Text Transformation Via Constraints and Word Embedding

Benjamin Bay, Paul Bodily, and Dan Ventura
Computer Science Department
Brigham Young University
Provo, UT 84602 USA
benjamin.bay@gmail.com, paulmbodily@gmail.com, ventura@cs.byu.edu

Abstract

In order to provide resources for artistic communities and further the linguistic capabilities of computationally creative systems, we present a computational process for creative text transformation and evaluation. Its purpose is to help solve the fundamental problem posed by the field of natural language generation, which is to computationally generate human-readable language. Our process entails the use of 1) vector word embedding to approximate meaning and 2) constraints to guide word replacement. We introduce *intentions* as objects that drive the generation of creative artefacts; a target theme, emotion, meter, or rhyme scheme may be represented via intention. Our implementation of this process, *Lyrist*, is oriented around poetry and song lyrics and successfully produces syntactically correct, human-voiced text. A preliminary evaluation suggests that our process successfully evokes human-recognizable sentiments and that even familiar texts are difficult to recognize after undergoing transformation.

Introduction

Language is an incredible tool used by humans. It expresses our most complex ideas and is woven into all of our creative tasks; the mediums of conversation, poetry, and song are built around it. As such, gaining power over language is integral to the problem of computational creativity (CC); two important goals are computational 1) understanding of language and 2) production of language (Bateman and Zock 2003). Natural language processing (NLP) and its subfield, natural language generation (NLG), address these problems.

Syntactically correct text is available in abundance and is easily produced by humans. However, maintaining such syntactic correctness and semantic cohesion in generative text is a major barrier within NLG. We bypass this challenge by transforming, rather than generating, texts. This allows us to modify individual words of an original text while maintaining its relative word relationships and original syntactical structure. In this paper we present a general framework for systems to 1) transform and 2) evaluate textual artefacts.

We present an implementation of this transformation framework, applied to the problem of generating poetry and song lyrics, as well as a protocol for creative intention.

Related Work

Our method of text transformation builds upon ideas from previous research in computational linguistics and CC.

Monteith et al. presented a successful process for creating melodic accompaniments based on input lyrics (2012). This approach to music generation assumes the availability of pre-existing lyrics. Our transformation process fills the essential role of lyric-generation for such systems.

Exploring the problem of automatic lyric generation, Oliveira, et al. built a system that generates lyrics with word stresses matching the rhythm of given melodies (2007). Later, Oliveira improved this system to generate text on a semantic domain using seed words (2015). For the sake of autonomy, we wanted to design a framework that did not necessarily rely on syllabic stress data to generate text.

Tobing et al. approached the problem of poetry generation via grammars and chart generation (2015). While their results are, for the most part, syntactically correct, they are not semantically cohesive, and their method is described as "a work in progress". Because of the complexity of generating speech via grammars, our framework currently avoids them altogether.

Colton et al. build a poetry generator that used poem templates and simile tuples of the form $\langle object, aspect, description \rangle$ (2012). While their simile knowledge base is high-quality, we seek a more comprehensive approach with regard to comparisons; in other words, we seek the ability to quantify the relationship between any set of words in the English language—by leveraging vector-based word embeddings for managing simile, analogy, and metaphor.

Toivanen et al. replace 50% of words in lyrical templates derived from existing songs (2013). Their method is simple, effective, and presents only low risk of losing morphological or global semantic cohesion. However, it only achieves limited transformation and limited creativity because it excludes prioritized constraints, the notion of intention, and the ability to estimate word meaning.

Hirjee and Brown use a confusion matrix of probabilities that any pair of phonemes will rhyme based on rhymes found in a corpus of hip-hop lyrics (2009). Such a matrix could be used to score potential rhymes between words; this is an intelligent approach. However, these probabilities are empirically derived, and we seek a more principled rhyme-

Input	Output
σ(walking)	running, stepping, hiking
τ(actress, mom, aunt, queen)	female, girl, woman
α(house, castle, roof)	parapet, battlements, spire

Table 1: Examples of word-vector operations made possible by word embedding. The spatial distribution of words in such vector spaces is such that nearby words are related, and basic vector arithmetic allows for computational approximation of word meaning. These operations accurately reveal basic relationships, such as "walking" being related to "hiking", or "female" being the common thread between "actress", "mom", "aunt", and "queen"; they also reveal interesting and imagerial relationships, such as the roof of a castle being a "parapet" or "battlements".

scoring algorithm rooted in phonology.

Gervás et al. discuss the challenges of automatic poem generation, and offer these classes of solution: understanding phonetics, using phonetic knowledge to drive poem generation, managing vocabulary, dealing with emotions, and managing comparison, analogy, and metaphor. (2007). We agree that these are among the primary challenges of poem generation (and more broadly, text transformation) and have implemented solutions in each case.

Finally, Gervás groups approaches to poetry generation into these rough categories: 1) template-based, 2) generate-and-test, 3) evolutionary, and 4) case-based reasoning (2002). Though not exclusive to poetry applications, our approach best fits into the first two categories. Rather than generate text, our framework transforms and evaluates it.

Framework

Our conceptual framework for the creative transformation of text includes a language model, a system of prioritized constraints, a text transformation process, and a protocol for computational intention. The language model is used to find appropriate replacements for portions of the text; the constraints filter these potential replacements and impose the intentions on the transformation process.

Language Model ($\mathcal{L}$)

The language model $\mathcal{L}$ is effected as a word embedding, which entails mapping words (usually from a large textual corpus) to vectors of real numbers in a multi-dimensional space such that collocated words are semantically related. More formally, $\varepsilon : W \to \mathbb{R}^n$, with W the set of all words and n the dimensionality of the embedding space.

Once such a vector space is constructed, the estimation of word meanings and relationships becomes possible using geometric operations, and the primary operations used here for text transformation are similarity σ, neighbor χ, theme τ, and analogy α. These are described in detail below and examples are provided in Table 1.

The similarity operator $\sigma : W^2 \to [0 \dots 1]$ is defined as

$$\sigma(w_1, w_2) = \left| \frac{\varepsilon(w_1) \cdot \varepsilon(w_2)}{\|\varepsilon(w_1)\| \|\varepsilon(w_2)\|} \right| \quad (1)$$

It computes the absolute value of the *cosine similarity* between the vectors associated with two words w_1, w_2, which, given the structure of the embedding space, is a good geometric surrogate for similarity of word meaning.

The neighbor operator $\chi : W \to 2^W$ computes a set of neighbors for a word w and is defined as

$$\chi(w) = \{y | v \in \mathbb{R}^n \wedge \sigma(v, \varepsilon(w)) < \theta \wedge y = \varepsilon^{-1}(v)\} \quad (2)$$

where θ is a threshold for controlling the number of neighbors and $\varepsilon^{-1} : \mathbb{R}^n \to W$ is a partial inverse function that extracts words from the embedding space. χ returns a set of words mapped to points in the space that are close to the point to which w is mapped. These neighbor words generally have similar usages as or associations with w.

The theme for a set of words Y is computed with the operator $\tau : 2^W \to W$, which is defined as

$$\tau(Y) = \varepsilon^{-1} \left(\zeta \left(\frac{1}{|Y|} \sum_{i=1}^{|Y|} \varepsilon(y_i) \right) \right) \quad (3)$$

where $y_i \in Y$, and $\zeta : \mathbb{R}^n \to \mathbb{R}^n$ maps vectors into the domain of ε^{-1}. That is,

$$\zeta(v) = \operatorname*{argmax}_{u \in \varepsilon[W]} \sigma(u, v) \quad (4)$$

where the notation $f[A]$ means the image of the set A under f (so, we are looking for vectors in the image of ε so that the inverse mapping ε^{-1} will be defined). τ finds the word whose vector embedding is closest to the centroid of all the vector embeddings of the words in Y; that is, it finds the "average" of the set Y, effectively summarizing the text represented by Y.

The analogy operator $\alpha : W^3 \to W$ is defined as

$$\alpha(a, b, c) = \varepsilon^{-1}(-\varepsilon(a) + \varepsilon(b) + \varepsilon(c)) \quad (5)$$

Because of the semantic structure of the embedding space, the output of this function is often a reasonable completion d for the analogical form $a : b :: c : d$.

Instead of training a language model (vector embedding) for every possible genre, dialect, and time period, we train one master model and filter its word suggestions with constraints.

Constraints ($\mathcal{C} = \langle \mathcal{C}_m, \mathcal{C}_r \rangle$)

Two types of constraints are used to filter potential solutions for the transformation process. Marking constraints, $\mathcal{C}_m$, determine which words in a text will be transformed, and replacement constraints, $\mathcal{C}_r$, determine which candidate replacements are valid. In practice, both types of constraints may be implemented as a conjunction of Boolean predicates, $\mathcal{C}_m = \{P_i^m\}$ and $\mathcal{C}_r = \{P_j^r\}$.

As an example, consider the case of marking constraints composed of single compound predicate $\mathcal{C}_m = \{P^m\}$, where

$$P^m(w) = \begin{cases} \text{TRUE} & \texttt{noun}(w) \lor \texttt{verb}(w) \lor \texttt{adj}(w) \\ \text{FALSE} & \text{otherwise} \end{cases}$$

that picks out those words that are either nouns, verbs or adjectives. Then, if the set of replacement constraints $\mathcal{C}_r$ were the following (in priority order):

$\mathcal{C}_r = \{$

1. spelling $\neq$ original spelling
2. lemma is not to equal to any other new lemma in the transformation
3. not on restricted list
4. found in an English dictionary
5. part of speech is identical to original's part of speech

$\}$

then for each noun/verb/adjective, a replacement word must be found that meets all of the criteria listed above: it must be a different word, must not have already been used, must not be blacklisted, must be English, and must have the same part of speech.

Transformation ($\mathcal{T}$)

Given a text T to be transformed, the set of marking constraints $\mathcal{C}_m$ determines which words $M \subseteq T$ will be replaced. For each word $m \in M$ to be replaced, the language model $\mathcal{L}$ is used to find a set of potential replacement words R, and each suggestion $r \in R$ is then filtered with a prioritized list of replacement constraints $\mathcal{C}_r$. Thus, the transformation $\mathcal{T}(T) = T'$ is computed as follows

$$M = \{t | t \in T \text{ and } \bigwedge_i P_i^m(t)\}$$

and for $m \in M$

$$R_m = \mathcal{L}(m)^1$$

and for $r \in R_m$

$$S_r = \{r | r \in R_m \text{ and } \bigwedge_i P_i^r(r)\}$$

In the case that $S_r = \varnothing$, replacement constraints can be weakened or dropped in reverse order of their priority until the set is no longer empty. In the case that $|S_r| > 1$, one member of S_r can be chosen probabilistically. The final result is the transformed text T' in which each marked word $m \in T$ has been replaced by a word m' that fits the language model $\mathcal{L}$ and meets the constraints in $\mathcal{C}_r$.

Intention ($\mathcal{I}$)

Just as an artist puts careful consideration into a creative task before it is carried out, creative systems may possess objectives which influence the eventual creation of artefacts. We call such objectives *intentions*. For our purposes, an intention has the following properties:

- Determinative - may be used to direct the creation of an artefact
- Evaluative - may be used to evaluate an artefact
- Selective - may be applied globally or locally within an artefact
- Conjunctive - may be combined with other intentions

We identify the following three classes of intention as applicable in the artefact generation context:

- Thematic intention ($\mathcal{I}_t$) - the semantic purpose of the artefact (e.g., subject, emotion).
- Cultural intention ($\mathcal{I}_c$) - the sociocultural context for the artefact (e.g., language, movement, genre).
- Structural intention ($\mathcal{I}_s$) - the target organization or arrangement of an artefact (e.g., technique, rhyme scheme, meter).

A tuple of sets corresponding to the above three classes may be used to represent overall intention for an artefact:

$$\mathcal{I} = \langle \mathcal{I}_t, \mathcal{I}_c, \mathcal{I}_s \rangle. \tag{6}$$

Intention can be imposed on the system through design decisions (e.g., which corpora to use in training the language model $\mathcal{L}$), additional replacement constraints (e.g., rhyming, syllable count), parameter selection (e.g., thematic seed words to be used with the α operator), etc.

Inspiration refers to the method or source from which intention is determined: a system with immutable intentions determined by the designer may produce interesting artefacts, but it will be considered less autonomous (and thus likely less creative) than a system whose intentions are mutable and that defines its intentions via some other form of inspiration.

Implementation

We built a poem and lyric transformer as a proof of concept for the transformation framework, and named it *Lyrist*.

We used Word2Vec's continuous bag of words learning model (Mikolov et al. 2013) to construct the embedding for our language model $\mathcal{L}$. We used a dimensionality of $n = 500$, and a window size of 5. Figure 1 shows the composition of the corpus used for training the model, by type of text. We sought to maximize the quantity of artistic and spoken texts, and use a large quantity of text overall, deriving our corpus from a data set of pop song lyrics, a data set of poems, COCA (Davies 2008), the NOW Corpus (Davies 2016), and Wikipedia.

In Lyrist, each word object carries the attributes listed in Table 2. We use Stanford CoreNLP (Manning et al. 2014) for part of speech tagging, named entity tagging, and lemmatization. Parts of speech are drawn from the Penn Treebank tag set (Marcus, Marcinkiewicz, and Santorini 1993). We use the CMU Pronouncing Dictionary (Kominek and Black 2004) to assign words phonemes and stresses. Phonemes are represented by ARPAbet, a phonetic transcription code designed specifically for English.

[1]We abuse notation here to mean that one or more operators associated with $\mathcal{L}$ are applied to m.

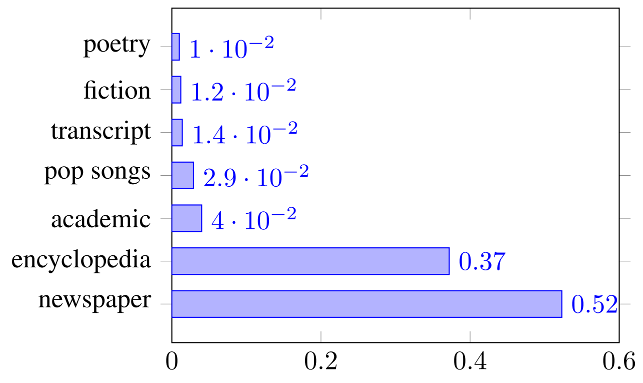

Figure 1: Proportions of text types used in the language model $\mathcal{L}$ of Lyrist. We consider creative texts (i.e., poetry, song lyrics, novels) and transcripts of actual human conversations to be the best training corpora for artistic text transformation purposes. However, these types of text are difficult to accumulate on a large scale, so we also include texts more accessible in bulk, such as academic papers, magazines, newspapers, and encyclopedias.

Lyrist makes use of both thematic and structural intentions; we currently implement the cultural intention of "English" only, leaving a cultural database and a more thorough treatment of intention for future work. Its intentions are currently human-defined, and we leave the implementation of a system on inspiration also for future work. To support the structural intentions, we implemented both a syllable parser and a rhyme scorer, in order to gain control over meter and rhyme. Here we provide details on the specifics of our implementation.

Syllabification A word is a sequence of syllables. A syllable is made of an onset ω, nucleus ν, and coda κ. The nucleus is the central vowel phoneme. The onset is the consonant phoneme(s) preceding the nucleus. The coda is the consonant phoneme(s) following the nucleus. Both the onset and coda may be empty.

Our syllable parser uses the fourteen phonotactic rules of English (Harley 2006) to syllabify sequences of phonemes. The algorithm works by:

1. labelling a word's nuclei ν
2. prepending onset phonemes ω to each nucleus ν while no phonotactic rule is broken
3. appending remaining coda phonemes κ to each nucleus ν while no phonotactic rule is broken

We do not use a dictionary for syllabic data. Hence, certain words of non-English origin are occasionally syllabified incorrectly.

Rhyme Rhymes are evaluated at the phoneme level by making use of a word's *stress tail*, defined as the nucleus and coda of the syllable with the greatest stress along with all following syllables. Our rhyme scorer works by:

1. extracting the stress tail from two words
2. aligning the stress tails' syllables
3. aligning the onset, nucleus, and coda of each syllable
4. scoring each aligned phoneme pair
5. scoring each syllable pair based on phoneme scores
6. scoring the word pair based on (stress tail) syllable scores

Phonemic scoring is computed one of two ways, depending on whether the phonemes are consonants or vowels. Consonant phonemes are characterized by three attributes: 1) voicing v, 2) place of articulation p, and 3) manner of articulation m. *Voicing* refers to whether vocal chords are used to pronounce a phoneme and may be represented as a Boolean value. *Place of articulation* refers to the point of contact where an obstruction occurs in the vocal tract to produce a consonant phoneme and can take one of seven nominal values: *alveolar*, *palatal*, *bilabial*, *velar*, *labiodental*, *interdental*, *glottal*. Finally, *manner of articulation* refers to the configuration and interaction of the tongue, lips, and palate when forming a consonant phoneme and can take one of seven nominal values: *fricative*, *stop*, *nasal*, *affricate*, *liquid*, *semivowel*, *aspirate*. Two consonant phonemes c_1 and c_2 are scored according to how well these three attributes match:

$$R_c(c_1, c_2) = \beta_v \delta_{v_1 v_2} + \beta_p \delta_{p_1 p_2} + \beta_m \delta_{m_1 m_2} \tag{7}$$

where δ_{ij} is the Kronecker delta function. We use the weights $\beta_v = 0.1$, $\beta_p = 0.45$, and $\beta_m = 0.45$

Vowel phonemes are similarly defined by three attributes: 1) voicing v, 2) frontness f, and 3) height h. *Frontness* refers to the distance from the back of the mouth when a vowel phoneme is formed. *Height* refers to the height of the tongue when a vowel phoneme is formed. Two vowel consonants u_1 and u_2 are scored according to (normalized) Euclidian distance between points of the English IPA vowel chart (Association 1999) (see Figure 2) superposed on the

Spelling	"Saturdays"
Lemma	"Saturday"
Phonemes	S-AE-T-ER-D-AY-Z
Syllables	S-AE-T · ER · D-AY-Z
Stresses	1 · 0 · 0
Stress tail	AE-T · ER · D-AY-Z
Part of speech tag	NNS
Named entity tag	WEEKDAY

Table 2: Example of useful data about the word w = Saturdays. We use phonemes, syllables, and stresses for phonetic constraints such as rhyme or alliteration. Part of speech tags allow us to constrain word replacements to those of identical part of speech. Named entity tags allow us to constrain word replacements to those of the same word family. Lemmas allow us to equally transform all words with the same base, rather than all words of this one specific form.

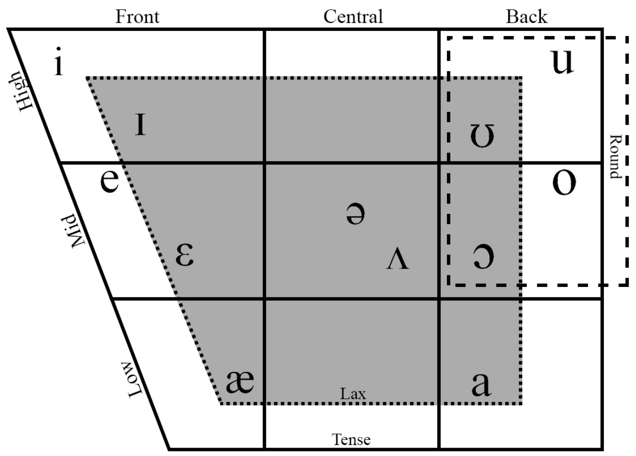

Figure 2: For rhyme scoring purposes, we estimate vowel similarity by finding the distance between phonemes on the standard IPA English Vowel Chart (1999). We superpose this chart on the Cartesian plane, with the horizontal axis measuring frontness and the vertical axis giving height.

Cartesian plane with axes frontness and height (voicing is ignored):

$$R_v(u_1, u_2) = \frac{1}{z}\sqrt{|f_1 - f_2|^2 + |h_1 - h_2|^2} \tag{8}$$

where z is a normalizing constant.

The equation for obtaining the rhyme score R_s of two syllables s_1 and s_2 is

$$R_s(s_1, s_2) = \beta_\omega \bar{R}_c(\omega_1, \omega_2) + \beta_\nu R_v(\nu_1, \nu_2) + \beta_\kappa \bar{R}_c(\kappa_1, \kappa_2) \tag{9}$$

where $\bar{R}_c$ means the average consonant phoneme score, in case the onset or coda contain multiple (consonant) phonemes. To amplify the relative importance of the nuclei, we use the weights $\beta_\omega = \beta_\kappa = 0.125$, and $\beta_\nu = 0.75$.

Finally, the equation for obtaining the rhyme score R for words w_1 and w_2, whose stress tails contain n_1 and n_2 syllables, is

$$R(w_1, w_2) = \frac{1}{n}\sum_{j=1}^{n} R_s(s_{1_j}, s_{2_j}) \tag{10}$$

where n is the greater of n_1 and n_2, and syllable s_{i_j} is the jth syllable in the stress tail of word i. To handle the case that $n_1 \neq n_2$, we define $R_s(s_1, -) = R_s(-, s_2) = 0$. Table 3 provides an example of this rhyme scoring process.

Output

Textual transformation can be used for various purposes, including to:

- alter a text's sentiment
- alter a text's meter
- introduce alliteration into a text

		stress tail———			
w_1	W	IY	P	IY	NG
w_2	N	IY	D	IY	NG
Phoneme rhyme scores	n/a	1.0	0.45	1.0	1.0
Syllable rhyme scores			0.93		1.0
Word rhyme score					0.97

Table 3: Example of rhyme scoring between the stress tails of "weeping" and "needing". This rhyme is imperfect (rhyme score < 1) due to the phoneme substitution of 'D' for 'P'. Since one is voiced and the other is not, they do not share the same place of articulation but they do have the same manner of articulation, their (consonant) rhyme score is 0.45. In our rhyme algorithm, we exclude initial phonemes that are not part of the stress tail from alignment and scoring.

- alter a text's rhyme scheme
- alter a text's rhymes while preserving its rhyme scheme

Inspecting actual output from Lyrist is useful in understanding the transformation process. Some helpful indicators of success in transformed creative artefacts are:

- Adherence to intentions $\mathcal{I} = \langle \mathcal{I}_t, \mathcal{I}_c, \mathcal{I}_s \rangle$
- Adherence to constraints $\mathcal{C} = \langle \mathcal{C}_m, \mathcal{C}_r \rangle$
- Syntactic correctness (grammar)
- Semantic cohesion (beyond syntactic correctness, does the text carry meaning and seem human-written?)
- Obfuscation of identity of source

As an example, given the following text,

Sorrow's my body on the waves
Sorrow's a girl inside my cake
I live in a city sorrow built
It's in my honey, it's in my milk[2]

and the marking constraints $\mathcal{C}_m = \{P^m\}$, where

$$P^m(w) = \begin{cases} \text{TRUE} & \texttt{noun}(w) \vee \texttt{verb}(w) \vee \texttt{adj}(w) \\ \text{FALSE} & \text{otherwise} \end{cases}$$

the following (bold) words are marked for replacement:

Sorrow**'s my **body** on the **waves
Sorrow**'s a **girl** inside my **cake
*I **live** in a **city sorrow built***
*It's in my **honey**, it's in my **milk***

Then, given the thematic intention
$I_t = \langle w_{t_o} = \text{"sorrow"}, w_{t_n} = \text{"honor"} \rangle$
and the structural intention
$I_s = \langle R_S = AABB \rangle$ and the replacement constraints

[2]from *Sorrow* by The National

$\mathcal{C}_r = \{$

1. spelling $\neq$ original spelling
2. lemma is not to equal to any other new lemma in the transformation
3. part of speech is identical to original's part of speech
4. rhyme score of matched line ending words is 1
5. syllable count of original and replacement word is equal

$\}$

the transformation produces the following:

> ***Honor****'s my* ***heart*** *on the* ***crests***
> ***Honor****'s a* ***woman*** *inside my* ***zest***
> *I* ***bide*** *in a* ***land honor divined***
> *It's in my* ***nectar****, it's in my* ***wine***

However, if the structural intention were then changed to
$I_s = \langle R_S = ABAB \rangle$
the result becomes

> *Honor's my heart on the* ***tides***
> *Honor's a woman inside my* ***roast***
> *I bide in a land honor divined*
> *It's in my nectar, it's in my* ***toast***

Sentiment is easily modified through transformation. Consider the following source text:

> *I loved beating these two terrible human beings. I would never recommend that anyone use her lawyer, he is a total loser.*[3]

With the following intentions and constraints,
$I_t = \langle w_{t_o} =$ "anger", $w_{t_n} =$ "kindess"$\rangle$
$\mathcal{C}_m = \{\sigma(w, w_{t_o}) > 0.5, (\texttt{noun}(w) \vee \texttt{verb}(w) \vee \texttt{adj}(w)\}$
this tweet's theme is changed from "anger" to "kindness". Only words similar to "anger" are marked, guaranteeing that the angriest words will be replaced and leaving more neutral words like "human" as they are:

> *I* ***hated*** *beating these two* ***profound*** *human beings. I would never* ***refuse*** *that anyone use her lawyer, he is a total* ***sweetheart****.*

This transformation adheres to its intentions and constraints, retains syntactic correctness, and is semantically cohesive. The new text's overall tone and meaning, though awkward in places, has a distinctly kinder feel.

Alliteration is a poetic device that is simply achieved through transformation. Given the following text,

> *Now folds the lily all her sweetness up* [4]

these intentions and constraints
$I_t = \langle w_{t_o} =$ "nature", $w_{t_n} =$ "desert"$\rangle$
$\mathcal{I}_s = \langle$alliterate$\rangle$
change the theme of this line of poetry from "nature" to "desert", while constraining for maximum alliteration:

> ***Currently cracks*** *the* ***cactus*** *all her* ***creaminess*** *up*

Again, this transformation adheres to its intentions and constraints, retains syntactic correctness, and is semantically cohesive. The words "cracks" and "cactus" have distinct associations with "desert", and 50% of words match each other alliteratively.

Texts using imagery and/or stream-of-consciousness may be successfully transformed with relatively few constraints. Given this source text:

> *Let us go then, you and I,*
> *When the evening is spread out against the sky*
> *Like a patient etherized upon a table;*
> *Let us go, through certain half-deserted streets,*
> *The muttering retreats*
> *Of restless nights in one-night cheap hotels*
> *And sawdust restaurants with oyster-shells:*
> *Streets that follow like a tedious argument*
> *Of insidious intent*
> *To lead you to an overwhelming question...*
> *Oh, do not ask, "What is it?"*
> *Let us go and make our visit.* [5]

the following text is the result of a transformation with the thematic intention to transform from love to mystery ($I_t = \langle w_{t_o} =$ "love", $w_{t_n} =$ "mystery"$\rangle$):

> ***Watch*** *us* ***proceed*** *then, you and I,*
> *When the* ***morning*** *is* ***transmitted*** *out against the* ***horizon***
> *Like a* ***glaucoma trapped*** *upon a* ***riddle****;*
> *Watch us* ***proceed****, through* ***unclear quarter shrouded alleyways****,*
> *The* ***mumbling mysteries***
> *Of* ***unconvinced evenings*** *in one-****evening inexpensive resorts***
> *And* ***styrofoam eateries*** *with* ***mussel-bomblets****:*
> ***Alleyways*** *that* ***after*** *like a* ***laborious*** *question*
> *Of* ***pernicious intention***
> *To* ***advantage*** *you to a* ***vast conundrum****...*
> *Oh, do not* ***determine****, "what is it?"*
> *Watch us* ***proceed*** *and* ***disentangle*** *our* ***trip****.*

[3]Trump, Donald (@realDonaldTrump). 23 May 2013, 4:24 p.m. Tweet.

[4]Excerpt from *Summer Night* by Alfred Tennyson

[5]First stanza of *The Love Song of J. Alfred Prufrock* by T. S. Eliot

The primary flaw of this transformation is occasional syntactic incorrectness. The eighth line, where "follow" is replaced with "after", is one such example. Errors such as this are preventable only with more powerful and more specific part of speech parsers. Part of speech code sets such as Penn Treebank are quite general and do not include data such as transitivity for verbs, number for pronouns, etc.

As a final example consider the following song lyrics, transformed by finding similar words with no overarching theme while maintaining rhyme scheme:

Lights go out and I can't be saved
Tides that I tried to swim against
Have brought me down upon my knees
Oh I beg, I beg and plead
Come out of the things unsaid
Shoot an apple off my head
And a trouble that can't be named
The tiger's waiting to be tamed[6]

Bulbs run *out and I can't be* ***hauled***
Currents *that I* ***attempted*** *to* ***run enthralled***
Have ***drawn*** *me down upon my* ***thighs***
Oh I ***sue****, I* ***sue*** *and* ***prize***
Come out of ***ways paternal***
Launch *an* ***orange*** *off my* ***colonel***
And a ***danger*** *that can't be* ***convened***
The ***tigress's awaiting*** *to be* ***weaned***

Discussion

In a survey conducted for preliminary evaluation of Lyrist, thirty-nine participants responded to questions pertaining to artefact 1) quality, 2) theme, and 3) source identification.

To measure quality, participants rated four textual artefacts using a Likert scale ranging from 1 to 5. The mean score was 3.05, with a standard deviation of 0.24, suggesting mainly neutral reactions to the transformed texts. One possible explanation is that "overall quality" is too general a measurement of success for creative artefacts. In the future we plan to have respondents evaluate rhyme schemes, rhyme choice, imagery associated with new theme, etc. according to a well-defined rubric for each of these categories.

To measure thematic accuracy, we transformed one original piece six times, each time using a distinct emotion as the new theme w_{t_n}: "excitement", "dignity", "love", "confusion", "paranoia", and "sadness". These were arbitrarily chosen and could have been replaced by other single-word themes. Details of the transformation $\mathcal{T}$ were kept private from participants. For each transformed piece, participants selected one emotion from the six as the emotion it best evoked. Table 4 gives the results in a confusion matrix. Overall the correct emotion was identified over two-thirds of the time. This suggests that drawing analogies from word embeddings is a good way to reflect thematic intention $\mathcal{I}_t$ in textual artefacts. Indeed, the overall correctness result was not higher because "love" was confounded with "excitement", and "confusion" with "love". It can be argued that the emotions "love" and "excitement" share many characteristics; this can be similarly argued for "confusion" and "love". Such confounding of similar themes is to be expected to some extent and may even be considered as evidence for the semantic quality of the language model $\mathcal{L}$.

	E	D	L	C	P	S
Excitement	**.904**	0	.048	.048	0	0
Dignity	.05	**.9**	0	0	0	.05
Love	.444	0	**.5**	0	0	.056
Confusion	0	0	.368	**.421**	.053	.158
Paranoia	0	.05	0	.35	**.55**	.05
Sadness	.053	.053	0	0	0	**.894**

Table 4: Sentiment confusion matrix for correct sentiment identification by survey participants. Certain sentiment pairs, such as dignity and confusion, were never confounded. But other sentiments, such as love and excitement, were confounded frequently. Column sentiments are abbreviated for legibility. The correct sentiment guess rate is bolded along the diagonal.

Participants were shown transformations of four popular English songs and asked to guess the title or artist of the original text. Afterwards, participants indicated which pop songs from a larger list they were familiar with, including the four original songs. Figure 3 shows the results. Guesses were counted as incorrect if the participant admitted to being familiar with a song but was unable to identify it after transformation. Overall two-thirds of guesses were incorrect, suggesting that the identity of even familiar texts is obfuscated after transformation. We see this as generally advantageous; a CC process is more likely to be deemed creative by unbiased observers if its technical specifics are not apparent.

Of course, even though this process guarantees accurate usage of constraints and mapping from original to new texts, it necessarily bears the combined error rates of its third-party dependencies, such as part of speech parsers. These errors can cause loss of syntactic correctness and semantic cohesion. After cleaning one's data, this condition can only be improved by improving the third-party tools themselves.

Additionally, an inherent flaw of string-based word embedding is relating words of identical spellings with different meanings, such as the noun and verb both spelled "wind". Any word with multiple definitions is probably less-than-ideally represented in such models.

Conclusion

Text transformation via constraints and word embedding is simple in theory, and powerful in practice. This concept builds upon research in computational linguistics and CC. Textual transformation requires large amounts of textual data for its language model and may additionally benefit from a large pool of original texts to transform. Word embedding allows computer systems to estimate word meaning and suggest analogous word replacements. Prioritized

[6] *Clocks* by Coldplay

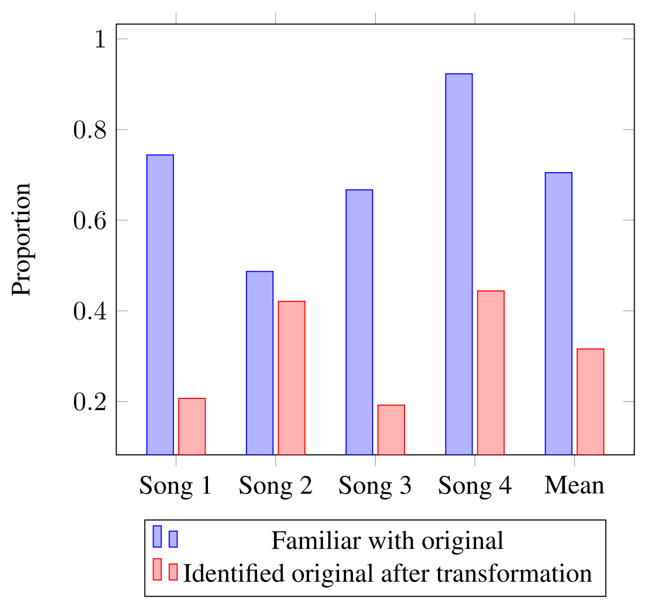

Figure 3: Proportion of participants that correctly identified original pop songs after textual transformation. Though most participants were familiar with the original text of four famous pop songs, most were unable to identify them from their transformed counterparts. This suggests that text transformation with Lyrist obfuscates the identity of texts well.

constraints deterministically filter out unwanted replacement words. A mechanism for representing intention provides a motivating goal, and the potential for inspiration driving this intention admits the possibility of greater system autonomy in the future. The process obfuscates textual origin and produces generally cohesive, thematically accurate text.

In the future, we plan to build an interactive web tool to showcase creative intentions and this transformation method. We also plan to construct a cultural database that allows the creation of artefacts incorporating cultural intention $\mathcal{I}_c$. We will explore pairing Lyrist with a system that uses an English grammar to break original texts into sentences, clauses, and phrases, allowing text transformation to function independently of complete human-written texts. Lyrist is currently being integrated with a musical artefact generator (Bodily, Bay, and Ventura 2017). We will explore the results of this pairing in future work.

References

Association, I. P. 1999. *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press.

Bateman, J., and Zock, M. 2003. Natural language generation. In *The Oxford Handbook of Computational Linguistics 2nd edition*. Oxford University Press.

Bodily, P.; Bay, B.; and Ventura, D. 2017. Human-level concept learning in computational creativity. In *Proceedings of the Eighth International Conference on Computational Creativity*, to appear.

Brownstein, J.; Yangarber, R.; and Astagneau, P. 2013. Algodan publications 2008-2013. *Journal of Intelligent Information Systems* 1–19.

Colton, S.; Goodwin, J.; and Veale, T. 2012. Full FACE poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, 95–102.

Davies, M. 2008. *The Corpus of Contemporary American English*. BYE, Brigham Young University.

Davies, M. 2016. The Corpus of News on the Web.

Gervás, P.; Hervás, R.; and Robinson, J. R. 2007. Difficulties and challenges in automatic poem generation: Five years of research at UCM. *e-poetry*.

Gervás, P. 2002. Exploring quantitative evaluations of the creativity of automatic poets. In *Proceedings of the Workshop on Creative Systems, Approaches to Creativity in Artificial Intelligence and Cognitive Science, 15th European Conference on Artificial Intelligence*.

Gonçalo Oliveira, H. R.; Cardoso, F. A.; and Pereira, F. C. 2007. Tra-la-lyrics: An approach to generate text based on rhythm. In Cardoso, A., and Wiggins, G., eds., *Proceedings of the 4th International Joint Workshop on Computational Creativity*.

Gonçalo Oliveira, H. 2015. Tra-la-lyrics 2.0: Automatic generation of song lyrics on a semantic domain. *Journal of Artificial General Intelligence* 6(1):87–110.

Harley, H. 2006. *English Words: A Linguistic Introduction*. Blackwell Publishing Ltd.

Hirjee, H., and Brown, D. G. 2009. Automatic detection of internal and imperfect rhymes in rap lyrics. In *10th International Society for Music Information Retrieval Conference*, 711–716.

Kominek, J., and Black, A. W. 2004. The CMU arctic speech databases. In *Proceedings of the Fifth ISCA Workshop on Speech Synthesis*.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J. R.; Bethard, S.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (System Demonstrations)*, 55–60.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics* 19(2):313–330.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Monteith, K.; Martinez, T.; and Ventura, D. 2012. Automatic generation of melodic accompaniments for lyrics. In *Proceedings of the Third International Conference on Computational Creativity*, 87–94.

Tobing, B. C., and Manurung, R. 2015. A chart generation system for topical metrical poetry. In *Proceedings of the Sixth International Conference on Computational Creativity June*, 308–314.