember. The graph also shows, however, that the NHMMs and NHMM-2 in particular, is able to generate mnemonics that maintain ease of memorization even for longer tasks.

Figure 6 shows seven mnemonic device tasks together with the highest-rated mnemonic devices (as per average memory score) generated by NhMMonic for the task.

Discussion

Survey results demonstrate that increased grammatical/semantic cohesion afforded by probabilistic Markov models are associated with gains in ease of memorization.

[3]`https://github.com/norkish/downbythebay/tree/master/DownByTheBay/src/dbtb/markov`

[4]Available via `https://spacefem.com/mnemonics`

[5]Mnemonics for all models can be seen at `https://tinyurl.com/yxczxjh7`

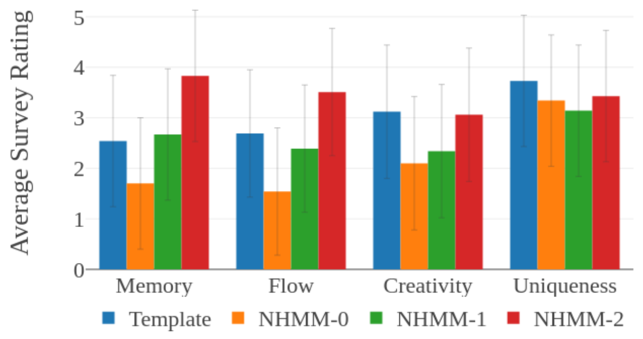


Figure 4: *Survey Results*. Average ratings from 320 evaluations across four metrics for four different mnemonic device generation algorithms. Error bars are standard deviation. The ease of memorization of mnemonics from the NHMM-2 model appears to be associated with improved flow with respect to other models.

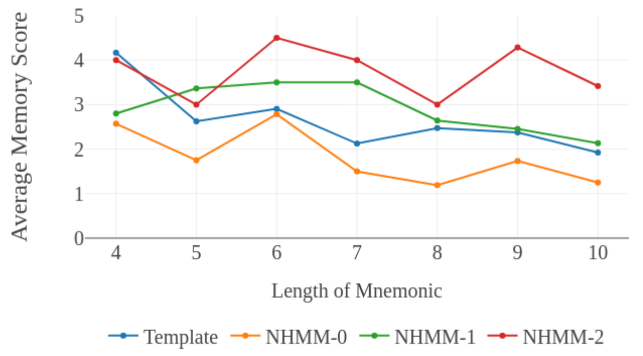


Figure 5: *Impact of Task Length*. As the length of the memorization task increases, the effectiveness of mnemonic devices decreases across all models, but at a much lesser rate for the NHMM-1 and NHMM-2 models. We hypothesize that this is owing to the sustained grammatical and semantic flow that these models achieve from the constrained Markov model.

The fact that increasing the Markov order leads to further gains in both *flow* and *memory* is further evidence of this correlation. These gains from increasing the Markov order were also mirrored in increased creativity scores, suggesting that in the domain of mnemonic device generation, there is an association between the creative success of a mnemonic device and how easily it can be remembered.

This association between the success or popularity of an artefact and the ease with which the brain is able to process and remember it has been observed in creative domains that do *not* deal directly with memorization tasks. A notable example is the study by Nunes, Ordanini, and Valsesia (2014) that demonstrates an association between the popularity of music and the degree of repetition in the song. Researchers observed that increased repetitiveness contributed to higher "processing fluency", meaning the ease with which the brain is able to grasp a new concept or artefact. A constrained Markov model, through its probabilistic transition model, naturally assigns higher probability to frequent word transitions (which we might assume have higher processing fluency) while using constraints to ensure that generated mnemonics also satisfy the basic requirements of a mnemonic device.

As is typical of Markov-based models, increasing the Markov order can also have negative consequences. The higher the order the more similarity exists between generated artefacts and the training data. Increasing the order also increases the likelihood of the model not being able to find a solution that satisfies both the (now more stringent) Markov constraints and non-Markovian constraints. Both of these problems can be overcome by training on more training data, but the amount of training data needed to sufficiently eradicate the problem increases exponentially with the Markov order.

Independent of the model training, some mnemonic tasks are inherently more difficult owing to the low frequency of words and word beginning with certain letters (this is, of course, language-specific). Consider for example trying to devise a mnemonic device in the English language for the first five dynasties of China, "Neolithic, Xia, Shang, Zhou, Qin". Solutions certainly exist, but unless the model sees examples in training of word pairs that would be suitable for each word pair in the task (less likely for infrequent collocates), the model will not be able to find them. On these types of tasks we might expect the non-Markovian models to perform better.

We considered other variations of constraints that might have further improved the results of our model. One improvement considered was to constrain more than just the first letter of each word in the mnemonic to match the task. We thought this might further increase the ease of memorization. However, it is generally the case that as constraints become more strict, the model is able to find fewer solutions, often leading to the model being unable to find satisfying solutions. Another improvement we considered was combining the Template and NHMM approaches through part of speech constraints in the NHMM model. We also considered ways to impose semantic themes within mnemonic devices either through unary semantic constraints or through vary-

Four Stages of Enlightenment: Stream-enterer, Once-returner, Non-returner, Arahant
"She offered no argument" (NHMM-2, 5.0)

Dantes 9 Circles of Hell: Limbo, Lust, Gluttony, Greed, Anger, Heresy, Violence, Fraud, Treachery
"Lovely little girl giggles as his voice for them" (NHMM-1, 5.0)

Last 10 Winners of the FIFA World Cup: France, Germany, Spain, Italy, Brazil, France, Brazil, West Germany, Argentina, Italy
"Four-year-old grandson she is bumped from behind with an inflection" (NHMM-2, 4.0)

First 9 ICCC Locations: Lisbon, Mexico City, Dublin, Sydney, Ljubljana, Park City, Paris, Atlanta, Salamanca
"Like most days she looked pretty puny and sickly" (NHMM-2, 4.5)

Stages of Grief: Denial, Anger, Bargaining, Depression, Acceptance
"Dreams about being dragged against" (NHMM-1, 5.0)

Levels of Biological Organization: Biosphere, Ecosystem, Community, Population, Organism, Organ System, Organ, Tissue, Cell, Molecule
"Blue eyes could pick out one of those clownish men" (NHMM-2, 4.5)

Cell Mitosis Cycle: Interphase, Prophase, Prometaphase, Metaphase, Anaphase, Telophase, Cytokinesis
"I pushed past me and the career" (NHMM-2, 5.0)

Figure 6: *Top-rated mnemonics generated by NhMMonic*. Seven mnemonic device tasks are shown. Each task consists of a description (bold and underlined) followed by a list of words requiring a mnemonic device. Below each task is the NhMMonic-generated mnemonic device that received the highest memorization score (with the exact model and score given in parentheses).

ing the training data. We leave these as exploratory ideas for future work.

Many forms of creativity have relational structure (e.g., rhyme schemes, repeated motifs, etc.). Unlike the example we have shown here which uses solely unary constraints, relational structure is most effectively realized using binary constraints. Sampling from constrained Markov models with binary constraints is known to be a much harder problem (see (Rivaud and Pachet 2017)), however recent work has been done towards providing reasonable solutions (Papadopoulos et al. 2015; Roy et al. 2016). This has relevance for imposing semantic constraints in models of mnemonic device generation because binary constraints can effectively be used to impose *floating constraints* (i.e., constraints that can be satisfied at variable positions) rather than specifying a specific word position where semantic constraints must be satisfied.

NHMM doesn't directly model all aspects of creativity. For example intention, explicit self-evaluation, others? Constraints themselves can be learned or imitated. One ramification of learned constraints is that in addition to whatever constraints are required to define typicality, additional constraints could themselves be probabilistically applied in generating artefacts. This would allow constraints to be "broken" (or rather never applied) with some degree of probability, demonstrating a method by which rules can be "intelligently" broken.

In this work we have discussed aspects of constrained probabilistic modeling that are well-suited for consistently generating novelty and typicality in computational creative artefacts. As an example, we have demonstrated the application of non-homogeneous Markov models to the problem of mnemonic device generation. Our results suggest that the constrained Markov model approach is able to effectively generate mnemonic devices that satisfy basic requirements of mnemonic devices while exhibiting elevated levels of grammatical/semantic flow, ease of memorization, and creative value.

Acknowledgements

Many thanks to the team at spacefem.com for their assistance using the spacefem mnemonic device generator.

References

Barbieri, G.; Pachet, F.; Roy, P.; and Esposti, M. D. 2012. Markov constraints for generating lyrics with style. In *Proceedings of the Twentieth European Conference on Artificial Intelligence*, 115–120.

Boden, M. A. 2003. *The Creative Mind: Myths and Mechanisms, Second Edition*. Routledge.

Bodily, P.; Bay, B.; and Ventura, D. 2017. Computational creativity via human-level concept learning. In *Proceedings of the Eighth International Conference on Computational Creativity*, 57–64.

Colton, S., and Wiggins, G. A. 2012. Computational creativity: The final frontier? In *Proceedings of the Twentieth European Conference on Artificial Intelligence*, 21–26. IOS Press.

Csíkszentmihályi, M. 1996. *Flow and the Psychology of Discovery and Invention*. Harper Perennial.

Davies, M. 2009. The 385+ million word Corpus of Contemporary American English (1990–2008+): Design, architecture, and linguistic insights. *International Journal of Corpus Linguistics* 14(2):159–190.

Jordanous, A. 2016. Four PPPPerspectives on computational creativity in theory and in practice. *Connection Science* 28(2):194–216.

Koren, L. 2010. *Which "aesthetics" do you mean? : Ten definitions*. Point Reyes, California: Imperfect Publishing.

Morris, R. G.; Burton, S. H.; Bodily, P. M.; and Ventura, D. 2012. Soup Over Bean of Pure Joy : Culinary ruminations of an artificial chef. In *Proceedings of the Third International Conference on Computational Creativity*, 119–125.

Nunes, J. C.; Ordanini, A.; and Valsesia, F. 2014. The power of repetition: Repetitive lyrics in a song increase processing fluency and drive market success. *Journal of Consumer Psychology* 25(2):187–199.

Onarheim, B., and Biskjaer, M. M. 2017. Balancing constraints and the sweet spot as coming topics for creativity research. In *Creativity in design: Understanding, capturing, supporting*. APA.

Pachet, F.; Roy, P.; and Barbieri, G. 2011. Finite-length Markov processes with constraints. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.

Papadopoulos, A., and Roy, P. 2014. Avoiding plagiarism in Markov sequence generation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2731–2737.

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

Pease, A.; Guhe, M.; and Smaill, A. 2010. Some aspects of analogical reasoning in mathematical creativity. In *Proceedings of the First International Conference on Computational Creativity*, 60–64.

Perez, G., and Régin, J.-C. 2017. MDDs: Sampling and probability constraints. In *Proceedings of the Twenty-Third International Conference on Principles and Practice of Constraint Programming.*, 226–242. Springer, Cham.

Pérez y Pérez, R., and Sharples, M. 2004. Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA. *Knowledge-Based Systems*.

Ritchie, G. 2007. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines* 17(1):67–99.

Rivaud, S., and Pachet, F. 2017. Sampling Markov models under constraints: Complexity results for binary equalities and grammar membership. *arXiv preprint*.

Roy, P.; Perez, G.; Régin, J.-C.; Papadopoulos, A.; Pachet, F.; and Marchini, M. 2016. Enforcing Structure on Temporal Sequences: The Allen Constraint. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 786–801. Springer.

Saunders, R., and Gero, J. S. 2001. The Digital Clockwork Muse: A Computational Model of Aesthetic Evolution. In *Proceedings of the Artificial Intelligence and Simulation of Behavior Convention*, 12–21.

Ventura, D. 2017. How to Build a CC System. In *Proceedings of the Eighth International Conference on Computational Creativity*, 253–260.

Wiggins, G. A. 2006. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems* 19(7):449–458.