A Weighted Mixture Approach to Text Prediction

Lei Qian
Department of Statistical Science
Duke University
Durham, NC 27708
lei.qian@duke.edu

Wei (Emily) Shao
Department of Statistical Science
Duke University
Durham, NC 27708
wei.shao@duke.edu

Leonardo Shu
Department of Statistical Science
Duke University
Durham, NC 27708
leonardo.shu@duke.edu

Victor Yifan Ye
Department of Economics
Duke University
Durham, NC 27708
yifan.ye@duke.edu

Abstract

We create an algorithm to generate new content via existing text using a mixture model consisting of a weighted prediction structure and several predictive components. Weights for a given category of text can either be obtained from visual examination of output, or derived via an iterative training process based on the Metropolis algorithm. Compared to single-model methods of text prediction, the mixture model achieves comparable predictive results while being computationally cheaper and more flexible in specifications.

1 Motivation

Hidden Markov models (HMMs) have been used broadly in the predictive modeling of speech and text (Bahi et al. 1986; Efros and Leung 1999; Fine, Singer and Tishby 1998). However, the achieving of consistent predictions remains non-trivially computationally costly: the standard Baum-Welch (BW) algorithm (Baum and Eagon 1967; Baum et al. 1970) requires a complexity strictly exceeding $2nm^2$ for a total of m discrete hidden states and n elements in the sequence. As an example, a moderate-length input sequence of 20,000 words with 50 hidden states requires 10^8 calculations per iteration without taking M-step costs into consideration. Any model that significantly exceeds this level of complexity is difficult to implement in a reasonable time-frame.

Subsetting the input sequence or utilizing a smaller number of hidden states will both significantly reduce the computational cost of Baum-Welch, yet at the cost of predictive performance. While simpler approaches to text prediction such as pure Markov Chain based models exist, they are problematic for two reasons. First, there is no "learning" process involved in a pure Markov Chain as new words are sequentially predicted solely based on the existing transition structure of the training text. Second, Markov Chain models that introduce local structure are often excessively restrictive. The Mark V. Shaney algorithm (Ellis and Pike [1981-1982?]) uses unique three-word sets to construct sentences that respect local structure, yet relies heavily on the amount of content for a given level of language use complexity.[1] Even if there exists a large body of content to perform this Markov Chain algorithm on, it is still very likely that that generated sentences will either be mostly identical or highly similar to sentences from the original text content.

[1]The specific date of writing of the Mark V. Shaney algorithm is unknown. See Dewdney (1989) for a discussion on early text training methods.

In this paper, we combine three differently-specified text prediction models to improve performance over each individual model. In particular, we separately estimate a sequence of hidden states as well as corresponding emission and transition matrices with Baum-Welch, a simple Markov-Chain transitional matrix estimating the transition probabilities of consecutive words, and a three-dimensional array of transition probabilities from Mark V. Shaney. From each model, a vector of probabilities for the next word in a sequence is derived and assigned a fixed weight, and the three weighted vectors are combined and standardized to predict the new word.

Section two begins with a description of our approach as well as derivations for the three models. Section three presents two case studies with the mixture model: presidential debate transcripts of Bernie Sanders and song lyrics from Adele and Jason Mraz. Section four discusses training procedures for weights and Section five concludes.

2 Approach

Our approach involves a linear combination of three models, a simple Monte Carlo model using a word-pair transition matrix, a Hidden Markov Model implemented with the B-W algorithm and a second order Markov Chain model (Mark V. Shaney). We first derive transitional probabilities according to each model.

2.1 Simple Monte Carlo

The basic model for the new-word prediction problem is, for a new text chunk of size n, to draw the jth word x_j using a transition matrix $\mathbb{W}$ of size W x W where W equals the number of unique words in the text. The transition matrix has its probabilities calculated empirically by matching how often each words is next to all others and then standardizing throughout.

Given the preceding word x^*_{j-1} and $\mathbb{W}$ with probability columns, the vector of probabilities given by the simple Monte Carlo model is the column of the unique word x_{i-1}, $V^{MC}_{x^*_{j-1}}$. We note that the length of $V^{MC}_{x^*_{j-1}}$ is the length of the total number of unique words in the text, with words that are never preceded by x^*_{j-1} assigned a probability of zero.

2.2 Hidden Markov Model

We apply the Baum-Welch algorithm for the Hidden Markov model. The Baum-Welch algorithm uses the Expectation-Maximization algorithm to find the maximum likelihood estimates of the parameters of the hidden Markov Model, which include an initial state distribution (π), a transition matrix (T) and an emission matrix (ϕ). [2]

We have $i, j = 1, ..., m$ hidden states and $w = 1, ..., W$ unique words, with text $x_1, \dots x_N$ with N words in total. The derivation of the Baum-Welch algorithm has the following two parts:

(1) The expectation (E) step calculates the expectation of the log likelihood of probabilities Q(θ, θ_k) given the current estimates θ_k and the proposed parameter θ.

$$\begin{aligned} Q(\theta, \theta_k) &= E_{\theta_k}(\log(P_\theta(x, z)|X = x)) \\ &= \sum_{i=1}^{m} P_{\theta_k}(Z_1 = i|x)\log(\pi_i) + \sum_{t=2}^{N}\sum_{i=1}^{m}\sum_{j=1}^{m} P_{\theta_k}(z_{t-1} = i, z_t = j|x)\log(T_{ij}) \\ &+ \sum_{t=1}^{N}\sum_{i=1}^{m} P_{\theta_k}(Z_t = i|x)\log(f_{\phi_i}(x_t)) \end{aligned}$$

$$\begin{aligned} \gamma_{ti} &= P_{\theta_k}(Z_t = i|x) \\ \beta_{tij} &= P_{\theta_k}(Z_{t-1} = i, Z_t = j|x) \end{aligned}$$

[2] Derivations from Jeffery Miller (2016) Stat 531 course notes, `https://sakai.duke.edu/access/content/group/46bdae81-e6e3-45fb-aed1-5cd4f5e335af/Notes/notes-1.pdf`

$$Q(\theta, \theta_k) = \sum_{i}^{m} \gamma_{1i} \log(\pi_i) + \sum_{t=2}^{N} \sum_{i=1}^{m} \sum_{j=1}^{m} \beta_{t_{ij}} \log(T_{ij}) + \sum_{t=1}^{N} \sum_{j=1}^{m} \sum_{i=1}^{m} \gamma_{ti} \log(f_{\phi_i}(x_t))$$

(2) The maximization (M) step computes the maximized initial states π, transition matrix T and emission matrix ϕ, accordingly. After applying Lagrange multipliers to solve for each parameter, we get:

$$\pi_i = \frac{\gamma_{1i}}{\sum_{j=1}^{m} \gamma_{1j}}$$

$$T_{ij} = \frac{\sum_{t=2}^{N} \beta_{tij}}{\sum_{t=2}^{N} \sum_{j=1}^{m} \beta_{tij}} = \frac{\sum_{t=2}^{n} \beta_{tij}}{\sum_{t=1}^{N-1} \gamma_{ti}}$$

$$\phi_{iw} = \frac{\sum_{x:x_t=w} \gamma_{ti}}{\sum_{w} \sum_{x:x_t=w} \gamma_{ti}}$$

By iteratively updating the E and M steps, parameters for the HMM converge. Let π^*, T^* and ϕ^* be the initial state vector, the transition matrix, and the emission matrix given after q iterations. We can then initialize the model by drawing z_1 from π^* and the first word of the generated text sequence, x_1^*, from ϕ^*, and draw a vector of z's $z_2 \dots z_n$ with T^*. Because Mark V. Shaney cannot begin until the 3rd word in the generated text, we also draw x_2^* given z_2 and ϕ^*, and assign probability vector $V_{z_j}^{HMM}$ to the jth term as given by the respective probability column in the emission matrix for all generated words such that $j > 2$.

2.3 Mark V. Shaney

Fundamentally, the Mark V. Shaney algorithm is a 2nd order Markov Chain where the probability of any given word (except the starting two) is dependent only on the previous pair. Given x_1^* and x_2^* fixed from the B-W algorithm output, we can draw a vector of probabilities, $V_{x_{j-1},x_{j-2}}^{MvS}$ for each x_j^* where each element p_k of $V_{x_{j-1}^*,x_{j-2}^*}^{MvS}$ is the probability of x_j^* being the unique word w_k given x_{j-1}^* and x_{j-2}^*.

In practice, storing information for all unique three-word sets is costly. For a pure Mark V. Shaney implementation, it is typically more convenient to generate probabilities for the new words given preceding words as each new word is generated. However, computing and storing the entire 3-dimension matrix $\mathbb{M}$ of size $W \times W \times W$ that consists of transition probabilities $(p_{x_j^*}, w_{x_{j-1}^*}, w_{x_{j-2}^*})$ for all unique 3-word sets saves computation time per iteration. Since we generate large quantities of text, this is the approach we select.

2.4 Mixture

Putting the three models together, we now introduce a vector of weights $l = (l_1, l_2, l_3)$ such that $\sum l = 1$. We note that each of the model probability vectors for x_j^* have an identical length of W and that each sum to 1. Therefore, the mixture vector of predictive probabilities for x_j^*:

$$V_{x_j^*}^{mix} = l_1 * V_{x_{j-1}^*}^{MC} + l_2 * V_{z_j}^{HMM} + l_3 * V_{x_{j-1}^*,x_{j-2}^*}^{MvS}$$

Is also a probability vector. By setting l_1, l_2, l_3 to 1 respectively and the other two to 0, the mixture model reduces to each of the component models individually. Conceptually, the mixture can be considered either as a weakening of the relatively fixed Mark V. Shaney structure via introducing deviations determined by lower-order structure and hidden states, or as a low-order prediction model that is strengthened by introducing a large probability to the most likely new words given by Mark V. Shaney.

We illustrate the mixture algorithm process using a toy example with the sequence of words $(A, B, C, E, D, A, B, B, E)$:

The unique three-word groups are:

$$\begin{Bmatrix} \{A,B,C\} \\ \{B,C,E\} \\ \{C,E,D\} \\ \{E,D,A\} \\ \{D,A,B\} \\ \{A,B,B\} \\ \{B,B,E\} \end{Bmatrix}$$

and unique two-word groups are:

$$\begin{Bmatrix} \{A,B\} \\ \{B,C\} \\ \{C,E\} \\ \{E,D\} \\ \{D,A\} \\ \{B,B\} \\ \{B,E\} \end{Bmatrix}$$

Now suppose that the first two words have been selected as (A, B) via Baum Welch. Also suppose that for the hidden states z_3, the vector of emission probabilities is given by $(p_A, p_B, p_C, p_D, p_E)'$. By Mark V. Shaney, the only two words that can follow from (A, B) are B and C. By simple Monte Carlo, the only words that follow B are B, C and E. Therefore, here we have $V_{AB}^{MvS} = (0, 1/2, 1/2, 0, 0)'$ and $V_B^{MC} = (0, 1/3, 1/3, 0, 1/3)'$. Given weights l, we can write the mixture probability vector as:

$$V^{mix} = l_1 \cdot \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \\ 0 \end{pmatrix} + l_2 \cdot \begin{pmatrix} p_A \\ p_B \\ p_C \\ p_D \\ p_E \end{pmatrix} + l_3 \cdot \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} l_2 p_A \\ \frac{1}{3} l_1 + l_2 p_B + \frac{1}{2} l_3 \\ \frac{1}{3} l_1 + l_2 p_C + \frac{1}{2} l_3 \\ l_2 p_D \\ \frac{1}{3} l_1 + l_2 p_E \end{pmatrix}$$

Which is the vector of probabilities from which we draw the 3rd word of the generated text.[3]

On a final note, for general text predictive purposes l_1 and l_2 should be small and l_3 should be close to 1. This is the case because of two reasons. First, deviations from the original text occurs generally at the sentence-level in Mark V. Shaney while at the individual-word level in the other two models. This means that deviations that occur because of introducing the HMM and the simple Monte Carlo tend to accumulate: over a given set of words, the probability of all words not deviating is guaranteed to be relatively small compared to the probability that at least one word deviates to a word with zero probability assigned in Mark V. Shaney.

Second, once the sequence of words deviate twice in a row, the Mark V. Shaney vector of probabilities is very likely zero for the next word: if no third word follows from the current and previous generated word, even a highly-weighted Mark V. Shaney probability vector will have no bearing on the new generated word. This means that if the sequence of words deviates to a two-word group which is not the first two elements of a three-word group in the text, the algorithm will continue deviating until some iteration produces a non-empty Mark V. Shaney probability vector.

[3] In the case where one or two the probability vectors are all zero (when there is no word following the current word), we re-standardize the mixture vector in the implemented algorithm to maintain consistency. We note that the probability vector given by the emission matrix in Baum-Welch cannot be entirely empty. Therefore, $V^{mix} \equiv \mathbf{0}$ will not occur and a new word can always be drawn.

3 Data and Results

We perform our mixture model using two types of texts: song lyrics from Adele and Jason Mraz and six presidential debate transcripts from Bernie Sanders.[4] Prior to entering the model we examine the text, remove all punctuation except commas and periods, remove duplicate periods and change all words to lowercase. Note that we elect to not remove numbers in the Bernie Sanders debate transcript text. The Adele and Jason Mraz lyric texts are approximate 7,000 words in length with approximately 700-800 unique words. The Bernie Sanders debate transcript is approximately 13,000 words with some 1,900 unique words.

3.1 Song Lyrics

We performed our model on the song lyrics of these two artists: Adele and Jason Mraz for their different song genres using a weight set of (0.02,0.005, 0.975). Although some of their songs fall into the category of blued-eye soul, most of Adele songs are soul and R & B, while most of Mraz's are pop rock and Reggae. Compared to Jason Mraz who usually sings about love and hope, Adele sings mixed feelings about heartbreak.

	Jason Mraz	Adele
1	93 Million Miles	Chasing Pavements
2	Geek in the Pink	Hello
3	I Won't Give Up	Make You Feel My Love
4	I'm Yours	Rolling In The Deep
5	Life is Wonderful	Rumor has it
6	Make It Mine	Set Fire To The Rain
7	The Beauty in Ugly	Skyfall
8	The Remedy	Someone Like You
9	The Woman I Love	Turning Tables
10	Wordplay	When We Were Young

Table 1: Songs Names Used in the Analysis

The following is an example of our generated song lyrics for Adele.

```
"youd play . you played it , with a beating . throw your soul through every
open door woah . youre gonna wish you never had met me . the scars of
your despair . tears are gonna fall , rolling in the deep . we could
have had it all . youre gonna wish you never had met me . it all .
but my knees were far too tingle pavements when need your both build with
but million shy blue you at im rolling reminded thinking made it you
something all it . let the sky fall . when it crumbles . we will stand
tall .  i could stay there , close my eyes , feel you here forever .
you had my heart inside of your love , they leave me breathless .
i wish nothing but the best for youi ive made up my mind , dont need to do ,
if im wrong i am right , dont need to think it over , i must have called
a thousand times . to make you feel my love . theres a fire starting in
my heart drops , and my back begins to tingle ."
```

Here is an example of our generated song lyrics for Jason Mraz.

```
"oh , oh my beautiful mother . she told me , son sometimes it may seem dark ,
but see . i may be skinny at times but im fat fulla rhymes . pass me the mic
```

[4]Presidential debate transcripts are obtained from the American Presidency Project. http://www.presidency.ucsb.edu/debates.php

```
and im callin it my own . if we never get down it wouldnt be a let down . i
tried to be right back after this . unless the dreamer is the new color for
fall . im yours .  do you want to come on , scooch on over closer , dear .
i may be skinny at times but im still looking up . god knows im tough enough
i am tough , i heard two men talking on the fourth of july , freedom ring .
well if you wanna get free la mirror . and when youre needing your space ,
to share this view of the night alone . if we never get down it wouldnt be
the one to take you home . you might think about me . do you believe , im the
geek yo wont can amounts gots , im yours . well , they were counting down the
days to stab felt ."
```

Our results seem fairly reasonable as we have coherent sentences with high readability. Yet, these are not sentences that are directly copied from the ten Adele songs we got the lyrics from. While these generated lyrics are readable, one may notice that certain phrases are commonly grouped together which makes sense given that it is typical in songs for a certain word to have a limited amount of neighboring words. For example, the word 'set' may be seen in the phrase "set fire to the rain" and may not be surrounded by any other words. Naturally, this will be reflected in the lyrics our model generates.

3.2 Bernie Sanders Debate Transcripts

We present a sample of the most sentences generated from Bernie Sanders debate transcripts using a weight set of (0.1, 0.1, 0.9). Note that we capitalize words as needed and reformat spacing to improve readability.

```
I have 5 million individual contributors who have destroyed flint by a voice
vote on the last period to the top 1 percent, and I believe that the American
election manufacturer fears less. So, lets ban assault weapons, not just
talking about it,ending gun manufacturing in our country. Well, Clinton, who
said that there are bills in congress, and I had [an] that was horrific,
trying to desegregate the Chicago police for trying to rebuild water systems
all over this country. Well, when you watch these Republican debates, you
know, you know what, I will do it. And second of all, to meet with a
disappearing middle class to hold a town meeting.
```

We note that sentences are somewhat meaningful and there is evidence of local structure. However, there exists an issue with topics of the sentences: we can see that consecutive sentences cover distinct topics. One readily identifiable reason for this discrepancy in performance is the sheer complexity of presidential debates: our song lyric text has a unique-to-total word ratio of close to 15:1, whereas the corresponding ratio for debate transcripts is close to 7:1. This means that the transition matrices are much more sparse in the debate transcripts compared to those for the song lyrics. Hence, it is much more likely for transitions to be unique or for the Mark V. Shaney and simple Monte Carlo probability vectors to be empty.

The main difference between debate transcripts and song lyrics is that debates transcripts are often comprised of conversational English (and contain a non-trivial amount of typing errors), and hence often lack the structure and coherency of the written English language. This is because that our model doesn't correct for grammar or mis-spelled words. Furthermore, the procedure itself is very sensitive to wrongly spelled words since they will be considered as unique and appearing only once in the text. It is likely that actual speeches of candidates or written articles will provide more coherent output.

4 Discussion

4.1 Why mixture?

While our mixture model does not necessarily outperform a pure B-W model with a large number of hidden states, we identify three advantages of the mixture approach compared to alternative ways of generating text. First, for a reasonable amount of unique words in the the training text, the mixture model allows for improved local structure with a simpler HMM and therefore at least one order of magnitude less computational complexity. The pure B-W model has a computational complexity lower bounded by $2q \cdot nm^2$ for n words, m hidden states and q iterations. Our model given m' hidden states has complexity $2q \cdot nm'^2 + W^2 + W^3$ for total number of unique words W. For a 10,000-word model with 500 unique words (complexity of typical song lyrics or children's novels) , a 100-hidden-state pure HMM requires at least 2×10^9 computations for 10 iterations, whereas our mixture model requires only approximately 1.3×10^8 computations. Even if we expand the number of unique words to 1,000 (general news articles, presidential debates without specific names and terms), the mixture model with a 10-state HMM is still less than half as computationally costly as the 100-state pure HMM.

Second, our mixture model is versatile in the sense that components can not only be individually improved (introducing more hidden states to the HMM or limiting the HMM emission vector to only words that have non-zero transition probabilities with simple Monte Carlo) but also replaced or added as needed. For one example, introducing a fourth and fifth vector of third-order and fourth-order Monte Carlo transitions would allow us to not only introduce local structure but also tune the amount of local structure at different levels (phrases versus complete sentences). We note that if these vectors are updates as the text is generated, introducing them would not conceptually be highly costly as long as n is not extremely large.

We also note that we do not necessarily need to fix weights throughout the generating of the sequence. Conceptually, it may be desirable to alter weights for different types of words (such as noun vs. verbs). As an example, it may be desirable to very strongly weight the Mark V. Shaney vector for words that follow immediately after periods, because in English we generally have a notion of what is or is not proper at the beginning of a new sentence. Another example may be to de-emphasize the role of Mark V. Shaney for words immediately following conjunctions.

Third, by weighting inputs of multiple models *ex ante*, we introduce perhaps not-undesirable subjectiveness into the prediction of text. A pure HMM models does not allow for much control over the product: output text is taken as a given and the process is governed solely by the transition and emission matrices. While this may be ideal for some purposes, for others it may be useful to either directly judge the quality of output and alter the model as needed, or to tune trade-offs in terms of predictive ability in the model to better suit specific text applications.

It should be noted that it is also possible to obtain objective output from the mixture model by training a set of weights over a given criteria. In particular, general indicators of a text chunk's readability can be used to tune the weights using cross-validation or a accept-or-reject Metropolis-type algorithm. Section 4.2 discusses two potential approaches in depth.

4.2 Training Weights

Conceptually, several criteria can be used to measure the general readability and quality of generated text. The obvious example is Zipf's law (Zipf 1949), which states that the frequency of any word in a natural language is approximately its own frequency rank in large bodies of text. More generally, we note that for text of a broadly similar type (debate speeches to speeches in other debates), the frequency distribution of the most popular words (conjunctions, frequently used nouns and verbs) should be roughly similar.

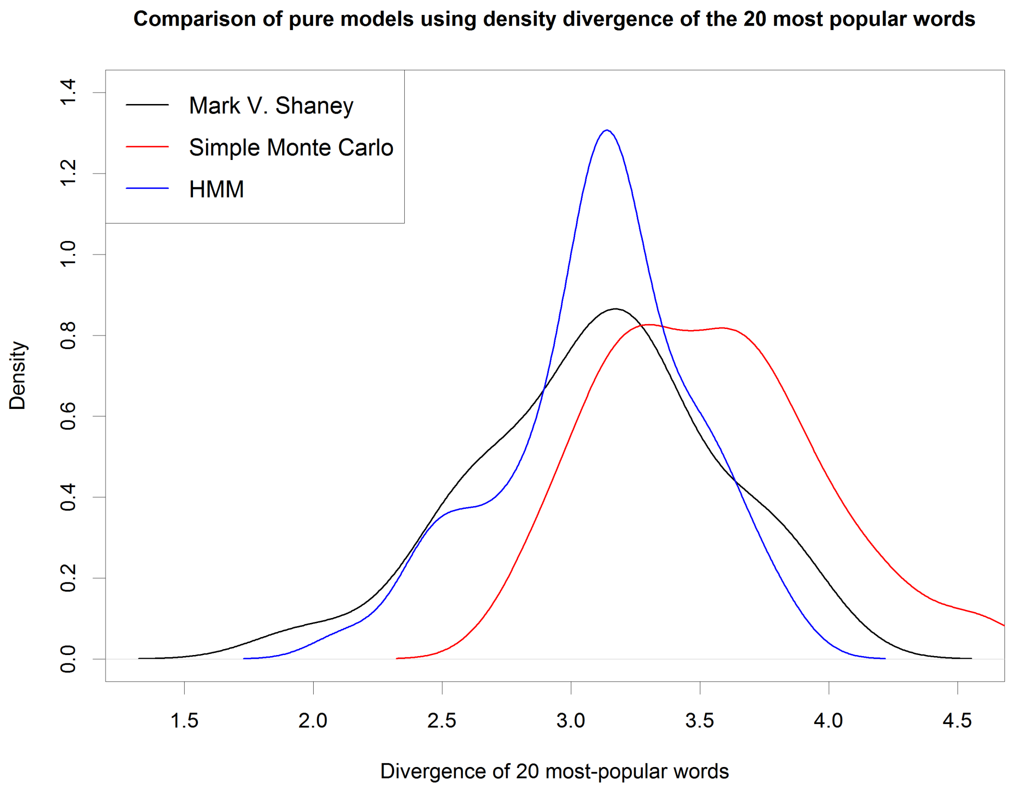
This leads to the first criteria we propose: for the mixture model and a given set of weights, we can divide the entire body of available text into training and testing sets, calculate the frequencies of the top C words and compare the respective distributions. Specifically, for a set of sorted, standardized word frequencies $(h_1, h_2, \dots h_C)$ in the generated text and a set of sorted, standardized frequencies $(h_1^*, h_2^*, \dots h_C^*)$, the density divergence is:

$$\alpha_C = \sum_{i=1}^{C} (h_i - h_i^*)^2$$

We note that what the ith most frequent word specifically is not important from a text prediction perspective: there are usually multiple words with similar levels of frequency in a given piece of text.

The key component in reducing this divergence is the correct proportionality of words inherently most frequent in all texts (“a", “the", “but", etc.), frequent words specific to the text in question (as examples, “inequality", “superdelegates" and “wage" in Bernie Sander’s debates), and general punctuation.

Using $C = 20$, we compute the distribution of divergence values α over 50 iterations of the text generating algorithm using generated text chunks of the same size as the holdout text. A small α indicates that the generated text and the holdout text are very ‘close’, and a large α indicates that the generated text and the holdout text are ‘far’. Results generated from the Bernie Sanders debate transcripts and a 10-state HMM are presented in Figure 1 below for each of the pure models:



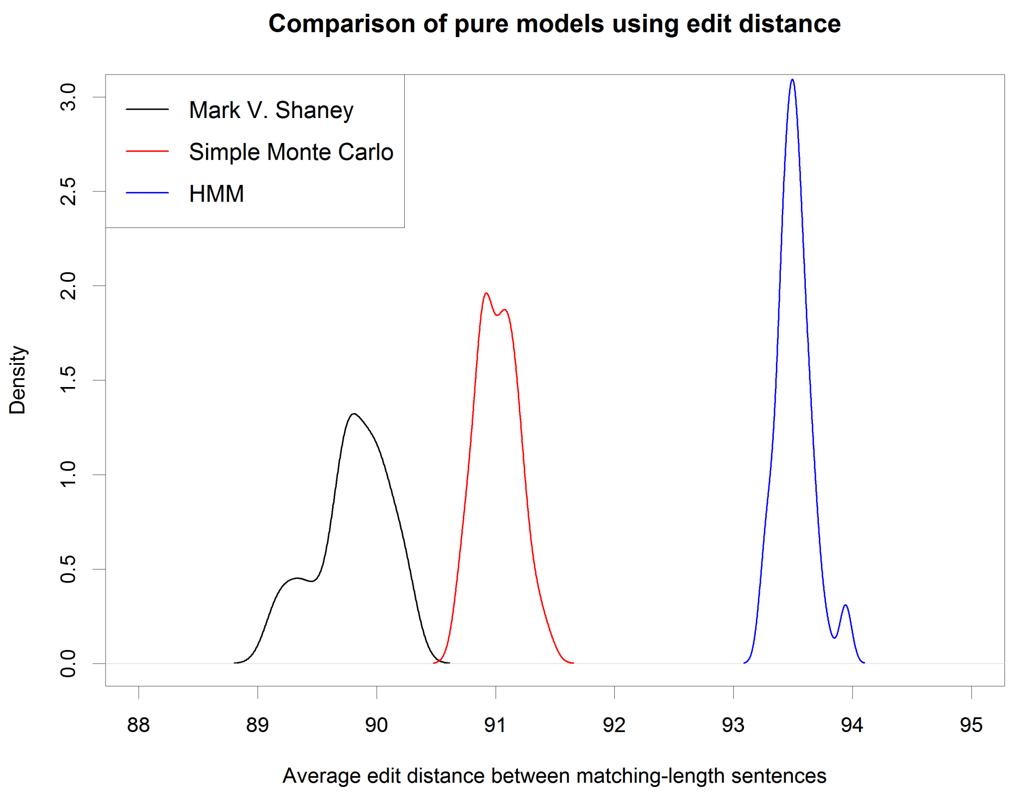
As indicated in Figure 1, both the Mark V. Shaney model and the HMM outperform the simple Monte Carlo in terms of divergence. The HMM distribution of divergence is denser, suggesting that it is potentially leveraging information of “ideas” in the text as reflected in the hidden states. However, the mean divergence remains higher than that of the Mark V. Shaney model.

Our second proposed criteria is based on the concept of Levenshtein (edit) distance (Levenshtein 1966; Wagner and Fischer 1974), which counts the number of insertions, deletions, and substitutions it would take to change one character string into another. A Levenshtein distance of 0 indicates that the two character strings are exactly alike.

For our example, we sort and order sentences in the generated text and the original, training text and compare sentences of unique lengths using edit distance. Note that if generated text chunks are very large and there are many sentences of identical length, some rule for breaking ties may be needed (distance of min-distance sorted pairs of equal-length sentences is an example). However, for our purposes we will simply sort unique sentences since it is unlikely that there are multiple sentences of identical length except for very short sentences.

We present distributions of the edit distance δ between the generated text and the original training text in Figure 2 using 50 iterations of generated text and the same specificities as in Figure 1. As indicated in the Figure, the pure Mark V. Shaney algorithm far outperforms both the simple Monte

Carlo and the HMM. The HMM actually performs worse than the simple Monte Carlo in this regard, most likely because the 10-hidden-state complexity is not sufficient to respect the original sentence structure and introduces an undesirable number of periods.



We note that both of these metrics are generally in favor of the Mark V. Shaney algorithm having a larger weight. This is not unexpected, given the fact that the original structure is not likely to be better-approximated by anything that isn't the original structure. This means that if only one out of the two criteria is used or if they are simply summed together, any iterative process that uses an accept-reject criteria will converge towards the weight (0,0,1) where all weight is placed on Mark V. Shaney.

We experiment with several alternatives for a Metropolis-style training algorithm using automatically-determined discounting of the Levenshtein distance as well as a single tuning parameter s for a composite distance criteria of $\alpha - s \cdot \delta$. While the results successfully converge to sets of weights reasonably quickly and generally emphasize the Mark V. Shaney model, we note that the output from the trained weights does not seem significantly better than weights determined subjectively by examining the output and changing weights according to the interpretation of the roles of the respective component models.

In a sense, this is also not surprising. Written language is complex and highly subjective in nature. What we broadly interpret as "sensible" is dependent not simply on sentence structure or using a correct number of frequent words but also involving grammar, proper emphasis and emotional depth. As humans, our judgment of whether a piece of text is sensible or not is most likely far superior than any algorithm without sufficient complexity. In this regard, the methods we propose are perhaps not enough for a rigorous test of "readability" that can be used as an accept-reject criteria. However, we are optimistic that this may be made possible by introducing other, alternative measurements of readability.

4.3 Applications

One interesting application of procedural text generation can be seen on Twitter currently @Deepdrumpf.[5] Created by Bradley Hayes, a postdoc at MIT, Deepdrumpf is a Twitterbot that is based on a deep-learning algorithm trained on transcripts of GOP presidential candidate Donald Trump's public speeches and debates. Given Trump's preference for simple and concise language, his speeches are perhaps particularly manageable to study. However, there is no reason why a similar approach cannot be applied to extract language patterns and parody other public figures such as career politicians and candidates of various types of elections.

Likewise, our model is applicable to replication text mining studies and, in particular, studies involving copyrighted texts. In text mining projects documents are often either copyrighted in entirety or with limit-access requirements (for one example, the text body of novels may fall under Fair Use for research purposes, but nonetheless cannot be made publicly and freely accessible). For projects involving such content, a simple, affordable way of reproducing text that meets given criteria could potentially allow for better replications of such studies and hence greater transparency in terms of research objectives.[6]

Yet another use of affordable procedural text generation is the creating of non-repetitive dialogues in games. The procedural generating of graphical and sound effect content has experienced a great surge of attention in game design in recent years. [7] Potentially, using an approach similar to our model on a given body of dialogue content creates unique variations that can be applied to reduce perceived iterativeness of non-player characters. While models with greater complexity such as those involving deep learning methods exist, our approach allows individual developers and small studios with limited resources to also create text via procedural generation.

5 Conclusion

Through the consideration of three distinct text prediction models, we successfully implement an algorithm that generates new content with more accuracy than any single component and is more computationally efficient than a comparable pure HMM model. We perform our model on two sample texts, respectively song lyrics from Adele and Jason Mraz and presidential debate transcripts from Bernie Sanders and discuss the output.

We further discuss potential extensions and applications of our mixture model, as well as potential methods of objectively determining the mixture weights via a Metropolis-based procedure. We conclude that the model has interesting applications for a variety of text prediction needs.

References

[1] Bahi, L.R. & Brown, P.F. &De Souza, P.V. & Mercer, R.L. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition.

[2]Baum L.E. & Petrie T. & Soules G. & Weiss N. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *Ann. Math. Statist*, 41, no.1 164-171.

[3] Baum L.E. & Eagon J.A. (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73, No.3 360-363.

[4] Dewdney A.K. (1989). Computer Recreations: A Potpourri of Programmed Prose and Prosody. *Scientific American* June 1989, No. 1, 122-125

[5]http://www.csail.mit.edu/deepdrumpf

[6]It could be argued that using randomized text (for non-classifed information) falls under similar protection as media content in photo collages for creative purposes (which is protected given certain constraints). However, as of this writing there does not seem to be legal precedent for publishing randomized text. We do note that "Works made through purely mechanical processes or with an automated selection and arrangement" is protected under current copyright law (USCO 2014).

[7]See `http://www.makeuseof.com/tag/procedural-generation-took-gaming-industry/` for a detailed discussion.

[5] Efros, A. A. & Leung, T. K. (1999). *Texture synthesis by non-parametric sampling.* The Proceedings of the Seventh IEEE International Conference on, Kerkyra, 1999, pp. 1033-1038 doi: 10.1109/ICCV.1999.790383.

[6] Fine, S. & Singer, Y. & Tishby, N. (1998). *The Hierarchical Hidden Markov Model: Analysis and Applications.* Kluwer Academic Publishers, Boston. Machine Learning, 32, 41–62 (1998).

[7] Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions, and reversals *Soviet Physics Doklady* 10, No. 8, 707-710.

[8] US Copyright Office (2014). *Visual Art Works* Section 906, Clause 906.6.

[9] Vitoria, N.G. & Abascal, J. (1995). Text prediction systems: a survey. *Universal Access in the Information Society*. March 2006, 4, No. 3, 188-203.

[10] Wagner R.A. & Fischel M.J (1974). The String-to-String Correction Problem *Journal of the ACM*, 21, No. 1 168-173.

[11] Zipf G. (1949). *Human Behavior and the Principle of Least Effort* Addison-Wesley. ISBN: 161427312X