In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission for extensive copying of my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

Phrasal Category Tagging for Improved Semantic Coherence in Constrained Hidden Markov Processes

by

Brandon Biggs

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in the Department of Computer Science

Idaho State University

August 2022

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Brandon Biggs find it satisfactory and recommend that it be accepted.

Paul Bodily,
Major Advisor

Leslie Kerby,
Committee Member

Marco Schoen,
Graduate Faculty Representative

ACKNOWLEDGMENTS

I would like to acknowledge my partner, Kayla. She has been incredibly supportive of my academic and professional goals. She has continually motivated and encouraged me when I needed it the most.

I would also like to acknowledge my advisor, Dr. Paul Bodily. Paul's patience, guidance, thoughtfulness, and expertise cannot be understated. I feel incredibly lucky to have had him as my advisor and could not have asked for a better experience because of him.

Additionally, I would like to thank my committee members, Dr. Leslie Kerby and Dr. Marco Schoen for their time and expertise. Furthermore, I want to extend thanks to Dr. Mustafa Mashal for being willing to participate on my committee as well.

I also want to extend a specific thanks to several Idaho State University administrative and professional staff. The support of Ronda Mahl, Ellen Combs, Jenni Overocker, Laura Blad, and Miriam Dance was incredible. The university is fortunate to have incredible people like them around.

Lastly, I want to thank the faculty and staff of the College of Science and Engineering. From working with everyone on a professional level to being a student and taking many of their courses, the college is full of remarkable people and my educational and professional career has grown because of them.

Contents

List of Figures . . . . . **vii**

List of Tables . . . . . **xiii**

Abstract . . . . . **xiv**

1 Introduction **1**

2 Definitions **5**

2.1 Markov Models 5

2.2 Hidden Markov Models 6

2.3 Constrained Markov Process 9

2.4 Constrained Hidden Markov Process 11

3 Probabilistic Generation of Sequences Under Constraints **13**

3.1 Abstract 13

3.2 Introduction 14

3.3 Related Work 15

3.4 Methods 17

3.4.1 Markov Processes 18

3.4.2 Hidden Markov Processes 20

3.4.3 Constrained Markov Processes 23

3.4.4 Constrained Hidden Markov Processes 25

3.5 Applications 27

3.6 Conclusion 31

4 "She Offered No Argument": Constrained Probabilistic Modeling for Mnemonic Device Generation 32

4.1 Abstract 32

4.2 Introduction 33

4.3 Parallels Between Computational Creativity and Constrained Probabilistic Modeling 36

4.3.1 Quality Assurance 40

4.4 Non-Homogeneous Markov Models 41

4.5 NhMMonic 42

4.6 Methods 43

4.7 Results 44

4.8 Discussion 45

5 A Leap of Creativity: From Systems that Generalize to Systems that Filter 53

5.1 Abstract 53

5.2 Introduction 54

5.3 Methods 56

5.4 Results 59

5.5 Discussion and Conclusion 61

6 Phrasal Category Tagging 66

6.1 Introduction 66

6.2 Methods 67

6.3 Evaluation 70

6.3.1 Results 73

6.4 Discussion 75

6.4.1 Ethical Concerns 76

6.5 Conclusion 77

7 Summary 79

References 80

List of Figures

2.1 Markov model trained the sentences "John likes the blue house at the end of the street. Mary lives by the blue house in the green house." 6

2.2 Hidden Markov model trained on the sentences "John likes the blue house at the end of the street. Mary lives by the blue house and in the green house." where the hidden state space is the lexical category of each word in the training set. 8

2.3 Example of a constrained Markov model where the first word must be "the" and the word "green" is not allowed in the generated sequence 10

2.4 Hidden Markov model trained the sentences "John likes the blue house at the end of the street. Mary lives by the blue house in the green house." where the hidden states are the parts of speech and a constraint has been applied that removes the hidden part of speech "JJ" (adjective). 11

3.1 A constrained Markov process (CoMP) with constraints requiring the first token rhyme with *red* and the last token be *red*. Pruned states and updated transitions are the result of applying constraints and then enforcing arc-consistency. 24

3.2 A high-level schematic of a constrained hidden Markov process (CHiMP) of length 4 constrained so that the last word is "red" and the first word rhymes with "red". Each column represents a position in the sequence to be generated. Each node represents a hidden state (i.e., part-of-speech) and a probability distribution for the observed states (i.e., words) that can be generated from that hidden state. By pruning observed states that are disallowed by constraints and then adjusting probabilities to maintain arc-consistency, the resulting model generates constraint-satisfying solutions with probability relative to the original probability distribution. Hidden states pruned directly from applying constraints are indicated by dark grey nodes and states pruned during arc-consistency are indicated by light grey nodes. 26

3.3 The effects of sequence length (and consequently number of constraints) on generalizability (i.e., number of unique sequences out of 10k sampled solutions) for a fixed random training set of 100 sentences. Each model is constrained such that words start with the same letter, and counts are averaged over 26 runs (a different letter constraint for each run). The added constraints from increasing sequence length have a compounding limiting effect in the CoMP model, whereas the abstraction of the CHiMP model serves to decouple constraints to avoid bottlenecks. 29

3.4 The effects of training corpus size on generalizability of the CHiMP (blue) and CoMP (orange) models. Generalizability is measured as number of unique sequences out of 100k sampled solutions. Each model is constrained such that words start with the same letter, and counts are averaged over 26 runs (a different letter constraint for each run). Shades show the effects of varying the sequence length (and consequently the number of constraints) on generalizability. The CHiMP model consistently generates more unique satisfying solutions than the CoMP model and is relatively immune to the effects of training set size or number of constraints. 30

4.1 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 [72]. 35

4.2 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. 38

4.3 *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. 49

4.4 *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. . 50

4.5 *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. 51

4.6 *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). 52

5.1 Ventura's [76] spectrum of creative systems provides a means by which to measure the progress of a system towards becoming creative. Characterizing challenges and solutions that are specific to each level in the spectrum helps to actualize the spectrum into becoming a guide for building more creative systems. 54

5.2 A high-level schematic of a constrained hidden Markov process (CHiMP) of length 4 constrained so that the last word is "red" and the first word rhymes with "red". Each column represents a position in the sequence to be generated. Each node represents a hidden state (i.e., part-of-speech) and a probability distribution for the observed states (i.e., words) that can be generated from that hidden state. By pruning observed states that are disallowed by constraints and then adjusting probabilities to maintain arc-consistency, the resulting model generates constraint-satisfying solutions with probability relative to the original probability distribution [30]. Hidden states pruned directly from applying constraints are indicated by dark grey nodes and states pruned during arc-consistency are indicated by light grey nodes. 57

5.3 The application of filters on two hypothetical models (A and B) demonstrates the requirement for larger solution spaces (increased generalization) in order to endure filtering with a usable solution space. Model B has a usable solution space after filtering; thus the model has moved further along in the spectrum from generalization to filtration. 59

5.4 Example results from generating 6-length tongue twisters (i.e., alliterative constraints) from both the CoMP and CHiMP models. Both models were trained on 10K sentences. Results are chosen from a randomly selected subset of 40 sequences from each model. The quality of tongue twisters is roughly equivalent between both models (both poor), but the CHiMP model is capable of generating exponentially more solutions. This suggests that increasing the Markov order in the CHiMP model (as an example of more stringent constraints) will have far less deleterious affects on the solution space as compared to a similar increase in the CoMP model. 61

5.5 The effects of sequence length on the number of total solutions generated by each model with a fixed training set size of 300 sentences. Both models are constrained such that each word in a sequence starts with the same letter; counts of total solutions are averaged over 26 runs (each run using a different letter from the English alphabet). We see that as the sequence length increases, total solutions for the CHiMP model increases exponentially (given the logarithmic scale) whereas the CoMP model stagnates. 62

5.6 The effects of training corpus size (number of training sentences) on the number of total solutions generated by each model with a fixed sequence length of 3. Both models are constrained such that each word in a sequence starts with the same letter; counts of total solutions are averaged over 26 runs (each run using a different letter from the English alphabet). The total solutions of both models increase in an almost parallel way; however, at 10K training sentences, CHiMP well exceeds 100M total solutions which contrasts CoMP at 1000 total solutions. 63

6.1 Parse tree of the sentence "John likes the blue house at the end of the street". The root tree node is an 'S' which represents the sentence as a whole. This is then broken into each phrasal and lexical category, each of which is color coded. 68

6.2 Parse tree of the sentence "John likes the blue house at the end of the street". The root tree node is an 'S' which represents the sentence as a whole. This is then broken into each phrasal and lexical category, each of which is color coded. This has the addition of each level of the parse tree being contained for the purposes of showing how the transition matrix is formed. 69

6.3 Semantic Survey results 74

6.4 Theme Survey results 75

List of Tables

2.1 Transition probabilities (δ) for M 7

2.2 Lexical categories used in the model and their acronym. 8

2.3 Transition probabilities (δ) for the hidden Markov process where hidden states are the lexical categories. 9

2.4 Emission probabilities (ϵ) for the hidden Markov process where hidden states are parts of speech. These are the probabilities of the hidden state (part of speech) producing the observable state (word). 9

2.5 Updated transition probabilities (δ) for the constrained Markov process where 2 constraints have been applied: the word "green" is not allowed and the generated sequence must start with the word "the". 10

2.6 Transition probabilities (δ) for the constrained hidden Markov process where hidden states are parts of speech and a constraint is applied to remove the JJ part of speech (adjective). 12

2.7 Emission probabilities (ϵ) for the constrained hidden Markov process where hidden states are parts of speech. These are the probabilities of the hidden state (part of speech) producing the observable state (word) and the constraint is that there cannot be any adjectives. 12

6.1 Phrasal categories used in the examples and their respective acronyms. . . . 70

Phrasal Category Tagging for Improved Semantic Coherence in Constrained Hidden Markov Processes

Thesis Abstract – Idaho State University (20222022)

State of the art machine learning models that focus on natural language processing are powerful, but also complex and expensive, both computationally and financially. Some generative tasks may require substantially large language models. Larger models are also not often accessible as access is sold as a service or requires advanced technical knowledge. Some natural language processing tasks however, such as short sequenced natural language generation may not require the use of these complex and expensive models. Hidden Markov models are a historically well known model that are observable, interpretable, and better suited for small scale generative sequence tasks. To further improve the generative capabilities, the constrained hidden Markov process (CHiMP) model was introduced in previous work to allow control over generated sequences by focusing on lexical categories and constraints on those lexical categories. This work improves upon the CHiMP model to an increased cohesive level by adding phrasal categories to the hidden state space, and by using floating constraints on the phrasal categories.

Keywords: natural language processing, markov model, constrained sequence generation, machine learning, statistical models

Chapter 1

Introduction

Natural Language Processing (NLP) is the area of computer science research focused on spoken and text languages. This includes understanding, creating, and manipulating language through computer-based methods [43]. This is an active research area that has produced automatic language translation [48], social media screening [21], and personal voice assistants [67]. These tools have been improved by understanding how language is used to express thoughts, emotions, and ideas [18].

NLP has advanced in three stages. The first stage is the symbolic approach which started in the 1950s and ended in the early 1990s [23]. This stage was characterized by computer-emulated natural language tasks given a set of rules. It was during this stage that language grammar was hand coded. The challenge was that language grammar exceptions exist. With the large list of rules for each language, this hand coded list of language rules was inefficient to maintain.

The second stage is statistical NLP which was driven by advancing computer hardware and machine learning algorithms [46]. Early statistical NLP tasks such as lexical categories (part-of-speech) tagging used Markov and hidden Markov models and semantic information to decipher parts of the English language [80]. Continuing work has led researchers to other statistical models such as the "Maximum Entropy Model" which also used context clues for lexical categories tagging [68], and n-gram models. Researchers have approached the problem from a linguistics background, arguing that at the time, many models had little room to improve, so tasks such as lexical category tagging needed to be reevaluated [47].

Starting in approximately 2010, a third stage began, which consists of neural network natural language processing [31]. Neural networks have rapidly advanced and become the standard machine learning method with common frameworks such as Tensorflow [4], Pytorch [60], and Keras [17]. The more notable natural language processing models that have come from these frameworks include: Generative Pre-Training Transformer (GPT) 1 [65], 2 [66], 3 [14] from OpenAI; BERT [26] from Google; RoBERTa [44] from Facebook; and Megatron-Turing Natural Language Generation model (MT-NLG) from Microsoft and Nvidia [38]. Each of these models use various machine learning techniques such as long short-term memory (LSTM) [34], recurrent neural networks [73], and transformers [81]. These models create NLP content such news articles [2], computer code [50], and some have been suggested to be capable of imitating humans [27]. However downsides exist; such as being expensive to create, and not interpretable. A byproduct of this lack of interpretability is that neural network models have not been specifically constrainable; that is they cannot be made to generate artifacts that strictly adhere to prescribed constraints. If constrained results were required, additional complexity would be added to filter results until a correct solution was found.

Although these neural network models have impressive performance, these improvements come at a significant cost. Each of the examples of notable neural network natural language processing models are trained on graphics processing units (GPUs). Some models are sufficiently large enough that they required thousands of very expensive GPUs. For example, attempting to train GPT-3 without GPUs would be near impossible as it would take hundreds to thousands of years to train on CPUs. Even using 8 Nvidia Volta-100 GPUs would take 37 human years [36]. One cloud provider, Xestop, allows researchers to rent 8 V100 GPUs for $8/hr [3]. Using Xestop, an estimated cost to train GPT-3 would cost around $2.6 million without additional overhead fees. Another cost estimate was $12 million [78] on more GPUs. The size of these models only continues to grow as a model 3 times the size of GPT-3 was also recently released [38].

Aside from the high costs of training these models, another frequently referenced disadvantage is their lack of interpretability and transparency [83]. In statistical NLP researchers can see the probabilistic reasoning that leads to a model's generated results. In neural networks, they are often referred to as black boxes, which means it's not clear as to how results are generated. While many modern day transformer networks are proficient at language generation, there also exists a problem with concise language generation. If a novel and targeted sequence was desired, transformer networks would need to generate sequences until one met the defined requirements. This process would require additional work and unnecessary computation time as many sequences would be generated and thrown away before a proper solution was discovered. This problem only grows as more computation time is required as the number of constraints increase.

Whereas neural network models are expensive and lack interpretability, statistical NLP models are highly efficient and are much easier to manipulate to desired effect. For this reason, there continues to be research done in the area of statistical NLP [15]. Focusing on statistical probabilities of word occurrences, or lexical category occurrences has been studied for some time. Labeled data is usually required meaning that the lexical categories of words must be tagged for the probabilities to be calculated.

Constraining generated sequences is not a new task. Constraining a sequence is a constraint satisfaction problem (CSP) [41]. In text sequence generation, constraining a sequence may be defined as requiring that a sentence starts with a specific word or letter, a sequence having an exact length, or having a combination of additional constraints. Some newer research attempts to use the strengths of neural networks models by apply differentiable constraints [40]. Other research attempts to use Monte Carlo Markov chains to generate sequences that are likely to meet all of the constraints using tree search algorithms on pre-trained language models [82]. Markov models with constraints are also not new, as text, rhyme, and subject noun matching constraints have been applied to finite length Markov models while being treated as a CSP [56].

In summary, sequences generated by neural networks excel in terms of cohesion and fluency, however they are costly and ineffective when generating sequences under specific constraints. In comparison, statistical models excel in transparency and constrainability, but generally lag in the quality and cohesion of generated results. Therein lies our challenge: can a model be designed that excels both at interpretability as well as cohesion? This thesis looks at a solution to this problem. In Chapter 2, definitions of Markov models, hidden Markov models, constrained Markov models, and constrained hidden Markov models are outlined. In Chapter 3 a constrained hidden Markov process, referred to as "CHiMP" is presented in the paper "Probabilistic Generation of Sequences Under Constraints" as published in the 2020 Intermountain Engineering, Technology and Computing (IETC) conference [30]. Chapter 4 looks at an early application of a constrained Markov process, referred to as CoMP in "She Offered No Argument": Constrained Probabilistic Modeling for Mnemonic Device Generation" published in proceedings of the tenth International Conference on Computational Creativity [11]. Chapter 5 continues the applications with "A Leap of Creativity: From Systems that Generalize to Systems that Filter" as published in the proceedings of the eleventh International Conference on Computational Creativity [29]. Chapter 6 expands on the CHiMP model, presenting new work by focusing on phrasal category tagging for improved semantic coherence. Lastly, Chapter 7 provides a short summary and conclusion.

Much of the foundational development of the model presented in this thesis was done in collaboration with the Idaho State University computer science student Porter Glines with whom the accepted publications representing Chapters 3, 4, and 5 were co-authored. Whereas Porter's subsequent research and thesis investigated the relative impact of the CHiMP model on the expressivity and cohesion of sequences in music versus natural language, the unique contribution in this thesis is an adaptation of the CHiMP model designed specifically to further improve the cohesion and ability using phrasal categories and applying floating constraints in natural language generation.

Chapter 2

Definitions

This thesis depends on understanding 4 statistical NLP models that build on one another. Although these models are included in the following chapters that represent published manuscripts, page limits for those publications necessitated brevity in the presentation of the model definitions. We include this chapter in the interest of providing an accessible format with visualized examples for the benefit of the reader. More formal definitions may be found in Chapter 3, Chapter 4, and Chapter 5.

2.1 Markov Models

A Markov model M can be defined mathematically by the triple:

$$M = \{Q, \delta, \pi\}$$

where Q is a set of states, π is a set of initial probabilities, and δ is the set of transition probabilities. In the context of English sentences, initial probabilities would be the first word of a sentence, while the transition probabilities are the probability that one word transitions to another.

A Markov model, M trained on two sentences is demonstrated in Figure 2.1. The transition probabilities, δ, of M are represented in Table 2.1. The two sentences:

1. John likes the blue house at the end of the street.
2. Mary lives by the blue house in the green house.

$$Q = \{John, likes, the, blue, house, at, end, of, street, Mary, lives, by, in, green\}$$

$$\pi = \{John : 0.5, Mary : 0.5\}$$

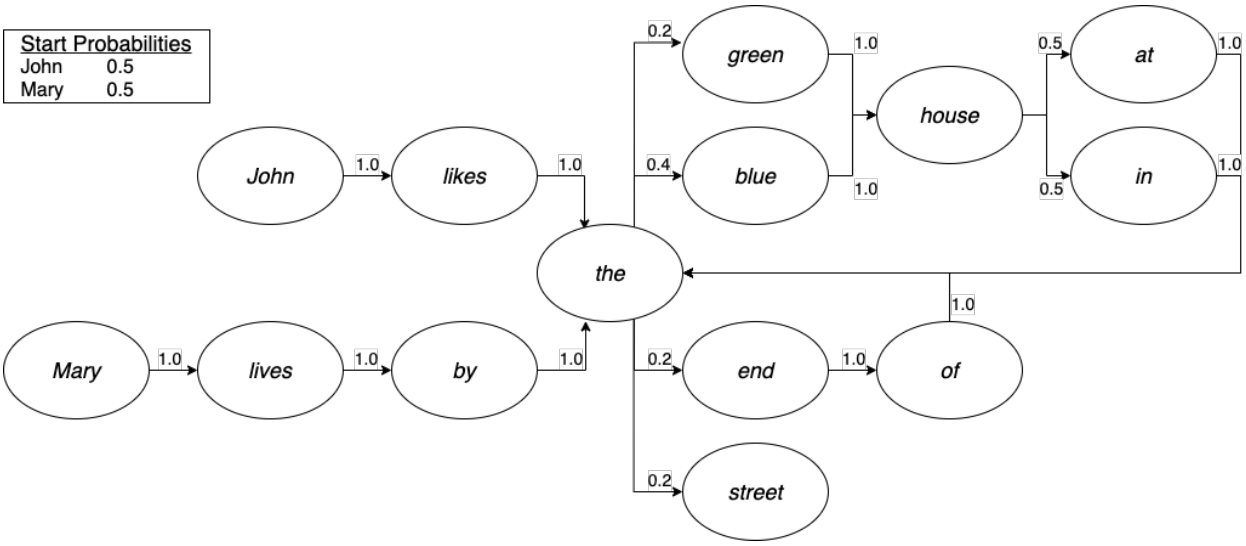


Figure 2.1: Markov model trained the sentences "John likes the blue house at the end of the street. Mary lives by the blue house in the green house."

2.2 Hidden Markov Models

One extension of Markov models are hidden Markov models. These extended models include an abstract layer of information about the state space. One such example that we focus on throughout the rest of this thesis is the transitions between the lexical categories (also known as parts-of-speech) of words rather than the transitions between the words themselves. Including this additional layer of abstraction allows the model to better mimic the understanding of language.

The hidden Markov model adds upon Markov models with two additional pieces of information, a set of hidden states and emission probabilities. A hidden Markov model can be mathematically defined by the quintuple

	John	*likes*	*the*	*blue*	*house*	*at*	*end*	*of*	*street*	*Mary*	*lives*	*by*	*in*	*green*
John	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
likes	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
the	0.0	0.0	0.0	0.4	0.0	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0	0.2
blue	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
house	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0
at	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
end	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
of	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
street	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Mary	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
lives	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
by	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
in	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
green	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 2.1: Transition probabilities (δ) for M

$$H = \{Q, V, \epsilon, \delta, \pi\}$$

where ϵ is the set of emission probabilities. Q becomes the hidden state space and V is the observable state space. The hidden states are pieces of information that are not immediately apparent in the training data, such as the lexical or phrasal categories. The emission probabilities are the calculated odds of each hidden state outputting a select observable state. Transition properties become the probabilities between the hidden states in state space rather than the observed states.

Figure 2.2 demonstrates a hidden Markov model H trained on the same two sentences

1. John likes the blue house at the end of the street.
2. Mary lives by the blue house in the green house.

where the abstracted hidden layer of information is the lexical category of each word in the training sentences. Table 2.3 demonstrates the transition values δ, Table 2.4 demonstrates

Acronym	Lexical Category
NNP	proper noun
VBZ	verb
DT	determiner
JJ	adjective
NN	singular noun
IN	preposition

Table 2.2: Lexical categories used in the model and their acronym.

the emission probabilities, and Table 2.2 explains each lexical category acronym.

$$Q = \{\text{NNP, VBZ, DT, JJ, NN, IN}\}$$

$$V = \{John, likes, the, blue, house, at, end, of, street, Mary, lives, by, in, green\}$$

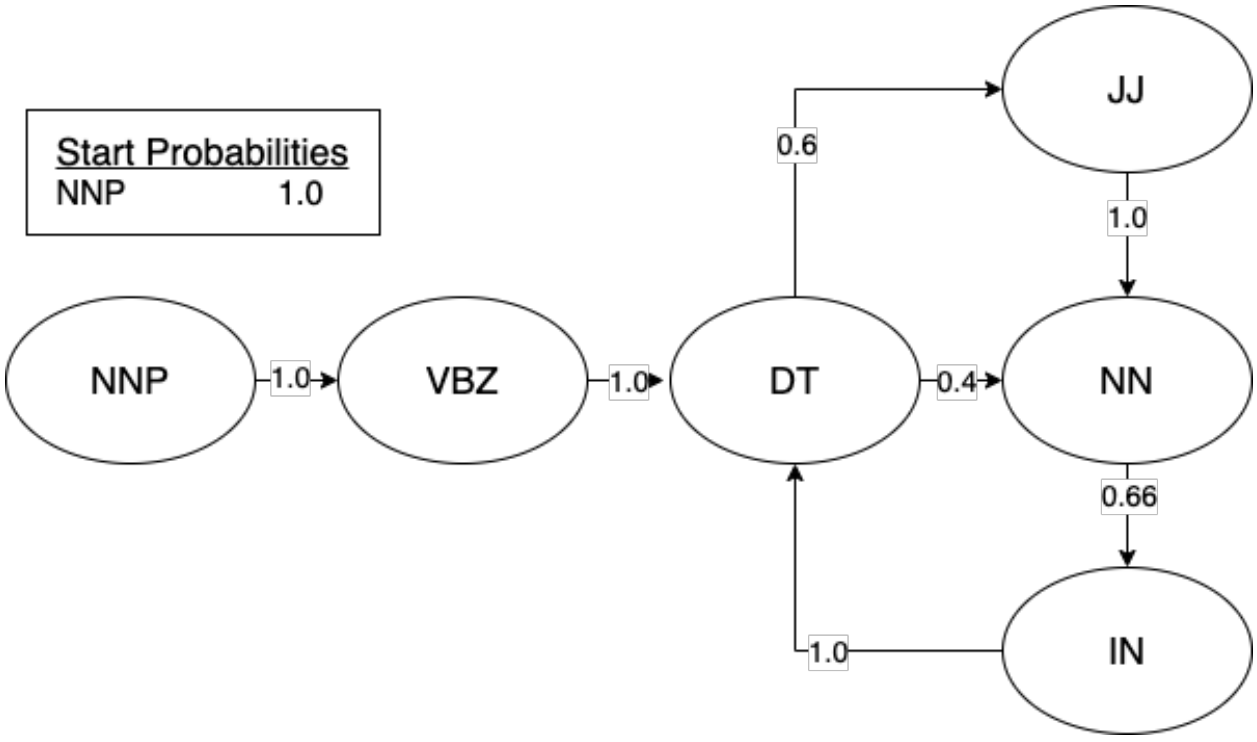


Figure 2.2: Hidden Markov model trained on the sentences "John likes the blue house at the end of the street. Mary lives by the blue house and in the green house." where the hidden state space is the lexical category of each word in the training set.

	NNP	VBZ	DT	JJ	NN	IN
NNP	0	1.0	0	0	0	0
VBZ	0	0	1.0	0	0	0
DT	0	0	0.2	0.4	0.4	0
JJ	0	0	0	0	1.0	0
NN	0	0	0	0	0	1.0
IN	0	0	1.0	0	0	0

Table 2.3: Transition probabilities (δ) for the hidden Markov process where hidden states are the lexical categories.

	John	*likes*	*the*	*blue*	*house*	*at*	*end*	*of*	*street*	*Mary*	*lives*	*by*	*in*	*green*
NNP	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0
VBZ	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0
DT	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
JJ	0.0	0.0	0.0	0.66	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.33
NN	0.0	0.0	0.0	0.0	0.6	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0	0.0
IN	0.0	0.0	0.0	0.0	0.0	0.25	0.0	0.25	0.0	0.0	0.0	0.25	0.25	0.0

Table 2.4: Emission probabilities (ϵ) for the hidden Markov process where hidden states are parts of speech. These are the probabilities of the hidden state (part of speech) producing the observable state (word).

2.3 Constrained Markov Process

Constrained Markov processes, and one of the primary inspirations of this thesis, come from a paper by François Pachet titled "Finite-Length Markov Processes with Constraints" [56]. In their paper, the authors outline a method for applying constraints to each position of the Markov model and updating the corresponding transition matrix. Using the same sentences

1. John likes the blue house at the end of the street.
2. Mary lives by the blue house in the green house.

constraints can be applied. As an example, let the first constraint be that the generated sequence has to start with the word "the" and let the second constraint be that the word "green" is not allowed in a generated sequence. Figure 2.3 provides an example of the updated constrained model.

$$Q = \{the, blue, house, at, end, of, street, in\}$$

$$\pi = \{the : 1.0\}$$

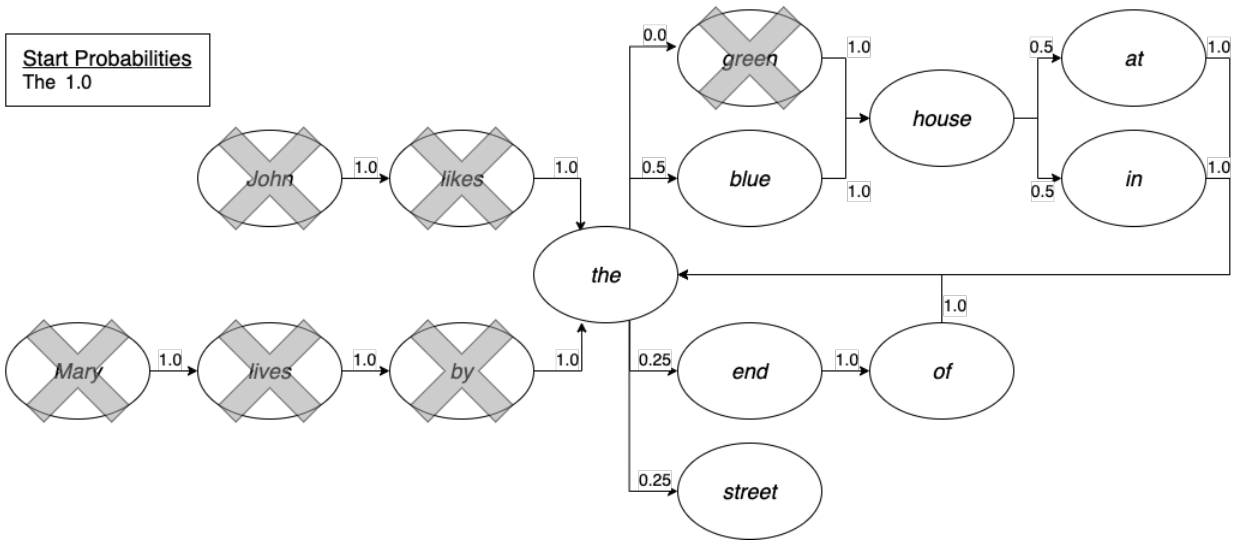


Figure 2.3: Example of a constrained Markov model where the first word must be "the" and the word "green" is not allowed in the generated sequence

	the	*blue*	*house*	*at*	*end*	*of*	*street*	*in*
the	0.0	0.5	0.0	0.0	0.25	0.0	0.25	0.0
blue	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
house	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.5
at	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
end	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
of	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
street	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
in	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 2.5: Updated transition probabilities (δ) for the constrained Markov process where 2 constraints have been applied: the word "green" is not allowed and the generated sequence must start with the word "the".

2.4 Constrained Hidden Markov Process

Constrained hidden Markov processes combine constrained Markov processes and hidden Markov models to allow constraints to be applied to either the observable states or the hidden states. Using the same training sentences

1. John likes the blue house at the end of the street.
2. Mary lives by the blue house in the green house.

and lexical categories for the hidden states, we can create Figure 2.2. Figure 2.4 shows the same hidden Markov model after applying a constraint that does not allow the "JJ" (adjective) lexical category in the model. Table 2.6 shows the transition probabilities δ and Table 2.7 demonstrates the updated emission probabilities ϵ when the constraint is applied.

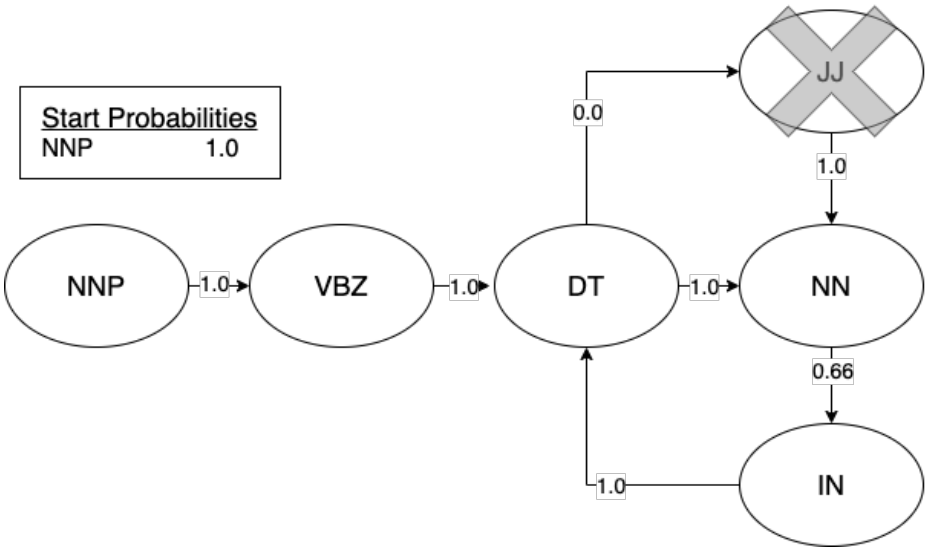


Figure 2.4: Hidden Markov model trained the sentences "John likes the blue house at the end of the street. Mary lives by the blue house in the green house." where the hidden states are the parts of speech and a constraint has been applied that removes the hidden part of speech "JJ" (adjective).

	NNP	VBZ	DT	NN	IN
NNP	0	1.0	0	0	0
VBZ	0	0	1.0	0	0
DT	0	0	0.33	0.66	0
NN	0	0	0	0	1.0
IN	0	0	1.0	0	0

Table 2.6: Transition probabilities (δ) for the constrained hidden Markov process where hidden states are parts of speech and a constraint is applied to remove the JJ part of speech (adjective).

	John	*likes*	*the*	*house*	*at*	*end*	*of*	*street*	*Mary*	*lives*	*by*	*in*
NNP	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0
VBZ	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0
DT	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NN	0.0	0.0	0.0	0.6	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0
IN	0.0	0.0	0.0	0.0	0.25	0.0	0.25	0.0	0.0	0.0	0.25	0.25

Table 2.7: Emission probabilities (ϵ) for the constrained hidden Markov process where hidden states are parts of speech. These are the probabilities of the hidden state (part of speech) producing the observable state (word) and the constraint is that there cannot be any adjectives.

Chapter 3

Probabilistic Generation of Sequences Under Constraints

This chapter was published in the 2020 Intermountain Engineering, Technology and Computing (IETC) conference [30]. I was directly involved in the development of the CHiMP model, the training process, the statistical evaluation of the model, and the normalization of constrained probabilities. This novel model is the base model used in the experiments conducted for my thesis.

Throughout the next 3 chapters, lexical categories will be referred to as "parts of speech". They can be used interchangeably and we just recently started using lexical categories as a replacement.

3.1 Abstract

There is growing interest in the ability to generate natural and meaningful sequences (e.g., in domains such as language or music). Many existing sequence generation models, including Markov and neural algorithms, capture local coherence, but have no mechanism for applying the structural constraints that are so often essential for the development of meaning. We describe a novel solution to this problem which combines hidden Markov models with constraints, allowing sequences which obey user-defined constraints to be generated according to data-driven probability distributions. Compared to other constrained probabilistic solutions, our Constrained Hidden Markov Process (CHiMP) has significantly greater expressivity, allowing the user to generate constrained sequences that are longer and which have more numerous structural constraints.

3.2 Introduction

Sequence generation is a common task in the field of artificial intelligence, particularly for domains such as music, procedural content generation, natural language generation, etc. Models used to generate sequences are trained on a corpus and are then iteratively sampled using some probabilistic distribution to generate a sequence of words. Many generative models (such as n-gram, Markov, and recurrent neural models) use previously generated tokens in the sequence to influence which future tokens are generated; however, they make no guarantee as to what a particular token will be [33, 64]. The stochastic nature of these models is desirable for many types of problems where generalization beyond the training corpus is essential to the generation of novel and yet coherent sequences.

Stochasticity becomes problematic, however, when needing to generate sequences whose meaning relies on structure. Music, for example, progresses sequentially; notes are followed by more notes. However, to elevate a musical phrase to be something interesting, higher features such as motifs need to be present in the sequence [5, 53]. Purely stochastic models have no way of ensuring that a motif or repeated pattern is generated in a longer musical phrase. Likewise coherent English sentences require proper subject-verb and noun-pronoun agreements, possibly between distant sequence positions. Quality generation of English sentences are important to any natural language application, but are increasingly in demand for applications such as virtual assistants or procedural content generation [42].

A common solution to the lack of structure in sequential data models is to combine these models with constraints. Constrained Markov processes in particular have been very successful at imposing structure when combined with constraints [7]. These models are capable of imposing structure in the form of unary constraints. For example, rhymes can be created by constraining words at different positions to belong to the same rhyme group. They have also been used in music generation [57] and to constrain against plagiarism [58].

A significant and well-known drawback of constrained Markov models (and constrained models in general) is the inability to find satisfying solutions when constraints become

numerous. This is due largely to the high coupling between states in a Markov process which causes changes in the state space at one position to directly and significantly affect the state space at neighboring positions. As more constraints are added, the diminishing state space problem is compounded to the point where the model is not able to find sequences that satisfy both the Markov constraints (i.e., state sequences allowed by the Markov transitions) and the unary constraints.

To address these shortcomings, we propose a novel solution to the constrained sequence generation problem that uses a constrained hidden Markov process (CHiMP) model. We demonstrate how the increased expressivity that emerges from hidden Markov processes with respect to non-hidden Markov processes can be similarly applied to a constrained Markov model in order to vastly increase the solution space for any constrained sequence generation problem. Lastly we provide a comparative analysis of these two models on a real-world sequence generation problem as a demonstration of the improved expressivity of the CHiMP model over other constrained Markov processes.

3.3 Related Work

Markov models (which are also very similar to n-gram models) are ideal for sequence generation; however, other types of models are commonly used for sequence generation. Neural networks in the form of recurrent neural networks (RNNs) can be used to repeatedly predict (i.e., generate) words in a sequence. RNNs differ from Markov models in that they are "fuzzy", meaning they perform higher-dimensional interpolation between training instances for their predictions [33]. This allows RNNs to synthesize and use their training data in a more complex way compared to Markov models. However, the fuzzy aspect of RNNs means that the model makes no guarantees about a generated sequence. For a generated English sentence, an RNN may exhibit a comprehension of a complex interaction present in the training data, but the longer sequence generated fails to satisfy subject-verb agreement. Combining RNNs with constraints is thus an area of ongoing research [35].

Long Short-term Memory (LSTM) are a variant of RNNs designed specificially to maintain a longer term memory about tokens that the model has previously generated [33]. LSTMs use multiple neural networks and memory cells to achieve this increased ability to remember inputs over longer periods in a sequence. This allows LSTMs to display a better understanding of lasting structure and patterns in a sequence over RNNs. An LSTM will have an easier time replicating a desired larger structure such as subject-verb agreement, however the model does not guarantee such structures will emerge.

Markov processes generate sequences by looking at the previous state (or previous n states for an n-order Markov model) and randomly sampling according to trained transition/emission probabilities to generate a new token. The process is iterated to generate a full sequence of tokens. The simplicity of these models lends to them being easy to implement and efficient to run [57]. Examples of Markov implementations include systems that react interactively to music input [55] and text-to-speech synthesis where speech waveform generation is generated via a hidden Markov model [75].

When generating sequences, Markov processes produce sequences that share a common style with the data the model was trained on. In the context of natural language synthesis, the style sharing properties of Markov processes can be exploited to change speaker identities, emotion of the speech, and the cadence of the speaker [75].

The defining feature of a Markov process is that it adheres to the Markov property which is that the next state in the sequence is determined only by the previous state to it, i.e.,

$$p(s_i|s_1, \ldots, s_{i-1}) = p(s_i|s_{i-1}).$$

In the context of sequence generation, the Markov property is well-suited to domains such as music where, aside from higher features, the next note relies on the previous note (i.e., music exhibits Markovian aspects) [12].

Previous efforts have been made to combine probabilistic models with constraint satisfaction. Pachet et al. introduce constrained Markov process (which we will refer to as

CoMP) as a method for applying user-defined constraints to a non-hidden Markov model [57]. Likewise, factor graphs combine a non-hidden Markov process with an automaton to create a system functionally equivalent to the one introduced by Pachet et al. [59]. A weakness in these models is that their ability to find satisfying solutions diminishes quickly as constraints are added or made more stringent. We quantitatively analyze this limitation in the applications section of this paper.

The CHiMP model maintains the strengths of the CoMP and Factor graph models, in guaranteeing the generation of constraint-satisfying solutions, while significantly expanding the solution space to mitigate the limitations posed in these latter models by adding numerous or strict constraints.

3.4 Methods

The primary difference between the CoMP model and the CHiMP model is that the former derives from a Markov model whereas the latter derives from a *hidden* Markov model. As such, we will review these two models first and then look at how constraints are added to these models to form the CoMP and CHiMP models

As a running example we will consider the problem of generating a sequence with the following set of constraints (hereafter referred to as the set C):

1. The sequence must be four words in length
2. The first word rhymes with *red*
3. The last word is *red*

Furthermore the sequence of words must be generated according to probabilities derived from the following training corpus (note that for the sake of simplicity, and because it is irrelevant in the context of the constrained models, we purposely ignore end-of-sentence tags):

NNP	**RB**	**VBZ**	**NN**
Ted	*now*	*likes*	*green*

NNP	**VBZ**	**NN**
Mary	*likes*	*red*

NNP	**RB**	**VBZ**	**NN**
Mary	*now*	*loves*	*red*

NNP	**VBZ**	**NNP**	**RB**
Fred	*sees*	*Mary*	*sometimes*

The tokens NN, RB, VBZ, and NNP represent part-of-speech (POS) tags equating respectively to a noun, an adverb, a verb in 3rd-person singular present, and a proper noun singular. For purposes of equal comparison all models are required to incorporate both POS and words into their state spaces. Markov models do this by combining both POS and words into a single state space; hidden Markov models explicitly separate POS and words into hidden and observed state spaces respectively.

3.4.1 Markov Processes

A Markov model is defined as a triple $M = (S_M, \pi_M, T_M)$ where S_M defines a finite set of observed states or symbols; $\pi_M : S_M \to [0, 1]$ represents the initial probabilities for states; and $T_M : S_M \times S_M \to [0, 1]$ represents the transition probabilities between states. Each of π_M and T_M represent a distribution and should thus sum to 1.0.

For the training corpus above, $M = (S_M, \pi_M, T_M)$ where $S_M = \{$(**NNP**, *Ted*), (**RB**, *now*), (**VBZ**, *likes*), (**NN**, *green*), (**NNP**, *Mary*), (**NN**, *red*), (**VBZ**, *loves*), (**NNP**, *Fred*), (**VBZ**, *sees*), (**RB**, *sometimes*)$\}$, and π_M and T_M are defined as follows:

π_M	
(**NNP**,*Ted*)	1/4
(**NNP**,*Mary*)	2/4
(**NNP**,*Fred*)	1/4
else	0

T_M	(**NNP**, *Ted*)	(**RB**, *now*)	(**VBZ**, *likes*)	(**NN**, *green*)	(**NNP**, *Mary*)	(**NN**, *red*)	(**VBZ**, *loves*)	(**NNP**, *Fred*)	(**VBZ**, *sees*)	(**RB**, *sometimes*)
(**NNP**, *Ted*)	0	1	0	0	0	0	0	0	0	0
(**RB**, *now*)	0	0	$\frac{1}{2}$	0	0	0	$\frac{1}{2}$	0	0	0
(**VBZ**, *likes*)	0	0	0	$\frac{1}{2}$	0	$\frac{1}{2}$	0	0	0	0
(**NN**, *green*)	0	0	0	0	0	0	0	0	0	0
(**NNP**, *Mary*)	0	$\frac{1}{3}$	$\frac{1}{3}$	0	0	0	0	0	0	$\frac{1}{3}$
(**NN**, *red*)	0	0	0	0	0	0	0	0	0	0
(**VBZ**, *loves*)	0	0	0	0	0	1	0	0	0	0
(**NNP**, *Fred*)	0	0	0	0	0	0	0	0	1	0
(**VBZ**, *sees*)	0	0	0	0	1	0	0	0	0	0
(**RB**, *sometimes*)	0	0	0	0	0	0	0	0	0	0

We generate a sequence $s = \{s_1, \ldots, s_n\}$ using M as follows:

1. Sample an initial state $s_1 \in S_M$ according to π_M
2. While sequence not finished,
   (a) Sample the next state $s_i \in S_M$ given s_{i-1} according to T_M

The sequence finishes either when an end-of-sequence token is sampled or when the sequence has reached some pre-determined length (as is the case in the CoMP and CHiMP models).

A Markov model M generates a sequence $s = \{s_1, \ldots, s_n\}$ of states from S_M with probability

$$P_M(s) = \pi_M(s_1) \prod_{i=1}^{n} T_M(s_{i-1}, s_i).$$

Trained on the example corpus, M generates the following sequences with the probabilities shown:

NNP	**VBZ**	**NN**		
Mary	*likes*	*green*		
1/2	$\times 1/3$	$\times 1/2$	=	1/12

NNP	**VBZ**	**NNP**	**RB**		
Fred	*sees*	*Mary*	*now*		
1/4	$\times 1$	$\times 1$	$\times 1/3$	=	1/12

Note that although the Markov process generates sequences according to probabilities derived from the training corpus, it generates sequences that fail to meet constraints as defined by C. The probability mass devoted to sequences which obey constraints in C is relatively small.

3.4.2 Hidden Markov Processes

A hidden Markov model is defined a five-tuple $H = (S_H, V_H, \pi_H, T_H, E_H)$ where S_H defines a finite set of (hidden) states; V_H is a finite set of observed states or symbols; $\pi_H : S_H \rightarrow [0, 1]$ represents the initial probabilities for hidden states; $T_H : S_H \times S_H \rightarrow [0, 1]$ represents the transition probabilities between hidden states; and $E : S_H \times V_H \rightarrow [0, 1]$ defines the probability of emitting a symbol given a particular hidden state. Each of π_H, T_H, and E_H represent a distribution and should thus sum to 1.0. For the training corpus above,

$S_H = \{NN, RB, VBZ, NNP\}$, $V_H = \{$*Ted, now, likes, green, Mary, red, loves, Fred, sees, sometimes*$\}$, and π_H, T_H, and E_H defined as follows:

π_H	
NN	0
VBZ	0
RB	0
NNP	1

T_H	**NN**	**VBZ**	**RB**	**NNP**
NN	0	0	0	0
VBZ	3/4	0	0	1/4
RB	0	1	0	0
NNP	0	2/5	3/5	0

E_H	*Ted*	*now*	*likes*	*green*	*Mary*	*red*	*loves*	*Fred*	*sees*	*sometimes*
NNP	1/5	0	0	0	3/5	0	0	1/5	0	0
RB	0	2/3	0	0	0	0	0	0	0	1/3
VBZ	0	0	1/2	0	0	0	1/4	0	1/4	0
NN	0	0	0	1/3	0	2/3	0	0	0	0

$O(Ln)$

$O(Ln^2)$

$B^{(0)}, \ldots, B^{(L-1)}$

Emission matrix	*Ted*	*now*	*likes*	*green*	*Mary*	*red*	*loves*	*Fred*	*sees*	*sometimes*
NNP	0	0	0	0	0	0	0	0	0	0
RB	0	0	0	0	0	0	0	0	0	0
VBZ	0	0	0	0	0	0	0	0	0	0
NN	0	0	0	0	0	2/3	0	0	0	0

To generate a sequence $s = \{(s_1, v_1), \ldots, (s_n, v_n)\}$, H follows the procedure:

1. Sample an initial hidden state $s_1 \in S_H$ according to π_H
2. Given s_1, sample a symbol $v_1 \in V_H$ according to E_H
3. While sequence not finished,
   (a) Sample the next hidden state $s_i \in S_H$ given s_{i-1} according to T_H
   (b) Sample the next symbol $v_i \in V_H$ given s_i according to E_H

The sequence finishes either when an end-of-sequence token is sampled or when the sequence has reached some pre-determined length.

Using H, a sequence $s = \{(s_1, v_1), \ldots, (s_n, v_n)\}$ is generated with probability

$$P_H(s) = \pi_H(s_1) E_H(s_1, v_1) \prod_{i=2}^{n} T_H(s_{i-1}, s_i) E_H(s_i, v_i)$$

Trained on the example corpus, H generates the following sequences with the probabilities shown:

1	$\times 2/5$	$\times 3/4$			
NNP	**VBZ**	**NN**			
Ted	*likes*	*green*			
$\times 1/5$	$\times 1/2$	$\times 1/3$	=	$1/100$	

1	$\times 3/5$	$\times 1$	$\times 1/4$		
NNP	**RB**	**VBZ**	**NNP**		
Mary	*sometimes*	*loves*	*Fred*		
$\times 3/5$	$\times 1/3$	$\times 1/4$	$\times 1/5$	=	$3/2000$

Note again that although this model will generate sequences according to the correct probability distribution, the probability mass devoted to sequences which match our desired constraints is relatively small.

It is important to note that, trained on the same corpus, the solution space of H is a superset of the solution space for M (e.g., the example solutions shown for M can be generated by H, but the reverse is not true). This increased expressivity comes purely as a result of adding the hidden layer in H. In broad terms, the solution space of M is $O(|S_M|^n)$ (where n is the length of the sequence) whereas the solution space of H is $O(|S_H|^n \times |V_H|^n)$. Shifting from a Markov to a hidden Markov model increases the size of the solution space by an exponential factor, $|V_H|^n$. This enhanced expressivity of hidden Markov models with respect to non-hidden models is one of the keys that makes the CHiMP model more robust when adding constraints: an exponentially larger solution space significantly increases the chances of maintaining satisfying solutions when pruned by constraints.

3.4.3 Constrained Markov Processes

At a high-level, a CoMP model $\tilde{M}$, derived from a Markov model M and a set of constraints C, can be thought of as a Markov model that creates a copy $T_{i,j}$ of T for each pair of sequence positions (i, j) in the sequence to be generated and then modifies each $T_{i,j}$ to ensure that constraints in C applying to positions i and/or j are met (e.g., zero out probabilities in $T_{3,4}$ transitioning to any state $s_4 \neq red$). This is demonstrated in Fig. 3.1. The CoMP model $\tilde{M}$ applies arc-consistency and re-normalization to sample constraint-satisfying solutions s with probability $P_{\tilde{M}}(s)$ that differs from $P_M(s)$ by a constant factor. Because this model is a form of tree-structured CSP, arc-consistency can be enforced in a single pass to ensure probabilities sum to 1.0 and that relative probabilities for constraint-satisfying solutions are maintained (for the details on this algorithm and the algorithm for re-normalization see [57]). Because the transition probabilities can vary by position, the CoMP model is sometimes referred to as a *non-homogeneous Markov model*.

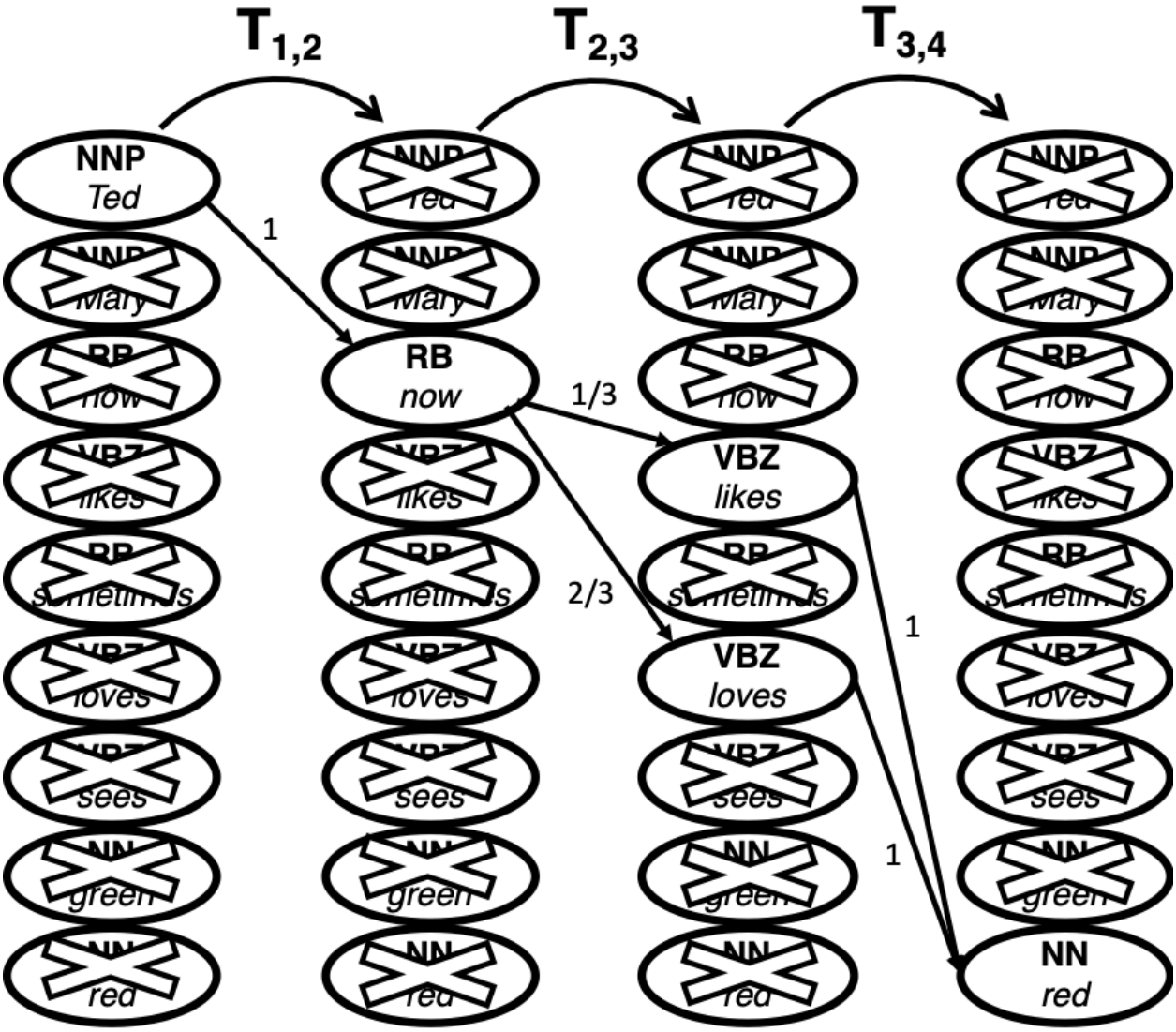


Figure 3.1: A constrained Markov process (CoMP) with constraints requiring the first token rhyme with *red* and the last token be *red*. Pruned states and updated transitions are the result of applying constraints and then enforcing arc-consistency.

Given the training corpus and constraint set C, a trained CoMP model (as shown in Fig. 3.1) is able to generate only two solutions:

Sequence s	$P_M(s)$	$P_{\tilde{M}}(s)$
Ted now likes red.	1/16	1/3
Ted now loves red.	2/16	2/3

Note that $\tilde{M}$ only generates constrain-satisfying solutions and that these solutions are generated according to probabilities derived from our training corpus. The challenge, however, is that these are the only two solutions. In general, the number of satisfying solutions for $\tilde{M}$ is very few compared to what will be shown next for the CHiMP model.

3.4.4 Constrained Hidden Markov Processes

The CHiMP model follows a pattern similar to that of the CoMP model except that instead of combining a *non-hidden* Markov model with a set of constraints C, the CHiMP model combines a *hidden* Markov model with C. Given a hidden Markov model $H = (S_H, V_H, \pi_H, T_H, E_H)$ and a set of constraints C, a constrained hidden Markov process $\tilde{H}$ generates sequences of hidden and/or observed states that obey the constraints in C. $\tilde{H}$, like $\tilde{M}$, allows for probabilities to vary by position. The difference is that CHiMP replicates and modifies both T_H *and* E_H probabilities for each position (see Fig. 5.2). After applying constraints, arc-consistency is enforced to remove nodes that do not lead to a solution. As in $\tilde{M}$, arc-consistency measures can be enforced in $\tilde{H}$ in a single pass. All matrices are re-normalized to ensure that the distributions $P_{\tilde{H}}$ and P_H differ by only a constant factor for satisfying solutions.

Given the training corpus and our rhyming constraints, the CHiMP model is able to generate the following 12 solutions, each with their respective probabilities.

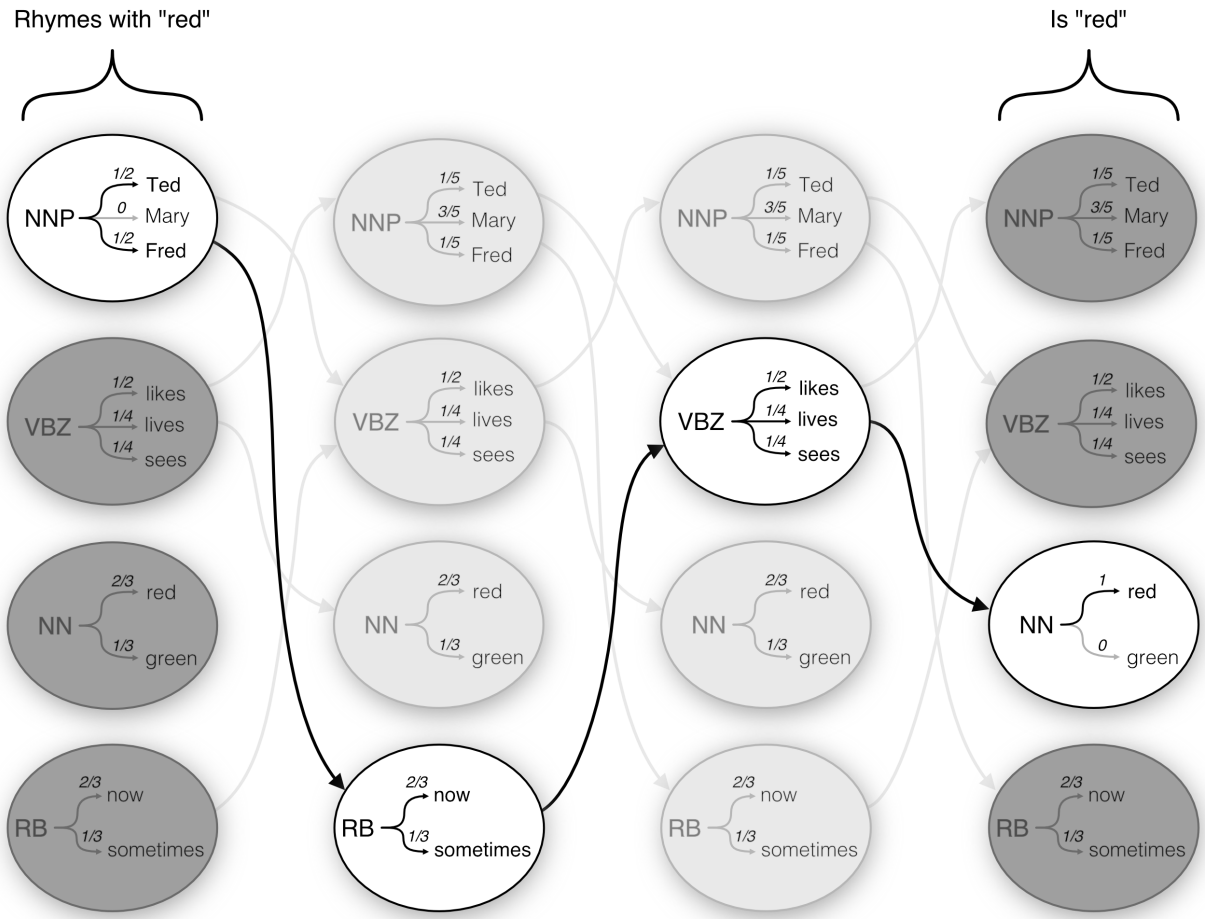


Figure 3.2: A high-level schematic of a constrained hidden Markov process (CHiMP) of length 4 constrained so that the last word is "red" and the first word rhymes with "red". Each column represents a position in the sequence to be generated. Each node represents a hidden state (i.e., part-of-speech) and a probability distribution for the observed states (i.e., words) that can be generated from that hidden state. By pruning observed states that are disallowed by constraints and then adjusting probabilities to maintain arc-consistency, the resulting model generates constraint-satisfying solutions with probability relative to the original probability distribution. Hidden states pruned directly from applying constraints are indicated by dark grey nodes and states pruned during arc-consistency are indicated by light grey nodes.

Solutions	Probabilities
Ted now likes red.	(4/200)
Ted now loves red.	(2/200)
Ted now sees red.	(2/200)
Ted sometimes likes red.	(2/200)
Ted sometimes loves red.	(1/200)
Ted sometimes sees red.	(1/200)
Fred now likes red.	(4/200)
Fred now loves red.	(2/200)
Fred now sees red.	(2/200)
Fred sometimes likes red.	(2/200)
Fred sometimes loves red.	(1/200)
Fred sometimes sees red.	(1/200)

By including hidden states in the constrained model $\tilde{H}$, the increased solution space results in the model being able to generate *significantly* more satisfying solutions than $\tilde{M}$. A second added benefit is afforded by the addition of hidden states: because constraints are applied on observed states and because there is no direct influence between observed states in H, there is an increased degree of separation between constraints. This helps to avoid the compounding effect of diminished state spaces resulting from individual constraints.

3.5 Applications

To demonstrate the improvements of the CHiMP model over the CoMP model, we compared the results of each model trained on the Corpus of Contemporary American English (COCA) [24] and provided the same set of constraints. In particular, we selected training sets from the 2012 fiction portion of COCA and constrained each model to only output sequences in

which the first letter of each word began with the same letter (e.g., a tongue-twister). We chose this problem because it represents a fairly general example of constrained sequence generation that is easily adapted to sequences of varying lengths. Results are averaged over 26 instances of the problem with each instance having constraints defined with a different letter of the English alphabet.

Our experiments were two-fold. First, we examined the number of sequences generated by each model as a function of sequence length. Since each word is constrained to start with a specified letter, as the sentence length increases, so does the number of constraints. This experiment compares how the models are affected by increasingly stringent constraints. Second, we examined the number of sequences generated by each model as a function of the size of the training corpus. Our motivation here is to compare how the models are affected by an increasingly restricted data set. In all cases, both models were trained on the same subset of data.

In Fig. 3.3, we see that as the sentence length increases and, by consequence of our problem, the constraints become more numerous, the CoMP model generate less and less unique sentences – even unable to generate sentences past a sentence length of 6 (for a fixed corpus size of 100 sentences). Each constraint added further tightens the restrictions put on possible solutions that CoMP can find. In contrast, CHiMP maintains expressivity (i.e., the ability to generate unique sentences) as sentence length grows and constraints are added.

In Fig. 3.4, CHiMP remains expressive even when trained on a very small training corpus (25 sentences). The trend of unique sentence averages increases as the training set increases until reaching the maximum number of sampled solutions (100K). CoMP's ability to generate novel and unique sentences is severely limited according to the size of its training set.

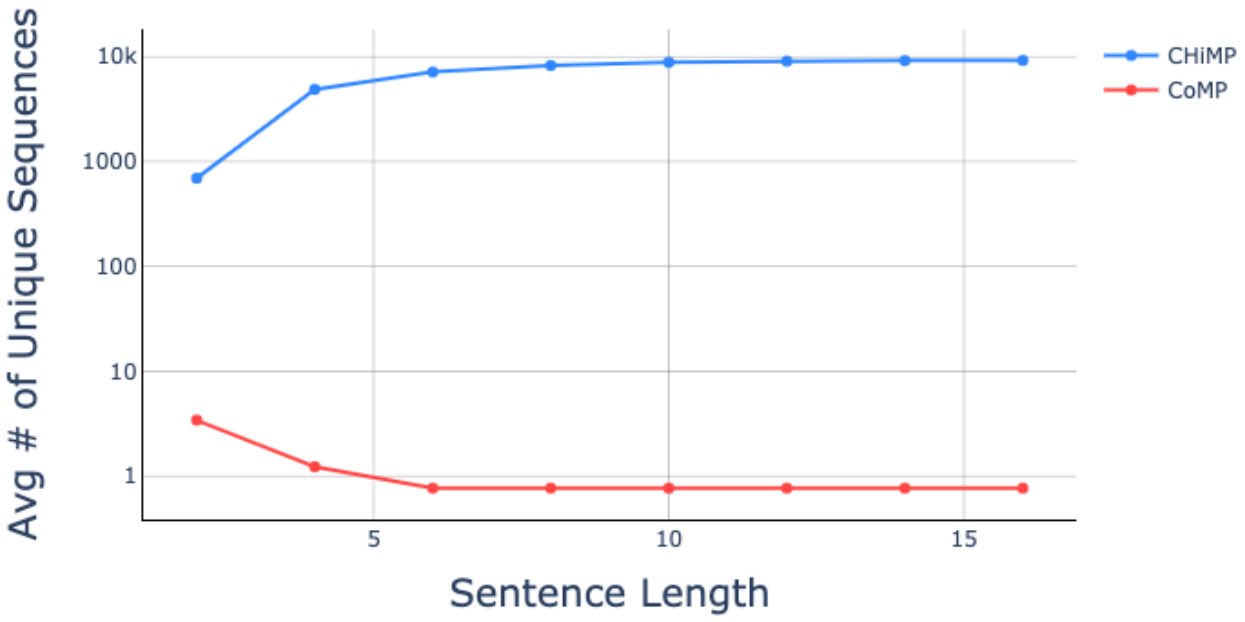


Figure 3.3: The effects of sequence length (and consequently number of constraints) on generalizability (i.e., number of unique sequences out of 10k sampled solutions) for a fixed random training set of 100 sentences. Each model is constrained such that words start with the same letter, and counts are averaged over 26 runs (a different letter constraint for each run). The added constraints from increasing sequence length have a compounding limiting effect in the CoMP model, whereas the abstraction of the CHiMP model serves to decouple constraints to avoid bottlenecks.

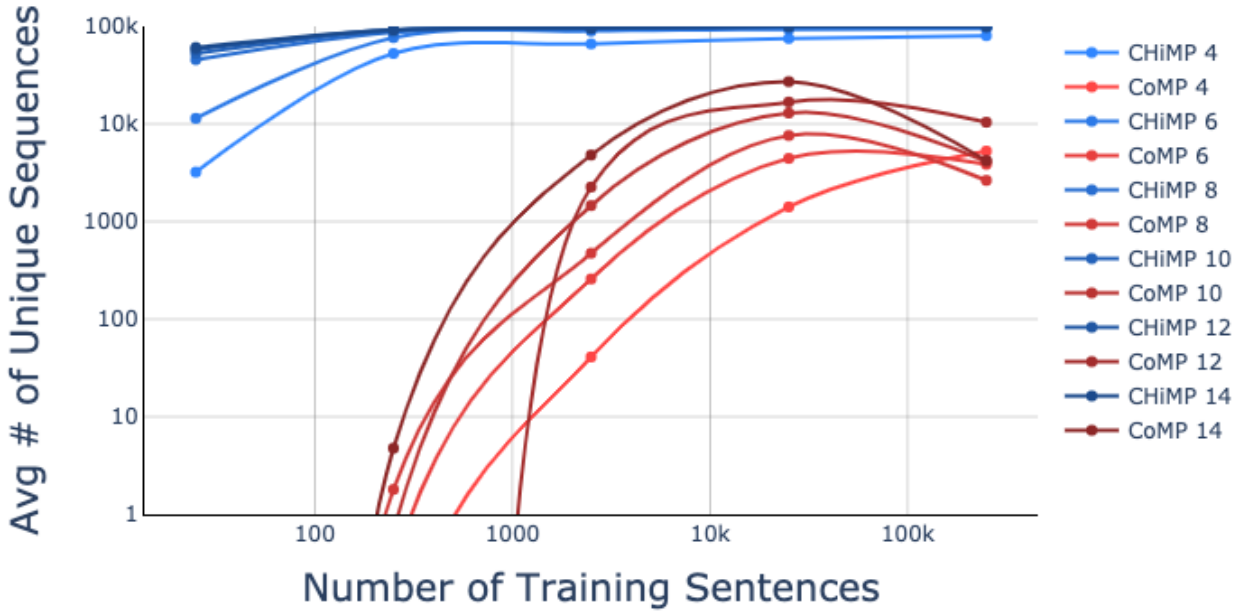


Figure 3.4: The effects of training corpus size on generalizability of the CHiMP (blue) and CoMP (orange) models. Generalizability is measured as number of unique sequences out of 100k sampled solutions. Each model is constrained such that words start with the same letter, and counts are averaged over 26 runs (a different letter constraint for each run). Shades show the effects of varying the sequence length (and consequently the number of constraints) on generalizability. The CHiMP model consistently generates more unique satisfying solutions than the CoMP model and is relatively immune to the effects of training set size or number of constraints.

3.6 Conclusion

We have demonstrated through quantitative analysis that under increasingly stringent constraints and increasingly limited training data, constrained hidden Markov processes are an ideal solution for maintaining reasonable solution spaces. It remains to be seen what *qualitative* affect the CHiMP model has compared to the CoMP model in terms of the *coherence* of generated sequences. Regardless, insofar as quality is reflected in the ability of the model to add and respect more constraints, CHiMP maintains a significant advantage.

The diminished fear of adding constraints afforded by the CHiMP model is a significant asset that opens many new and exciting avenues for research in sequence generation. Where many problems are easily defined using constraints, the difficulty of generating structured sequences shifts from the inability of the model to find solutions to challenging the system designer to think of new and creative ways to derive and define constraints.

Chapter 4

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

The paper provided in this chapter provides an introduction to constrained models and how they might be used in computational creativity. This paper was published at the International Conference on Computational Creativity (ICCC) in 2019 [11]. I was involved with the creation of the constrained models, writing and presenting the published work at the ICCC conference, and with creating interactive tools that were used to create the mnemonics presented in the paper.

4.1 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

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

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.

4.2 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" [20]. 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 [37], 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 [22] (e.g., see [49]. Related is the model of a dynamic knowledge base [63] 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 [8]). Generate-and-check is another model that has been suggested as being representative of the creative process [61].

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. [22] 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. [72] 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 4.1). Margaret Boden [8], 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." [79] elaborates a formal mechanism of Boden's concept of creativity by defining two rule sets, $\mathscr{R}$ and $\mathscr{T}$. Of 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. [69], 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., [77]). 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

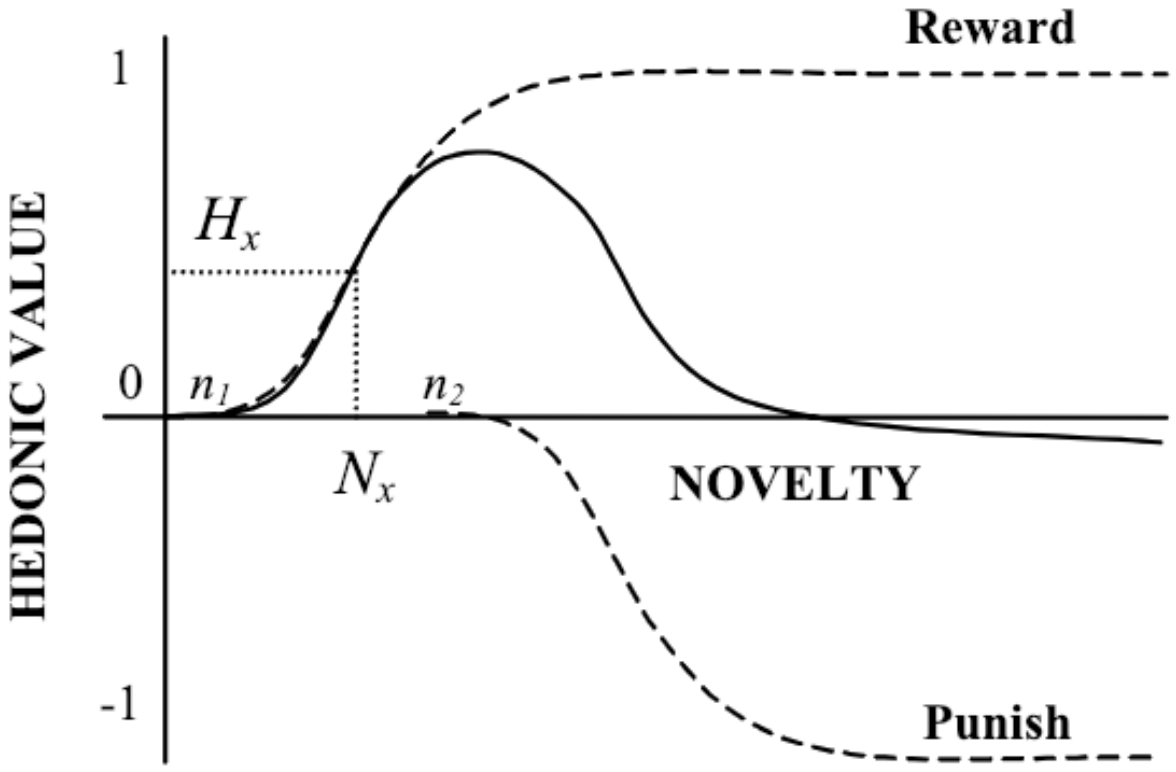


Figure 4.1: The Wundt curve models value as the sum of two nonlinear functions: H_x which rewards novelty, and N_x which punishes novelty beyond some threshold of typicality, from [72].

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).

4.3 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 [39].

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 4.2). 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 [54].

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 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

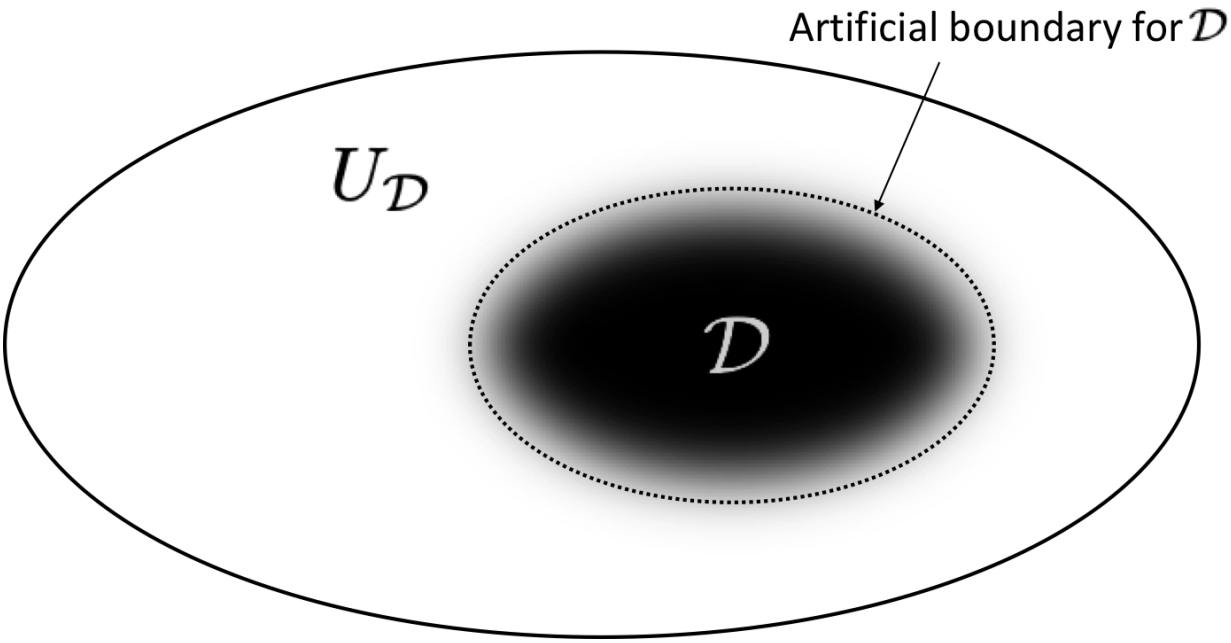


Figure 4.2: In many forms of creativity, the set of domain artefacts $\mathcal{D}$ exists as a structured subset of a larger domain $U_{\mathcal{D}}$ of all artefacts that can be represented using the same language as is used to describe artefacts in $\mathcal{D}$. Due to the inherent difficulty of defining 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.

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 [61]. This solution not only creates inefficiencies, but often assigns low probability to artefacts belonging to $\mathcal{D}$ [77]. In such cases it becomes improbable that the system generates valid artefacts in reasonable time [57].

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" [54].

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 [62]; MDDs that enforce constraints on non-discretized temporal sequences [71]; factor graphs for imposing constraints represented as regular languages [59]; and non-homogeneous Markov models [57]. 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) [58]. 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, [62] 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."

4.3.1 Quality Assurance

We have discussed how constrained probabilistic models are well-suited for explicitly modeling typicality and novelty, but what about quality? As [8] 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 [9]).

4.4 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 [57].

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 4.3b). 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 $P_{\mathcal{N}}(s) = P_{\mathcal{M}}(s|s \in S_C)$ [57].

NHMMs have been applied to model music generation, generating melodies constrained to begin and end on the same note [57]. [7] apply NHMMs to generate lyrics matching rhyme, syllable stress, part-of-speech, and semantic constraints.

4.5 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 4.3). 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.

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

[2] `https://github.com/po-gl/ConstrainedMarkovModel`
[3] `https://github.com/norkish/downbythebay/tree/master/DownByTheBay/src/dbtb/markov`

4.7 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 [24]. 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 4.6 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]Available via `https://spacefem.com/mnemonics`

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

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 4.4 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 4.5 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 4.6 shows seven mnemonic device tasks together with the highest-rated mnemonic devices (as per average memory score) generated by NhMMonic for the task.

4.8 Discussion

Survey results demonstrate that increased grammatical/semantic cohesion afforded by probabilistic Markov models are associated with gains in ease of memorization. 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 [53] 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 varying the training data. We leave these as exploratory ideas for future work.

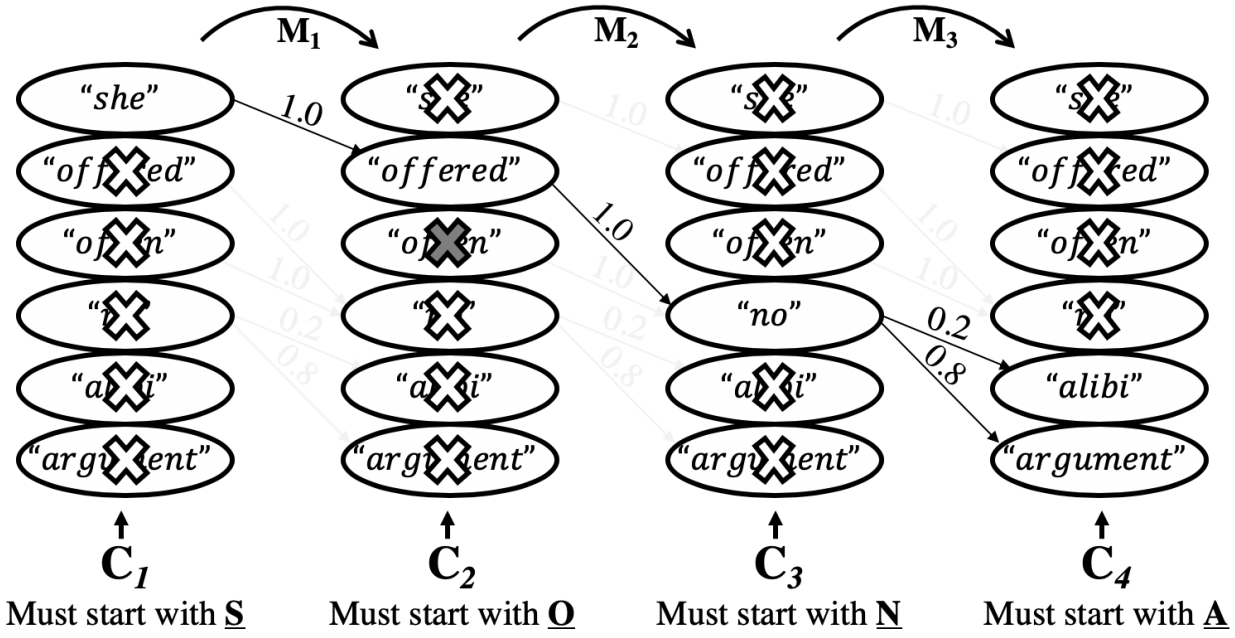
Many forms of creativity have relational structure (e.g., rhyme schemes, repeated motifs, etc.). Unlike the example we have shown here which uses solely unary constraints, relational structure is most effectively realized using binary constraints. Sampling from constrained Markov models with binary constraints is known to be a much harder problem (see [70]), however recent work has been done towards providing reasonable solutions [59, 71]. 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.

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 4.3: *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.

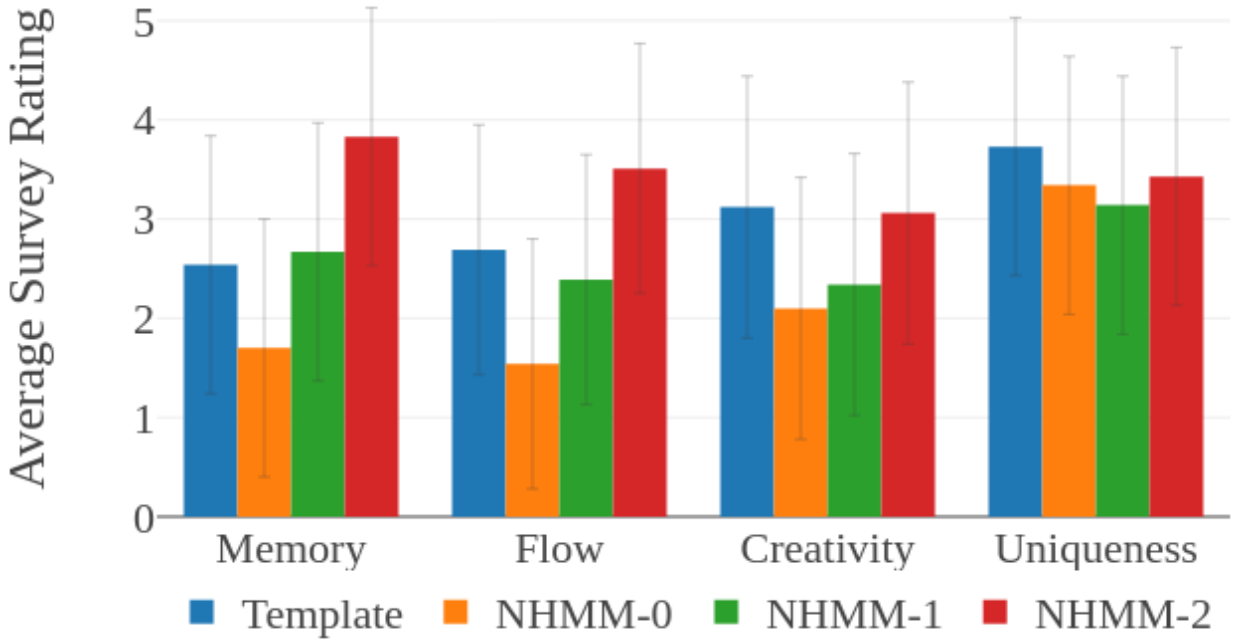


Figure 4.4: *Survey Results.* Average ratings from 320 evaluations across four metrics for four different mnemonic device generation algorithms. Error bars are standard deviation. The ease of memorization of mnemonics from the NHMM-2 model appears to be associated with improved flow with respect to other models.

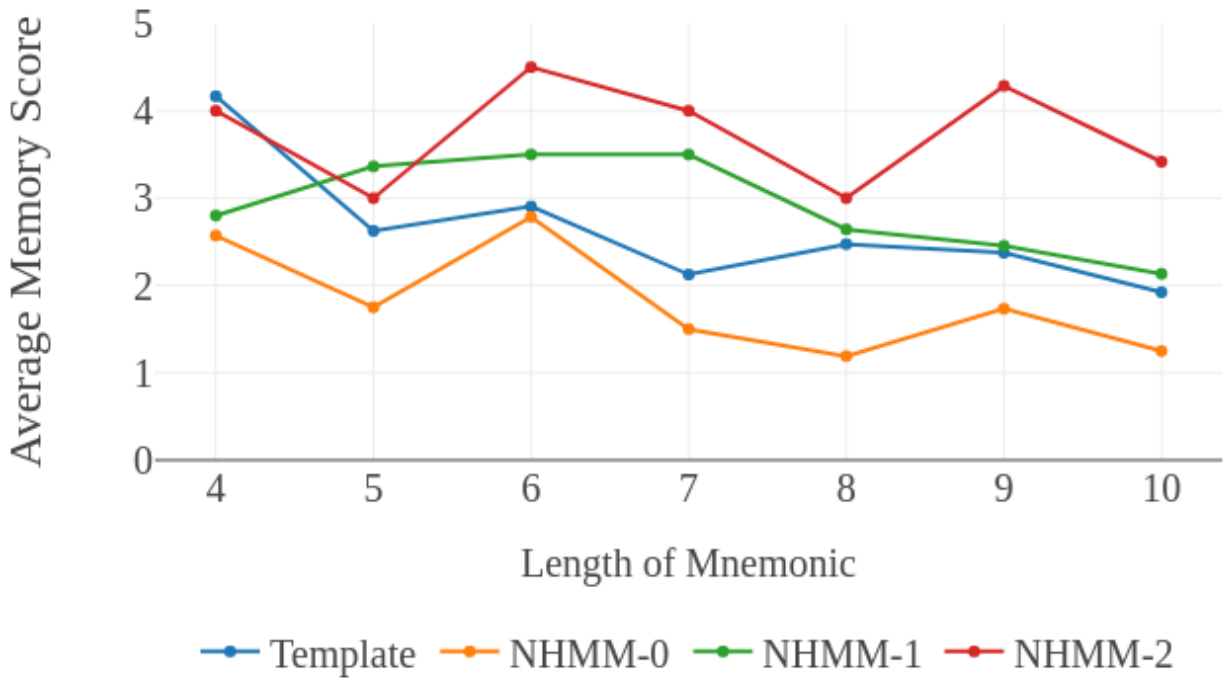


Figure 4.5: *Impact of Task Length.* As the length of the memorization task increases, the effectiveness of mnemonic devices decreases across all models, but at a much lesser rate for the NHMM-1 and NHMM-2 models. We hypothesize that this is owing to the sustained grammatical and semantic flow that these models achieve from the constrained Markov model.

Four Stages of Enlightenment: Stream-enterer, Once-returner, Non-returner, Arahant

"She offered no argument" (NHMM-2, 5.0)

Dante's 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, Molecul

"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 4.6: *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).

Chapter 5

A Leap of Creativity: From Systems that Generalize to Systems that Filter

In this chapter, the provided published paper provides an insight on how the CHiMP model provides a larger solution space than previous models, such as CoMP. This work was published at the International Conference on Computational Creativity (ICCC) in 2020 [29]. I contributed to this work by creating an accessible framework for applying constraints to the constrained hidden Markov process (CHiMP) as well as formalizing the model in such a way that it was accessible to other collaborators.

5.1 Abstract

In his work "Mere Generation: Essential Barometer or Dated Concept?", Ventura [76] categorizes creative processes along a spectrum of increasing creativity. While the spectrum provides insight into the dimensions through which creativity can be augmented, it does not of itself provide insights into how to advance a system through these dimensions. In this paper, we present some theoretical and practical insights on advancing along one commonly problematic rung of this ladder, namely from a system that exhibits *generalization* (i.e., the ability to generalize beyond an inspiring set) to a system that exhibits *filtration* (i.e., the ability to self-evaluate and filter results). One potential challenge in this transition is that filtration requires having a sufficiently large number of solutions to filter from the generalizing model. We propose that one solution to this problem is achieved not through increasing the size of the inspiring set (an obvious solution that brings additional problems), but rather through amplifying the generalization of the system to produce a greater set of novel

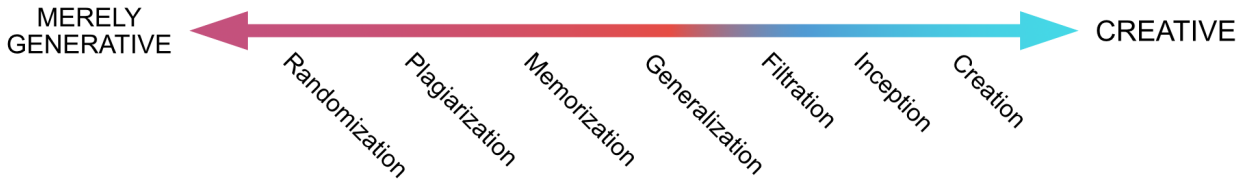


Figure 5.1: Ventura's [76] spectrum of creative systems provides a means by which to measure the progress of a system towards becoming creative. Characterizing challenges and solutions that are specific to each level in the spectrum helps to actualize the spectrum into becoming a guide for building more creative systems.

artefacts to filter. We compare a new version of a system, NhMMonic, for generating creative mnemonic devices with a new conceptualization model that allows greater generalization. We demonstrate how filtration, which was not possible in the early version of NhMMonic, only becomes feasible with the more generalizable model.

5.2 Introduction

The field of Computational Creativity (CC) has been supported in its quest by several significant contributions in the domain of CC theory. One such contribution exists in Ventura's spectrum of creative systems [76]. This spectrum suggests that there exist at least seven different levels along the path towards computational creativity including levels such as randomness, memorization, generalization, and filtration (see Figure 5.1). Ventura asserts that along this spectrum, real computational creativity starts at least as early as generalization with filtration representing perhaps a conservative threshold.

While this spectrum is useful for measuring the progress of applied CC systems, it leaves two important questions unanswered:

1. *For each level of the spectrum, what challenges are CC systems likely to encounter?*
2. *What suggestions can be made to overcome those challenges?*

Answers to these questions would provide a way to actualize the spectrum into a guide for augmenting the creativity of computational systems.

Our motivation in considering these issues came about in the context of our previous work using constrained Markov models to generate mnemonic devices [10]. Markov models are an example of a generalizing model. The application of constraints to Markov models represents the act of filtration. In applying constraints to generate mnemonic devices, it frequently occurred that no satisfying solutions could be found.

The purpose of this paper is to provide answers to two questions stated above with specific regard to systems that have achieved the level of *generalization* and are attempting to make the "leap" to the level of *filtration*. This step is of interest as it marks the transition from a budding creative system to an intentionally creative system. This leap is significant in light of the fact that of the last four levels of the spectrum—where true creativity is said to emerge—this is the first step.

Generalization systems produce artefacts using an internal *conceptualization*—a model which embodies an understanding of a domain and allows for the creation of artefacts that belong to the domain [77]. Examples of conceptualizations include using long short-term memory models for music generation [51], neural networks for visual art [52], and Markov processes for music and text generation [7, 57].

One particular challenge we have repeatedly observed in the development of CC systems at this level is the challenge of dealing with diminishing solution spaces. This problem arises commonly when attempts are first made to add filtration to a generalization system because filtration by definition implies the reduction of a system's solution space. The purpose of the filtration step is to equip the system with self-evaluative capabilities for restricting the artefacts it generates based on measurements of fitness. However, a well-known trade-off arises: stricter filtering leads to better, but fewer results. In some cases the results are so few that it becomes difficult to justify that the system is capable of generating anything, let alone artefacts that are novel. How can systems overcome this challenge?

A simple solution for increasing the solution space is to simply increase the size of the inspiring set. For many conceptualizations of CC systems this alone will increase the overall throughput of the system, and often increases the generalizability of the system as well. However, for most domains, finding a larger inspiring set ranges from being impractical to an impossibility. What more practical solutions exist?

We propose and illustrate through example how increasing the generalizability of a generalization system through abstraction and regularization can increase the solution space without requiring a larger inspiring set. Well-known methods exist for generalization of most conceptualization models used for CC systems, including L1 and L2 regularization for neural networks, shortening the Markov window length in Markov processes, generalizing the fitness function for genetic algorithms, and abstracting rules in rule-based systems. Through regularization and abstraction, a system is able to better leverage the knowledge in an inspiring set in order to increase the solution space.

In demonstrating the impacts of abstraction and generalization, we comparatively consider the performance of two models: a less abstract model (CoMP) and a more abstract model (CHiMP). We assess the ability of each model to intentionally produce *novel* artefacts. We choose to focus explicitly on the creative attribute of *novelty*—setting aside the attributes of value and intentionality—inasmuch as it is the attribute of creativity most directly relevant to our discussion [69, 76]. We discuss the impacts of generalization on value in the discussion section below.

5.3 Methods

NhMMonic [10], is a CC system designed to generate mnemonic devices. At its heart, NhMMonic uses a constrained Markov process (CoMP) for its conceptualization model. This constrained Markov process allows for the combination of a (non-hidden) Markov process (e.g., trained on words) and a set of unary constraints (e.g., word-starts-with constraints) such that the model is able to generate constraint-satisfying sequences according to Markovian

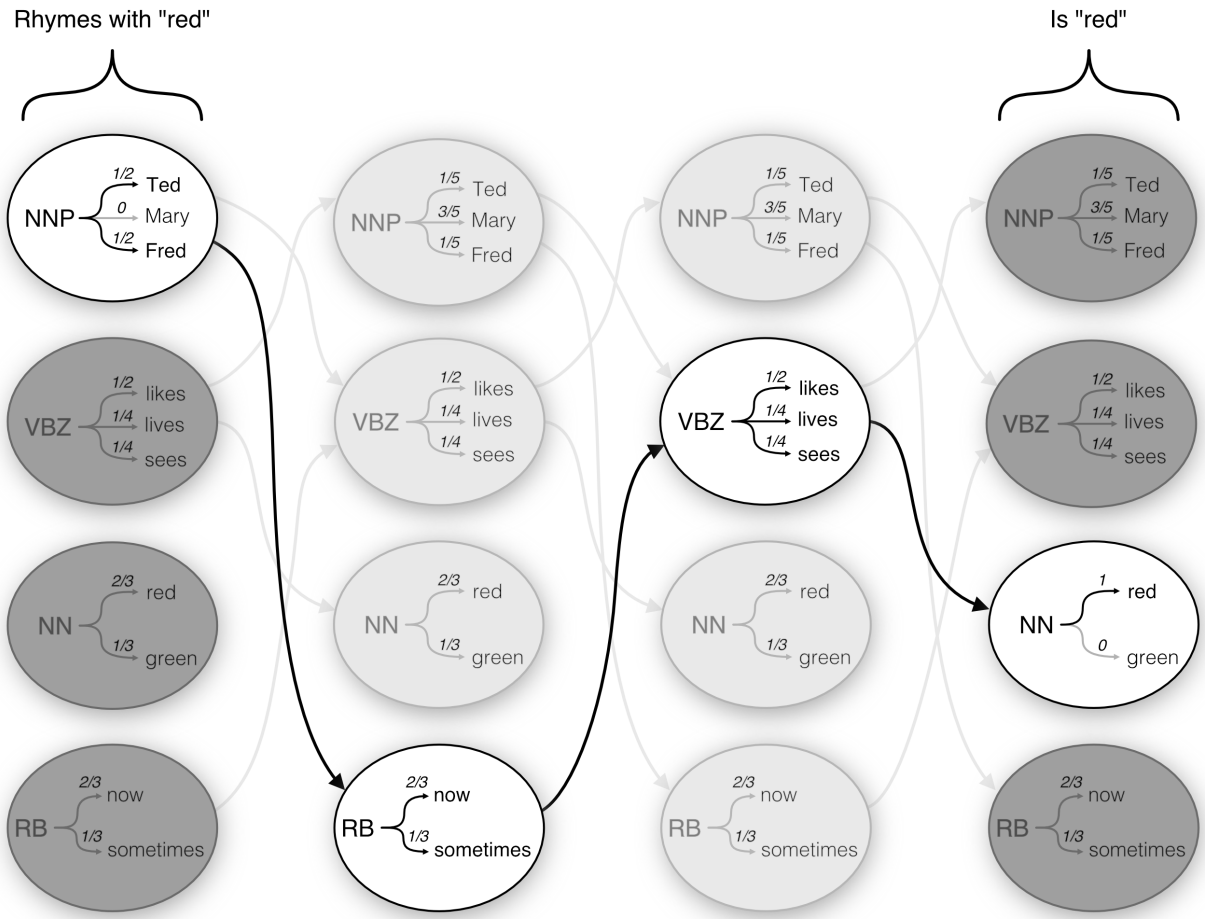


Figure 5.2: A high-level schematic of a constrained hidden Markov process (CHiMP) of length 4 constrained so that the last word is "red" and the first word rhymes with "red". Each column represents a position in the sequence to be generated. Each node represents a hidden state (i.e., part-of-speech) and a probability distribution for the observed states (i.e., words) that can be generated from that hidden state. By pruning observed states that are disallowed by constraints and then adjusting probabilities to maintain arc-consistency, the resulting model generates constraint-satisfying solutions with probability relative to the original probability distribution [30]. Hidden states pruned directly from applying constraints are indicated by dark grey nodes and states pruned during arc-consistency are indicated by light grey nodes.

probabilities [57]. In previous work we demonstrated through qualitative surveys the strength of this model (particularly at higher Markov orders) for generating effective mnemonic devices. A byproduct of our analysis revealed that for many mnemonic device problems, the addition of constraints (i.e., filtering) resulted in NhMMonic being incapable of finding satisfying solutions despite being trained from relatively large inspiring sets.

A known method for increasing the generalization of Markov models is through the introducing of an abstract hidden layer resulting in a model known as a *hidden* Markov process. Direct dependencies between adjacent observed sequence elements are dissolved in the hidden Markov process, allowing for greater decoupling between sequence elements. This generally results in hidden Markov processes having significantly higher expressivity with respect to their *non-hidden* counterparts.

To combat the challenges facing NhMMonic with respect to a diminishing solution space, we designed a new conceptualization model for the system that combines hidden Markov processes with constraints in much the same way that constrained Markov processes combined *non-hidden* Markov processes with constraints [30]. The resulting model is called a constrained Hidden Markov process (CHiMP) which is visualized in Figure 5.2. The CHiMP model was chosen under the hypothesis that increased abstraction, resulting in increased generalization, would lead to a significantly larger solution space.

In implementing a filtration system, it is apparent that a large solution space is needed. Using two hypothetical models A and B (seen in Figure 5.3) we illustrate the restriction that solution space imposes on a system's ability to step from a generalization system to a filtration system. Model A fails to have a solution space after filtering and thus remains a conceptualization for a generalization system. Model B, however, has a larger beginning solution space β due to an increase in the model's ability to generalize the inspiring set. Thus model B has a usable solution space β' after filtering and can be categorized as a filtration system.

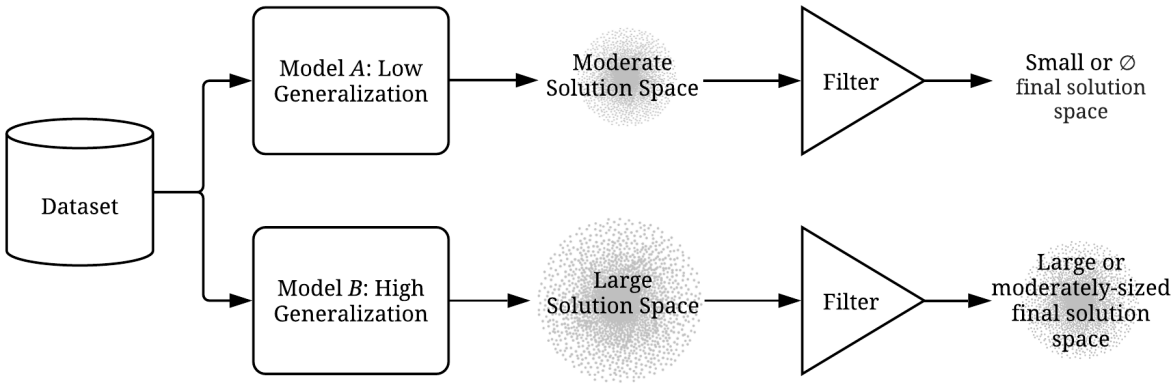


Figure 5.3: The application of filters on two hypothetical models (A and B) demonstrates the requirement for larger solution spaces (increased generalization) in order to endure filtering with a usable solution space. Model B has a usable solution space after filtering; thus the model has moved further along in the spectrum from generalization to filtration.

5.4 Results

In demonstrating the increased generalization (and hence increased solution space) of CHiMP over CoMP, we compared the results of each model trained on the Corpus of Contemporary American English (COCA) [24] and provided the same set of constraints. In particular, we selected training sets from the 2012 fiction portion of COCA and constrained each model to only output sequences in which the first letter of each word began with the same letter (e.g., a tongue-twister). We chose this problem because it represents a fairly general example of constrained sequence generation that is easily adapted to sequences of varying lengths. Results are averaged over 26 instances of the problem with each instance having constraints defined with a different letter of the English alphabet.

Some qualitative results are shown in Figure 5.4. It should be noted that within the subset of 40 sequences generated by CHiMP, no duplicate or similar solutions where present; whereas 6 sequences were duplicates (or very similar) in the subset generated by CoMP.

We examined the effect of changing the sentence/model length on the novelty of the system in terms of the total number of unique solutions capable of being generated by each model (see Figure 5.5). As the sentence length increases, so too do the number of constraints

on the sequence to be generated. In the abstracted CHiMP model, this is inconsequential; the model can afford to make restrictions at the observed node that do not affect transitions between sequence positions (which are isolated in the hidden layer). Only occasionally do a sufficient number of pruned states combine to require the pruning of a hidden state node, but such is a relatively rare occurrence.

By contrast, the effects of increased sentence length on the CoMP model are severely limiting. Each added position would typically add a number of novel unique solutions *if it did not come with the addition of a new constraint.* The newly constrained position has direct influence on previous observed sequence states and thus pruning values from the domain of these variables directly results in the removal of transitions between adjacent sequence positions. This results in a relatively slow growth in the solution space as sentence length grows.

The increase in the CHiMP model appears to be exponential owing to the multiplicative effect achieved by maintaining large domains for adjacent variables in the hidden layer.

Similar trends in the impact on novelty are manifest when we vary the training set size, keeping sentence length constant (see Figure 5.6). We see that the size of the solution space for the CHiMP model increases exponentially. The CoMP model also appears to have some slightly exponential growth, but at a significantly lower rate. This is again what we would expect to see. Increasing the training set size (when such is a possibility) still has a more significant impact on CHiMP than on CoMP model.

The results shown in Figures 5.5 and 5.6 suggest that CHiMP, with respect to CoMP, facilitates exponentially more novelty. The solution space of the CoMP model is by definition a subset of the solution space of the CHiMP model, and for most training and constraint sets will be a substantially smaller subset. It is expected that of the novel results produced by CHiMP, some will have higher value than the solutions shared by both models. Because the CHiMP model abstracts to a more significant degree from the training set than the CoMP model, we might expect a greater portion of the novel solutions to be of lower value. The

CoMP Tongue Twisters:

late last light levels like lady
Diaz did dinosaurs died dell drove
max mowed my mother made my
language lessons last look little lamb

CHiMP Tongue Twisters:

queen Quanhe quite quiet queasy qualified
flower facing forward for from forester
free feeling facing followed free fate
every educated Elizabeth expected Erika enchanting

Figure 5.4: Example results from generating 6-length tongue twisters (i.e., alliterative constraints) from both the CoMP and CHiMP models. Both models were trained on 10K sentences. Results are chosen from a randomly selected subset of 40 sequences from each model. The quality of tongue twisters is roughly equivalent between both models (both poor), but the CHiMP model is capable of generating exponentially more solutions. This suggests that increasing the Markov order in the CHiMP model (as an example of more stringent constraints) will have far less deleterious affects on the solution space as compared to a similar increase in the CoMP model.

suggestion from qualitative results shown in Figure 5.4 is that there is no obvious degradation of value. However, we do not currently have results to fully assess the extent to which value degrades (or doesn't). In any case the expressivity of the CHiMP model enables a simple solution: introduce new or stricter filtering by increasing the number and stringency of constraints.

5.5 Discussion and Conclusion

In progressing from a generalizing system to a filtration system, our results provide meaningful insight into two important questions relating to Ventura's spectrum of creative systems:

1. *For the filtration level of the spectrum, what challenges are CC systems likely to encounter?*
2. *What suggestions can be made to overcome these challenges?*

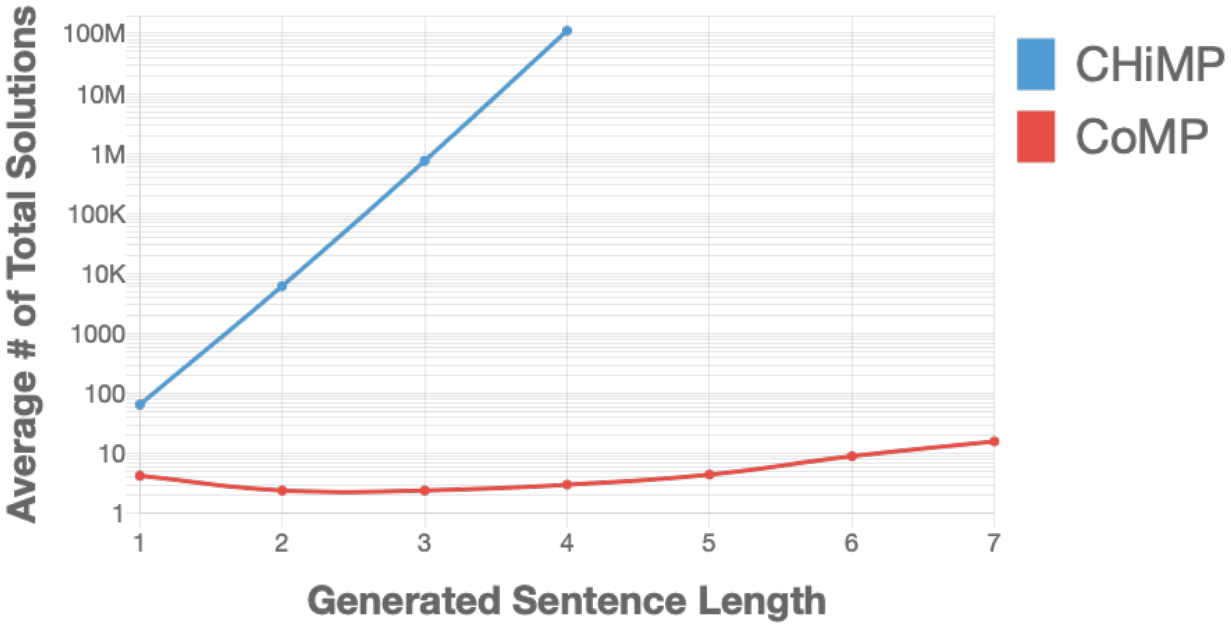


Figure 5.5: The effects of sequence length on the number of total solutions generated by each model with a fixed training set size of 300 sentences. Both models are constrained such that each word in a sequence starts with the same letter; counts of total solutions are averaged over 26 runs (each run using a different letter from the English alphabet). We see that as the sequence length increases, total solutions for the CHiMP model increases exponentially (given the logarithmic scale) whereas the CoMP model stagnates.

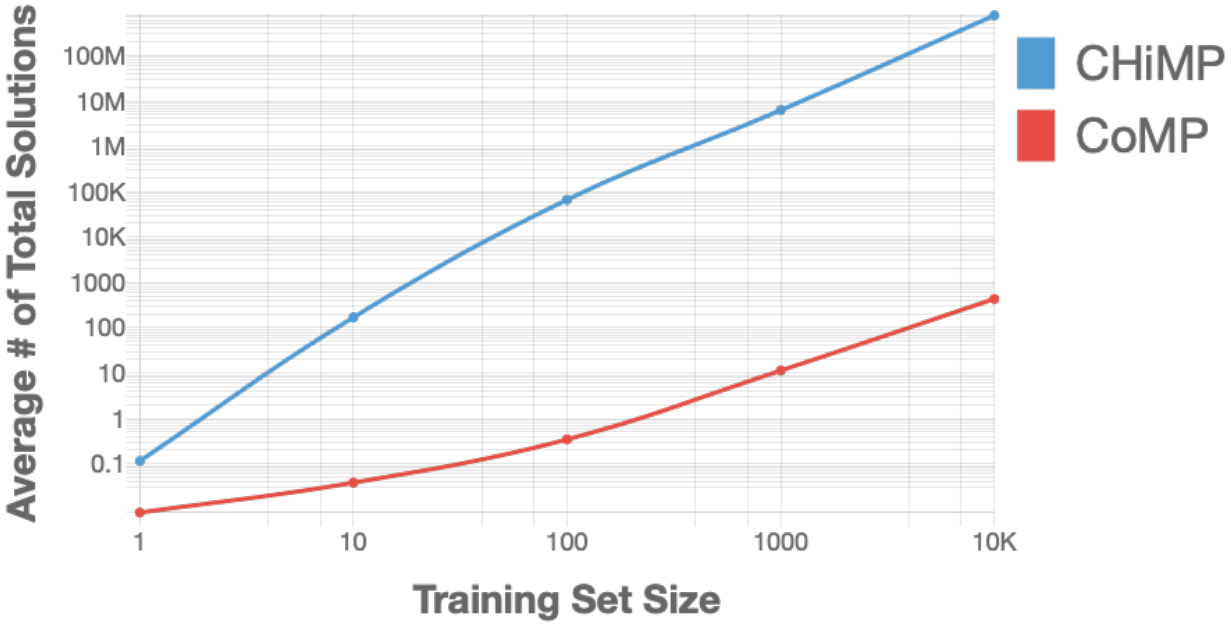


Figure 5.6: The effects of training corpus size (number of training sentences) on the number of total solutions generated by each model with a fixed sequence length of 3. Both models are constrained such that each word in a sequence starts with the same letter; counts of total solutions are averaged over 26 runs (each run using a different letter from the English alphabet). The total solutions of both models increase in an almost parallel way; however, at 10K training sentences, CHiMP well exceeds 100M total solutions which contrasts CoMP at 1000 total solutions.

A significant challenge for CC systems attempting to transition to a filtration system is as more constraints (or filters) are put on the system, the solution space diminishes to the point of being too small to filter. As demonstrated in the CoMP model (Figure 5.5), the insufficient solution space prevents being able to apply more constraints and filters to produce higher quality artefacts.

The problem is not specific to our results or to Markov models. Filtering, by nature, reduces the solution space. As shown in Figure 5.3, any CC system with low generalization may fail to have a usable solution space after filtering.

Greater generalization can address the aforementioned problem. We see from our results that our model with greater generalization, CHiMP, excels in solution space size even as constraints are added (see Figure 5.5). The primary difference between CoMP and CHiMP is an added layer of abstraction in CHiMP that affords greater generalization. The solution to a diminished solution space is to increase the level of abstraction in the model. This increases the generalization ability of the model and results in a solution space substantial enough to "survive" filtering.

Increased constraints allow for greater creativity and quality because the system can use constraints to explicitly articulate and enforce the system's goals and intentions. For example, in Markov models, increasing the Markov order (a form of adding more constraints) significantly improves the coherency of natural language, but the solution space is heavily diminished. With the CHiMP model, the solution space is sufficiently enlarged to avoid these devastating consequences to the solution space. Besides changes to the Markov order, other possibilities open up for using constraints to filter results to further improve quality, including semantic constraints, structural constraints, and even more complex $n-ary$ constraints. It is also often the case that constraints can be easily described in human-interpretable language, enabling the system to provide framing for its creative behavior, contributing to an increased perception of creativity in CC systems [19].

It is important to acknowledge the negative consequences of increasing the generalization in a learning model. In particular, generalization decouples dependencies between variables which can result in a loss of information during variable assignment. For example, generalizing to a hidden Markov model takes a significant toll on language coherence. In short, the novelty achieved by generalization comes with a trade-off in value. We hypothesize that this deterioration can be offset in the application of filters to preserve the information lost. We plan to examine this issue in future work.

Through developing a system (CHiMP) that more effectively achieves filtration, we have discovered insights into the challenges present in the leap from *generalization* to *filtration* and how to overcome them. The challenge of diminishing solution spaces can be overcome by amplifying the generalizing ability of the system through abstraction. Having realized the leap from generalization to filtration, the community is now poised to address the challenge of making the subsequent leaps along Ventura's spectrum of creative systems, advancing past filtration into *inception* and ultimately *creation*.

Chapter 6

Phrasal Category Tagging

This chapter constitutes the majority of the work that is unique to this thesis. We plan to submit a modified version of this chapter for publication and presentation in proceedings of AAAI 2023.

6.1 Introduction

The work presented in Chapter 3 and Chapter 4 demonstrated the CHiMP model being used with words as observed states and parts of speech as hidden states. They used the CHiMP model to generate sequences based on lexical categories as the hidden states and words as the observed states. As discussed in Chapter 5, the abstraction of the CHiMP model has the adverse side effect of diminishing the coherence of the model with respect to the non-hidden CoMP model. Thus the problem is presented: How can we mitigate the negative impact of diminished coherence while maintaining the expressive capabilities? In this chapter we will explore expanding the hidden state space to phrasal categories as a substitute to lexical categories. Phrasal categories consist of sentences phrases such as a noun phrases (NP) or verb phrases (VP) [28]. If the CHiMP model is used to generate sequences of phrasal categories (rather than simply words) from a set of hidden state space, can we increase the coherence while maintaining the expressive capabilities?

In the context of the English language, a phrasal category is one or more words that form a grammatical unit [28] [13]. There are 5 primary phrases: *noun phrase, verb phrase, adjective phrase, adverb phrase, and prepositional phrase* [1]. By using phrasal categories

as the hidden state space rather than the lexical categories, we can apply constraints to a specific word in the phrase or we can use a floating constraint where the constraint is applied to any word in the phrase. For example, when looking at the phrasal categories of the sentence "John likes the blue house at the end of the street." there are several phrases that could be selected for the phrasal categories. The possible noun phrases are "John", "the blue house", "the end", "the street", "the end of the street". Floating constraints present the possibility for the phrase to contain specific words in any position, a certain number of words, a certain number of syllables, etc. If we were to continue to use just the lexical categories, very explicit constraints would have to be applied to each position to ensure the a phrase that meets the constraints is generated. Additionally, phrasal categories maintain some level of cohesion as we are often evaluating multiple words together. This allows for the model to use data that is seen as more natural rather than each word being pieced together based on the statistical probability of them occurring in the training data.

Figure 6.1 shows the parse tree of the sentence "John likes the blue house at the end of the street". This parse tree parses the sentence into each individual phrasal category, then into each lexical category that makes up the phrasal category, and then shows each word that makes up the lexical category. All English sentences can be parsed like this, and this is how we evaluate each sentence for our model. It's important to note that a phrasal category can be comprised of other phrasal categories. Figure 6.1 also shows this hierarchy as some phrasal categories branch out into other phrasal categories.

6.2 Methods

To train the extended CHiMP model on phrasal categories, a parse tree of each sentence is created. Figure 6.1 shows an example parse tree on the sentence "John likes the blue house at the end of the street". Table 6.1 shows what each phrasal category acronym represents. Each phrasal category transition was counted and added to the δ transition matrix and the corresponding output of words was added to the emission probability ϵ matrices. As Figure

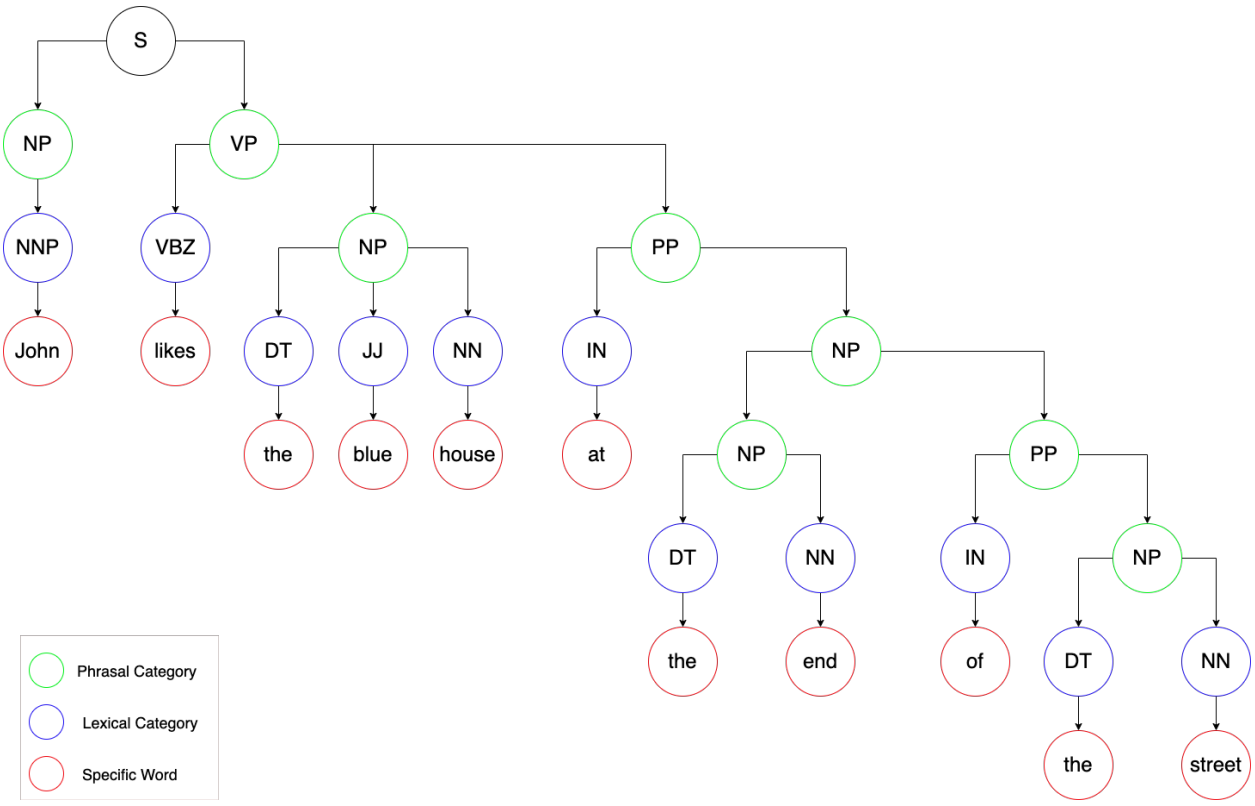


Figure 6.1: Parse tree of the sentence "John likes the blue house at the end of the street". The root tree node is an 'S' which represents the sentence as a whole. This is then broken into each phrasal and lexical category, each of which is color coded.

6.1 demonstrates, some phrasal categories consist of other phrasal categories. To properly map the transition probabilities for each phrasal category to the next, each level of the tree was considered independently of other levels. Figure 6.2 shows each level of the parse tree labeled.

With the exception of using phrasal categories instead of lexical categories, the model is trained exactly as it would be if it was using just lexical categories. Each sentence is split up into the specific hidden and observable nodes, and an emission, transition, and initial probabilities are created. Once the model is trained on sentences constraints could be applied to any position, and a sequence can be generated. In the previous example of "John likes the blue house at the end of the street", 3 phrasal categories can be parsed and each include one

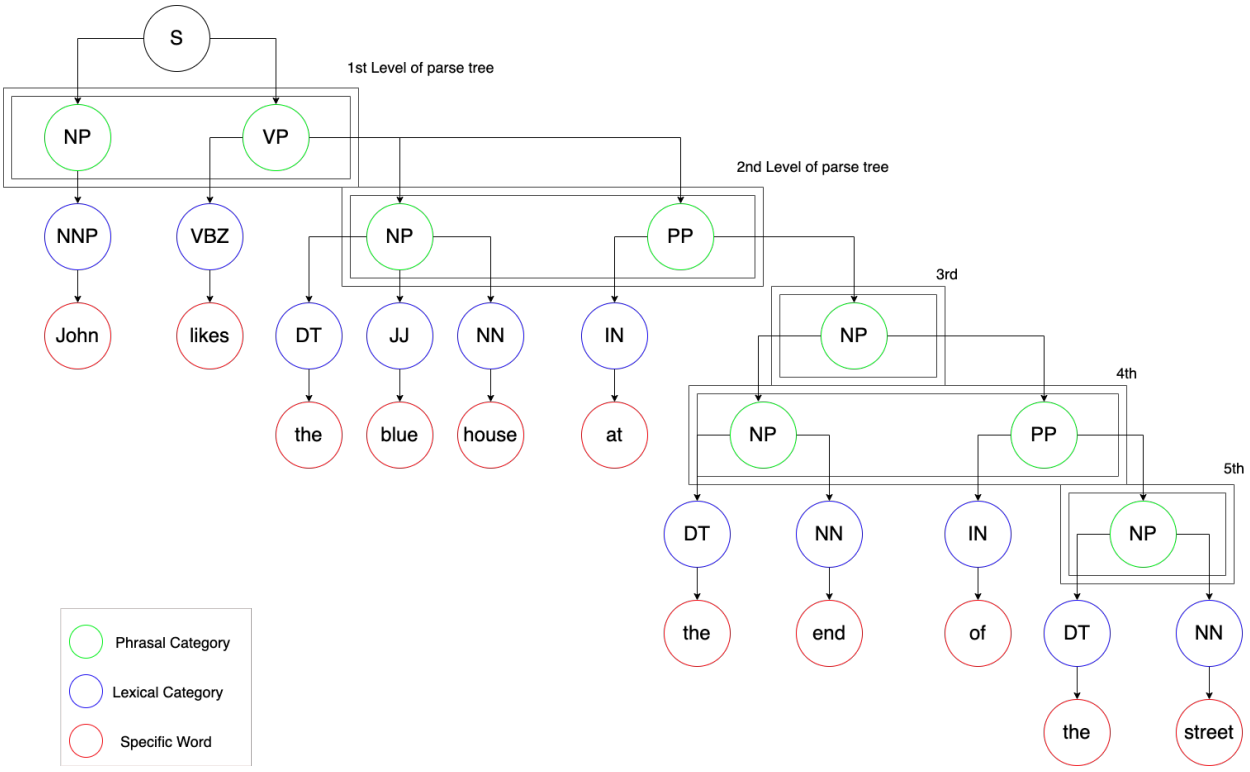


Figure 6.2: Parse tree of the sentence "John likes the blue house at the end of the street". The root tree node is an 'S' which represents the sentence as a whole. This is then broken into each phrasal and lexical category, each of which is color coded. This has the addition of each level of the parse tree being contained for the purposes of showing how the transition matrix is formed.

or more phrases.

$$\text{NP} = \{\textit{John, the blue house, the end, the street}\}$$

$$\text{VP} = \{\textit{likes the blue house at the end of the street}\}$$

$$\text{PP} = \{\textit{at the end of the street, of the street}\}$$

Careful consideration and an adequately sized dataset must be used to ensure sentences aren't repeated due to the overlapping nature of the phrases. If only this sentence was used to train the model, and a 3 phrase sequence was requested, sentence such as: "the street likes the blue house at the end of the street" or "John at the end of the street" would be possible.

Acronym	Lexical Category
NP	noun phrase
VP	verb phrase
PP	prepositional phrase

Table 6.1: Phrasal categories used in the examples and their respective acronyms.

Because this adapted model focuses on phrases in the observable states, an opportunity is presented to use floating constraints. A floating constraint is a constraint that can be applied to any position of the phrase. If a noun phrase, such as "the blue house" is evaluated, any word in the phrase could be considered to meet a constraint, unless the position of the constraint was explicitly specified. The original CHiMP model required carefully constructed constraints in which the position of the constraint had to be specified. For instance, if the constraint had a specific wording requirement many possible sentences would have been removed from the model before the original CHiMP model could find an appropriate solution.

6.3 Evaluation

To evaluate the phrasal category based CHiMP model in an attempt to answer our research question "How can we mitigate the negative impact of diminished coherence while maintaining the expressive capabilities?", we trained 3 models. The first and second models were the default CoMP and CHiMP models outlined in Chapter 3. The third model was the new phrasal category CHiMP model. To train each model, 100 thousand sentences were picked randomly from the 2012 fiction section of the Corpus of Contemporary American English (COCA) data set [25]. After the sentences were randomly selected, each sentence was parsed and used to create the probability matrices for the models. The lexical and phrasal categories were derived from the training sentences using the Charniak-Johnson parser (bllip-parser) [16] and the Natural Language Tool Kit [45].

To create a model capable of generating sequences we believed capable of answering our research question, each model was used to generate a limerick task. Generally a limerick is a short five line poem where the first two lines and the fifth line are slightly longer and

have a different rhyme scheme than the third and fourth lines. In order to generate these limericks, we had to explicitly define what a limerick is, allowing sequences fitting the general definition of a limerick to be created. In our first and second models, because these were focusing on specific words and not phrases, we had to define each position of the sequence explicitly. We defined a limerick as a 25 word phrase with each observed state having a certain amount of syllables. We did this with the intention of creating sequences with the precise amount of syllables in each line. In this case, we were looking for lines 1,2,5 to each have 8 syllables, while lines 3,4 had 5 syllables. We also had to ensure that the last words of each phrase rhymed. In total, the base CHiMP and CoMP models had approximately 30 constraints each to properly define each part of the sequence for it to be close to what might be considered a proper limerick.

The phrasal category CHiMP model was similar, however, because phrases were in the observable states instead of just single words, the length of the generated sequences could be set to 5 instead of 25 like with the previous models. This significantly reduced the total number of constraints that had to be applied. It also slightly changed each constraint in that the entire phrase had to be considered, thus floating constraints were used instead of explicit single word constraints.

An additional step was taken for some limericks and that was to add a theme to the limericks in an attempt to further constrain the generated sequences. These themes were randomly chosen "nature" based themes such as "mountain" or "lake". The sequences with the theme constraints were evaluated based on a similarity threshold calculated on the Word2vec word embeddings [32]. The Word2vec similarity score would compare each word provided with the theme. If these words were within a similarity value of eachother, the word was deemed "similar enough" and was not removed from the matrices. There were some challenges with this when it came to applying the constraints. Where should the constraints be applied? Should each line of the limerick have the same constraint? What threshold should be used when specifying that a given word or line meets the theme? These were all questions

that presented themselves that are topics for future work, however we still continued with a basic set of themes and a low threshold when applying constraints to the phrasal and lexical CHiMP models.

To verify the outcome of this new CHiMP model, a survey was created that asked random participants to rank these generated limericks. With the constraints in place, to gather limericks for the survey, each of the models were used to generate 10 limericks based on randomly selected rhyme schemes. Of these limericks, we removed limericks that had repeating rhyming words, as that occasionally occurred when narrowing down our solution space based on all of the constraints. We also removed limericks that were explicit in nature as to not offend any of the random participants who may be taking the survey. There were also human generated limericks that were used as a control for each task. These human generated limericks were created by friends of the author with the only guidance being the rhyme scheme that needed to be used.

Besides comparing each of the models, another goal was to reduce the opaqueness of natural language processing models. Even if the model does not create equally good or better solutions to other state of the art models, showing how the model works, and ensuring it can be used without "guessing" how it might work, has value.

There were two sections in the survey. The first section looked at the lexical coherence (i.e., the arrangement of words and phrases to create well-formed sentences). These were rated on a Likert scale of 1-5.

1. Very poor - Completely incohesive;
2. Poor - Mostly incohesive;
3. Fair - Equally cohesive and incohesive;
4. Good - Mostly cohesive;
5. Excellent - Completely cohesive;

The second section of the survey ranked how well the limericks fit a specific theme. The purpose of this section was to see how well floating constraints could be applied to a generated sequence. These were also on a Likert scale from 1-5.

1. Very poor - No relation to the theme;
2. Poor - Very little relation;
3. Fair - Some relation;
4. Good - The theme is clear;
5. Excellent - The theme is clear and the limerick fits the theme;

Once the survey was created, it was distributed via a social media website called Reddit. More specifically, the survey and a short description of the tasks being asked of each participant. The survey was posted on the r/samplesize reddit forum. This reddit forum, also called a subreddit, is intended for surveys of many different backgrounds and is not specifically aimed at any demographic. Less than 10% of the survey takers were colleagues of the authors.

6.3.1 Results

In the first section of the survey where participants were asked to rank the overall cohesion of the generated limericks, the control limericks scored the highest. It was expected that the human generated limericks would have the highest score. The goal was for the phrasal category CHiMP model to score higher than the original CHiMP and CoMP models, and using the average, this was accomplished.

In considering the results of the 3 models that generated limericks, we found that the overall coherency score for the phrasal category based CHiMP, referred to as "CHiMP 2.0", was 2.50. The CHiMP model based on lexical categories, referred to as "CHiMP 1.0", was given an average coherency score of 1.42, and the CoMP model has a score of 1.86. The control limericks had an average score of 4.2. Figure 6.3 provides the summary of the results.

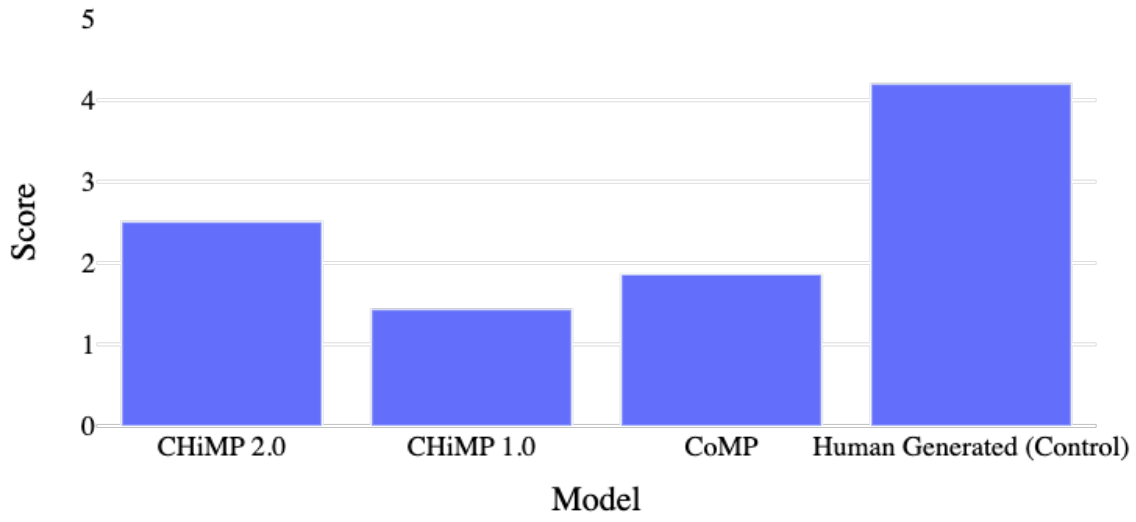
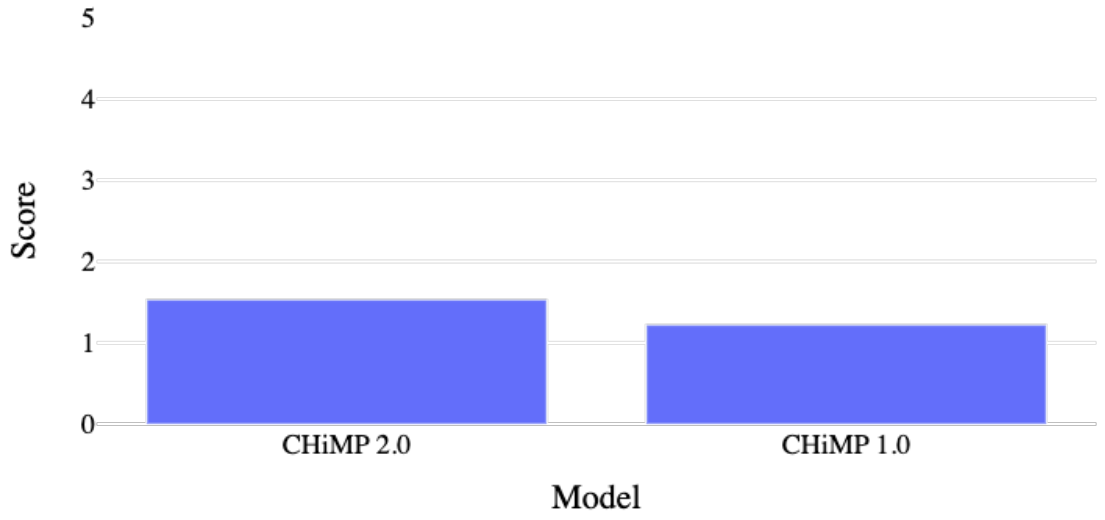


Figure 6.3: Semantic Survey results

For the second half of the survey, only the phrasal and semantic category CHiMP models were used. 6.4 provides the summary of the results. It's apparent that these results were not good. There are may suspected reasons for this. First, the dataset may not have been large enough. Since words and phrases not seen in the dataset cannot be used in the generated sequences, increasing the dataset size would open up the solution space allowing for potentially better themed phrases. Second, the constraints could have been too restrictive. When applying theme constraints, a threshold score had to be used. Lowering this score may have helped more sequences be generated, however it would also lower the scores of how well the limerick fit the specific theme. Third, more theme constraints could be applied to the limericks. For this section of the survey, only the first line of each limerick had this constraint applied. Applying the constraint to each line may change the perception of it fitting a specific theme. All of these issues are potential topics for future research.

Figure 6.4: Theme Survey results



6.4 Discussion

In the introduction, we asked the question: How can we mitigate the negative impact of diminished coherence while maintaining the expressive capabilities? After exploring phrasal and lexical categories more in depth, we believe including a combination of phrasal and lexical categories is the best way to achieve the improved coherence while maintaining the expressive capabilities. The lexical phrases open the door to an increasing level of options when explicit constraints are required. If each part of the requested sequence needs to have specific constraints, such as with mnemonic devices as we wrote about in Chapter 4, then a hidden Markov model just based on lexical categories is probably sufficient. However, for less specific generative tasks, such as limerick or haiku generation or other short sequences, the phrasal category hidden states will provide a better method of maintaining natural language coherence while maintaining most of the expressive capabilities of previous models.

As desired, the phrasal category based CHiMP model performed better than both of the other two computer models. Based on the constraints though, the outcome for both parts

of the survey could have been different. Looking first at the outcome of the cohesion semantic portion of the survey, it makes intuitive sense that a model that uses phrases instead of just the words from the training set would be more cohesive. This really holds true when longer phrases are requested, as we did when generating the limericks in the survey. Instead of generating five words that either were seen at some point in the training data for the CoMP model, or their corresponding semantic categories for the CHiMP model, we can generate just one sequence that was seen verbatim somewhere in the training data. However if we had decided to apply more constraints to the phrasal category model, this may not have been the case. Since a phrasal category can contain just a single semantic category, then we could effectively generate the same sequences.

In addition to the increased cohesion of the generated sequences, the phrasal category based model also allows for floating constraints which add another level to the expressive capabilities, especially when one wants to look at the computationally creative power of these models. It may be hypothesised that when a large amount of constraints is applied to the model, the model is not actually being that expressive and is only providing the small amount of possible solutions that come from the constraints. Floating constraints help with this problem by opening up where constraints are applied and reducing the need to be so explicit. This is seen in the second half of the survey when theme based limericks were generated. If we had applied the theme constraints in a different order, different position, higher threshold, or applied more of them, the results may also have been different. The models may not have been able to generate solutions as well though. This was a problem for the original CoMP model and that is why it was not in the survey.

6.4.1 Ethical Concerns

Because this model was trained using human generated data, there is a need to look at the possible ethical implications that could be generated from using this data source. Because this data primarily comes from the public and is pulled through news articles, social media

posts or other publicly available outlets, much of the data could be biased. Researchers working with this type of data need to be aware that results generated by the model may not be suitable for all ages or might have racist, sexist, transphobic, or other issues. It is not the intention of the authors to promote any of this type of language in the generated results. All of the sequences that were presented in the survey were approved by the Internal Review Board at Idaho State University.

While a survey was conducted to judge the generated results, care was taken to remove potentially offensive words from the limericks that were put into the survey. Common words that are may be considered "swear" words were removed.

Another ethical concern is plagiarism. It's not unheard of for NLP AI, even those trained on substantially more data, to plagiarise [74]. For example, the GPT3 trained AI meant for generating computer code would plagiarise a famous algorithm used in the video game Quake 3 [6]. The model created and trained in this work is no different and is even more likely to plagiarize. Because we are focusing on phrases, there is a chance phrases can appear in a similar or exact order that they appear in the training data. This is a statistical model looking at the probabilities of phrases appearing, so small phrases are used directly, however it is not intentional to plagiarize the authors of the original work from phrase to phrase.

6.5 Conclusion

Continued development on the CHiMP model to include phrasal categories led to the discovery of challenges and many additional research questions that will need further exploration. The challenge of increasing the cohesion of generated sequences has been improved by using these phrasal categories, however more work is necessary. Additional research on the question of constraints is also needed as a base set of constraints was used for each of the generated sequences in this chapter, however they may not have been the optimal constraints. Nevertheless, the survey performed in this chapter, albeit with a small sample size, does

show that we can increase the coherence of generated sequences by focusing on the phrasal categories in the hidden states instead of the lexical categories.

Chapter 7

Summary

In Chapter 1, a natural language processing was briefly introduced and statistical NLP was discussed. Chapter 2 provided an introduction to Markov Models and Hidden Markov and use mathematical definitions to define them as well as demonstrative figures. In Chapter 3, a previous published work at the Intermountain Engineering, Technology and Computing conference was included to show how generation of sequences with constraints applied can be used in conjunction with Markov Models. Chapter 4 continues with a computational creativity based application on constrained Hidden Markov Models with mnemonic device generation. This paper was published at the International Conference on Computational Creativity. Chapter 5 continues with an analysis of the solution spaces that are generated with constrained hidden Markov models. This paper was also published at the International Conference on Computational Creativity a year after the paper in Chapter 4. Chapter 6 provides new methods and applications of the CHiMP model by focusing on phrasal categories rather than just the parts of speech or the lexical categories. The phrasal categories add the additional capabilities of being able to use floating constraints instead of individual position constraints.

Bibliography

[1] Phrases. URL https://www.englishclub.com/grammar/sentence/phrases.htm.

[2] A robot wrote this entire article. are you scared yet, human? — gpt-3, Sep 2020. URL https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3.

[3] Price: GPU server rental: Xesktop gpu rendering, Jun 2021. URL https://xesktop.com/price/.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[5] Hussain-Abdulah Arjmand, Jesper Hohagen, Bryan Paton, and Nikki S. Rickard. Emotional Responses to Music: Shifts in Frontal Brain Asymmetry Mark Periods of Musical Change. *Frontiers in Psychology*, 8(DEC):2044, dec 2017. ISSN 1664-1078. doi: 10.3389/fpsyg.2017.02044. URL http://journal.frontiersin.org/article/10.3389/fpsyg.2017.02044/full.

[6] Gregory Barber. Github's commercial ai tool was built from open source code, Jul 2021. URL https://www.wired.com/story/github-commercial-ai-tool-built-open-source-code/.

[7] Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. Markov constraints for generating lyrics with style. In *Proceedings of the Twentieth European*

Conference on Artificial Intelligence, pages 115–120, 2012. ISBN 9781614990970. doi: 10.3233/978-1-61499-098-7-115.

[8] Margaret A. Boden. *The Creative Mind: Myths and Mechanisms, Second Edition*. Routledge, 2003. ISBN 0203508521. doi: 10.4324/9780203508527.

[9] Paul Bodily, Benjamin Bay, and Dan Ventura. Computational creativity via human-level concept learning. In *Proceedings of the Eighth International Conference on Computational Creativity*, pages 57–64, 2017.

[10] Paul M. Bodily, Porter Glines, and Brandon Biggs. "She Offered No Argument": Constrained Probabilistic Modeling for Mnemonic Device Generation. In Kazjon Grace, Michael Cook, Dan Ventura, and Mary Lou Maher, editors, *Proceedings of the 10th International Conference on Computational Creativity*, pages 81–88, Charlotte, North Carolina, 2019. Association for Computational Creativity.

[11] Paul M Bodily, Porter Glines, and Brandon Biggs. " she offered no argument": Constrained probabilistic modeling for mnemonic device generation. In *ICCC*, pages 81–88, 2019.

[12] F. P. Brooks, A. L. Hopkins, P. G. Neumann, and W. V. Wright. An Experiment in Musical Composition. *IRE Transactions on Electronic Computers*, EC-6(3):175–182, 1957. ISSN 03679950. doi: 10.1109/TEC.1957.5222016.

[13] Keith Brown and Jim Miller. *Syntax: A linguistic introduction to sentence structure*. Routledge, 2020.

[14] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL `https://arxiv.org/abs/2005.14165`.

[15] Jason Brownlee. Gentle introduction to statistical language modeling and neural language models, Aug 2019. URL `https://machinelearningmastery.com/statistical-language-modeling-and-neural-language-models/`.

[16] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, 2005.

[17] François Chollet et al. Keras. `https://keras.io`, 2015.

[18] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[19] Simon Colton. Creativity Versus the Perception of Creativity in Computational Systems. *Proceedings of the AAAI Spring Symposium on Creative Systems*, 2008.

[20] Simon Colton and Geraint A. Wiggins. Computational creativity: The final frontier? In *Proceedings of the Twentieth European Conference on Artificial Intelligence*, pages 21–26. IOS Press, 2012. ISBN 9781614990970. doi: 10.3233/978-1-61499-098-7-21. URL `http://www.idi.ntnu.no/~agnar/Documents/Colton_Wiggins12.pdf`.

[21] Glen Coppersmith, Ryan Leary, Patrick Crutchley, and Alex Fine. Natural language processing of social media as screening for suicide risk. *Biomedical informatics insights*, 10:1178222618792860, 2018.

[22] Mihály Csíkszentmihályi. *Flow and the Psychology of Discovery and Invention*. Harper Perennial, 1996. ISBN 0060928204. doi: 10.1037/e586602011-001.

[23] Robert Dale, Hermann Moisl, and Harold Somers. *Handbook of natural language processing*. CRC press, 2000.

[24] Mark Davies. 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, 2009.

[25] Mark Davies. Corpus of Contemporary American English (COCA), 2015. URL `https://doi.org/10.7910/DVN/AMUDUW`.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

[27] Katherine Elkins and Jon Chun. Can GPT-3 Pass a Writer's Turing Test? *Journal of Cultural Analytics*, 1(1):17212, 2020.

[28] Robert Freidin. A transformational approach to english syntax: Root, structure-preserving, and local transformations, 1978.

[29] Porter Glines, Brandon Biggs, and Paul M Bodily. A leap of creativity: From systems that generalize to systems that filter. In *ICCC*, pages 297–302, 2020.

[30] Porter Glines, Brandon Biggs, and Paul Bodily. Probabilistic generation of sequences under constraints. In *Proceedings of the First Intermountain Engineering, Technology, and Computing Conference*, in press.

[31] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[32] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[33] Alex Graves. Generating Sequences With Recurrent Neural Networks. aug 2013. URL `http://arxiv.org/abs/1308.0850`.

[34] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[35] Gaëtan Hadjeres and Frank Nielsen. Anticipation-rnn: Enforcing unary constraints in sequence generation, with application to interactive music generation. *Neural Computing and Applications*, pages 1–11, 2018.

[36] Ram SagarI have a master's degree in Robotics and I write about machine learning advancements. How to take full advantage of GPUs in large language models, May 2021. URL `https://analyticsindiamag.com/how-to-take-advantage-gpus-large-language-models-gpt-3/`.

[37] Anna Jordanous. Four PPPPerspectives on computational creativity in theory and in practice. *Connection Science*, 28(2):194–216, 2016. doi: 10.1080/09540091.2016.1151860.

[38] Paresh Kharya and Ali Alvi. Using deepspeed and megatron to train megatron-turing nlg 530b, the world's largest and most powerful generative language model, Oct 2021. URL `https://developer.nvidia.com/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-a`

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

[40] Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. Controlled text generation as continuous optimization with multiple constraints. *arXiv preprint arXiv:2108.01850*, 2021.

[41] Vipin Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI magazine*, 13(1):32–32, 1992.

[42] Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. Data-Driven News Generation for Automated Journalism. pages 188–197. Association for Computational Linguistics (ACL), jan 2018. doi: 10.18653/v1/w17-3528.

[43] Elizabeth D Liddy. Natural language processing. 2001.

[44] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL `http://arxiv.org/abs/1907.11692`.

[45] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[46] Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[47] Christopher D Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International conference on intelligent text processing and computational linguistics*, pages 171–189. Springer, 2011.

[48] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *arXiv preprint arXiv:1708.00107*, 2017.

[49] Richard G Morris, Scott H Burton, Paul M Bodily, and Dan Ventura. Soup Over Bean of Pure Joy : Culinary ruminations of an artificial chef. In *Proceedings of the Third International Conference on Computational Creativity*, pages 119–125, 2012. ISBN 9781905254668. URL `http://computationalcreativity.net/iccc2012/wp-content/uploads/2012/05/119-Morris.pdf`.

[50] Aishwarya Narasimhan, Krishna Prasad Agara Venkatesha Rao, et al. Cgems: A metric model for automatic code generation using gpt-3. *arXiv preprint arXiv:2108.10168*, 2021.

[51] Aran Nayebi and Matt Vitelli. GRUV: Algorithmic Music Generation using Recurrent Neural Networks. *Deep Learning for Natural Language Processing*, 2015.

[52] David Norton, Derrall Heath, and Dan Ventura. Finding creativity in an artificial artist. *Journal of Creative Behavior*, 2013. ISSN 00220175. doi: 10.1002/jocb.27.

[53] Joseph C. Nunes, Andrea Ordanini, and Francesca Valsesia. The power of repetition: Repetitive lyrics in a song increase processing fluency and drive market success. *Journal of Consumer Psychology*, 25(2):187–199, 2014. ISSN 10577408. doi: 10.1016/j.jcps.2014.12.004.

[54] Balder Onarheim and Michael Mose Biskjaer. Balancing constraints and the sweet spot as coming topics for creativity research. In *Creativity in design: Understanding, capturing, supporting*. APA, 2017. URL `http://orbit.dtu.dk/files/103339029/Balancing_Constraints_and_the_Sweet_Spot.pdf`.

[55] François Pachet. The Continuator: Musical Interaction With Style. *Journal of New Music Research*, 32(3):333–341, sep 2003. ISSN 0929-8215. doi: 10.1076/jnmr.32.3.333.16861. URL `http://www.tandfonline.com/doi/abs/10.1076/jnmr.32.3.333.16861`.

[56] François Pachet, Pierre Roy, and Gabriele Barbieri. Finite-length markov processes with constraints. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[57] François Pachet, Pierre Roy, and Gabriele Barbieri. Finite-length Markov processes with constraints. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011. ISBN 9781577355120. doi: 10.5591/978-1-57735-516-8/IJCAI11-113.

[58] Alexandre Papadopoulos and Pierre Roy. Avoiding plagiarism in Markov sequence generation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2731–2737, 2014. URL `www.aaai.org`.

[59] Alexandre Papadopoulos, François Pachet, Pierre Roy, and Jason Sakellariou. Exact sampling for regular and Markov constraints with belief propagation. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, pages 341–350. Springer, 2015. doi: 10.1007/978-3-319-23219-5{_}24. URL `https://www.researchgate.net/publication/283343726`.

[60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga,

Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[61] Alison Pease, Markus Guhe, and Alan Smaill. Some aspects of analogical reasoning in mathematical creativity. In *Proceedings of the First International Conference on Computational Creativity*, pages 60–64, 2010.

[62] Guillaume Perez and Jean-Charles Régin. MDDs: Sampling and probability constraints. In *Proceedings of the Twenty-Third International Conference on Principles and Practice of Constraint Programming.*, pages 226–242. Springer, Cham, 2017. URL `http://www.constraint-programming.com/people/regin/papers/sampling.pdf`.

[63] Rafael Pérez y Pérez and Mike Sharples. Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA. *Knowledge-Based Systems*, 2004. ISSN 09507051. doi: 10.1016/S0950-7051(03)00048-0.

[64] L. R. Rabiner and B. H. Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, 3(1):4–16, 1986. ISSN 07407467. doi: 10.1109/MASSP.1986.1165342.

[65] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[66] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[67] Paul Jasmin Rani, Jason Bakthakumar, B Praveen Kumaar, U Praveen Kumaar, and Santhosh Kumar. Voice controlled home automation system using natural language processing (nlp) and internet of things (iot). In *2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM)*, pages 368–373. IEEE, 2017.

[68] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Conference on empirical methods in natural language processing*, 1996.

[69] Graeme Ritchie. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17(1):67–99, 2007. ISSN 09246495. doi: 10.1007/s11023-007-9066-2. URL https://link.springer.com/content/pdf/10.1007%2Fs11023-007-9066-2.pdf.

[70] Stéphane Rivaud and François Pachet. Sampling Markov models under constraints: Complexity results for binary equalities and grammar membership. *arXiv preprint*, 2017. URL https://arxiv.org/pdf/1711.10436.pdf.

[71] Pierre Roy, Guillaume Perez, Jean-Charles Régin, Alexandre Papadopoulos, François Pachet, and Marco Marchini. Enforcing Structure on Temporal Sequences: The Allen Constraint. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, pages 786–801. Springer, 2016. doi: 10.1007/978-3-319-44953-1{_}49. URL https://www.researchgate.net/publication/304490456.

[72] Rob Saunders and John S Gero. The Digital Clockwork Muse: A Computational Model of Aesthetic Evolution. In *Proceedings of the Artificial Intelligence and Simulation of Behavior Convention*, pages 12–21, 2001. URL https://pdfs.semanticscholar.org/420b/9355610e47a21398024ff3b423d66a002717.pdf.

[73] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[74] Chad Thiele. Gpt-3 ai plagiarism and fact-checking, Mar 2021. URL https://aicontentdojo.com/gpt-3-ai-plagiarism-and-fact-checking/.

[75] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Junichi Yamagishi, and Keiichiro Oura. Speech synthesis based on hidden Markov models. *Proceedings of the IEEE*, 101(5):1234–1252, 2013. ISSN 00189219. doi: 10.1109/JPROC.2013.2251852.

[76] Dan Ventura. Mere Generation: Essential Barometer or Dated Concept. *Proceedings of the Seventh International Conference on Computational Creativity, ICCC*, (June):22–29, 2016. URL https://pdfs.semanticscholar.org/51eb/ae710bf2c884b6c0b1479085604554fc9da4.pdf.

[77] Dan Ventura. How to Build a CC System. In *Proceedings of the Eighth International Conference on Computational Creativity*, pages 253–260, 2017. URL http://computationalcreativity.net/iccc2017/ICCC_17_accepted_submissions/ICCC-17_paper_20.pdf.

[78] Kyle Wiggers. Openai's massive gpt-3 model is impressive, but size isn't everything, Jun 2020. URL https://venturebeat.com/2020/06/01/ai-machine-learning-openai-gpt-3-size-isnt-everything/.

[79] Geraint A. Wiggins. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7):449–458, 2006. doi: 10.1016/j.knosys.2006.04.009.

[80] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971.

[81] Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[82] Maosen Zhang, Nan Jiang, Lei Li, and Yexiang Xue. Language generation via combinatorial constraint satisfaction: A tree search enhanced monte-carlo approach. *arXiv preprint arXiv:2011.12334*, 2020.

[83] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.