Raging with the Machine in the Uncanny Valley: Human–AI Cocreativity in the Eurovision-Themed AI Song Contest

Alexandra L. Uitdenbogerd,* Oliver Bown,† Charlton Hill,‡ Caroline Pegram,‡ Justin Shave,‡ and Brendan Wright§

*School of Computing Technologies
Royal Melbourne Institute of Technology University
GPO Box 2476, Melbourne, Victoria 3001, Australia
†School of Art and Design
University of New South Wales
Oxford Street and Greens Road
Paddington, New South Wales 2021, Australia
‡Uncanny Valley
200 Crown Street, Darlinghurst, New South Wales 2092, Australia
§Faculty of Engineering
University of New South Wales
Sydney, New South Wales 2025, Australia
sandra.uitdenbogerd@rmit.edu.au, o.bown@unsw.edu.au, {charlton, caroline, shave}@uncannyvalley.com.au, brendan.wright@unsw.edu.au

Abstract: We report here the processes involved in creating our entry in the 2020 AI Song Contest, "Beautiful the World"; the technical innovations from the project; and the decision-making that divided tasks between human and machine in a way that ensured that the final creation was AI-inspired but human-created, starting from generated melodies, lyrics, and timbres. Key innovations include the use of lyric stress patterns as queries to a stress-based melody index to a database of generated melodies, and the creation of a novel instrument timbre with differential digital signal processing, trained on Australian animal calls. We reflect on how human–AI cocreativity occurred during the process and how it may develop in the future.

The Dutch public broadcaster VPRO held an AI Song Contest in 2020, in which teams from Eurovision-eligible countries were invited to submit a Eurovision-style song created with AI. This provided an opportunity for new collaboration in the area of human–computer cocreativity, with teams containing a broad range of relevant skills, such as machine learning (ML), music composition and production, music information retrieval, and natural language processing. All these skills contributed to the entry our team submitted to the contest.

Our goal for the project was to create a Eurovision-influenced competitive entry in the AI song contest. In terms of research motivations, as proposed by Pearce, Meredith, and Wiggins (2002), the primary motivation was composition, with the secondary motivation being the development of tools for composers. A key difference in our approach, compared with those of other teams in the contest, was the automatic generation of artificially sung fragments from the best matches between generated lyric lines and melody snippets, based on the similarity of their stress patterns, simplifying how songs are constructed from generated material.

In this article, we present our project as a case study in which we ask how the approach and techniques used to create the song support cocreativity. We discuss the goals and techniques of the project, as well as the human–human and human–AI

Computer Music Journal, 47:1, pp. 44–63, Spring 2023
doi:10.1162/COMJ_a_00674

cocreativity as played out in the project, with reflection on the future.

Literature Review

Past research into algorithmic composition, generative systems, and human–AI cocreativity are all relevant to this project. We present past research on these topics in the rest of this section.

Algorithmic Composition and Generative Systems

Algorithmic composition is nearly 1,000 years old, with examples going back to the work by Guido d'Arezzo around the year 1026 (cf. Edwards 2011), where vowels of a text were mapped to note values. This was simple rule-based composition. Michael Edwards also notes that in Mozart's time, the game of "musical dice" was played, effectively being a forerunner of stochastically generated music. Naturally, the use of neural networks for music generation has a much shorter history. Although computers were used to play music as early as 1950 (cf. Doornbusch 2004), the first known computer-based algorithmic compositions were Caplin and Prinz's computer implementation of musical dice in 1955 (cf. Ariza 2011), "Push Button Bertha" in 1956 (Klein 1957) and Hiller's Illiac Suite from 1957 (Hiller and Isaacson 1958). As noted in Chris Ariza's article, Caplin and Prinz also went on to generate melodies using transitional probabilities, much like approaches based on Markov chains. The first published use of neural networks for generating music appeared in 1989 (cf. Fernández and Vico 2013).

There are several possible ways to approach the generation of songs with lyrics. The lyrics can be used to influence the rhythm of the melody, such as Nick Collins's (2016) use of stress patterns extracted from lyrics triggering rules for syllable locations within melodies. Ackerman and Loker (2017) used lyrics from MusicXML files to train random forests for a system that generates melodies from lyric input. An alternative is to search a database of indexed generated melodies for a suitable match to a given lyric, or vice versa, which is the approach taken for our project.

Human–AI Cocreativity for Popular Music

Nicholas Davis (2013) coined the term "human–computer cocreativity" to refer to interacting with a computer to achieve creative aims, where "the contributions of human and computer are mutually influential, which is the nature of collaboration and improvisation." Prior to the competition held in 2020, the state of the art in related AI–human cocreativity for popular music was such that multiple songs had been produced, including the Beatles-like "Daddy's Car" (Ghedini, Pachet, and Roy 2016) and the Eurovision-inspired "Blue Jeans and Bloody Tears" (Hadas 2021). The Australian music and technology collective Uncanny Valley created an AI Christmas carol in which melody and lyric lines were generated using neural networks trained on Christmas carol repertoire (the song can be heard at https://youtu.be/XruXCyrzI7Y). In all these cases, the music was generated from symbolically encoded musical data. In contrast, Zukowski and Carr (2018) used neural networks to produce streams of audio in the style of specific bands. Although there was little structural cohesion to the output, and no formal evaluation, the authors of that paper noted in further publication that the resulting timbres were most successful for genres that appreciate noise and abrupt shifts in rhythm (Carr and Zukowski 2018). Cocreativity for this endeavor was not explored until the 2020 competition, where the expanded team created a work that involved interpretation of generated audio and "riffing" on the result, which had a profound impact on the final work (Huang et al. 2020). A more adventurous audio-based offering is Holly Herndon's (2019) album *Proto*, discussed in her doctoral dissertation with the same title.

Goals

The regulations of the AI Song Contest stated that the AI judging panel and the general voting audience had equal weighting. The panel's assessment was

to be based on how the Eurovision datasets (lyrics and MIDI data) provided to contestants were used, whether the song structure was interesting, and the extent to which melody, harmony, lyrics, and audio were generated—although later there was greater emphasis on creativity (Micchi et al. 2021). The audience voting was based on "Eurovisionness," originality, lyrics, and the song in general. Thus, to be competitive required creating an appealing song suggestive of Eurovision style, using AI and algorithmic techniques to a substantial extent.

Creative Goal

Our main goal was to create something that would be reflective of the spirit of Eurovision and be uplifting and fun, to match our team's philosophy. We also wished to acknowledge the loss of Australian wildlife during the 2019–2020 bushfire crisis in the music, to celebrate this wildlife and the resilience of nature.

Division of AI and Human Labor

Through our previous AI song project we learned to see the quirky beauty, and also the occasional amazing insights, in machine-generated content. We wanted the algorithms to seed everything we did, intending to set up the models to generate melodies and lyrics, and then work with the models' outputs. Rather than laboriously combining the generated melodies and lyrics by hand, we intended to automatically match them based on lyrical and melodic stress. By automatically generating synthetically sung versions of these matched snippets, manual selection would be simpler and based on a smaller set of options. These would be auditioned to find earworms that would serve as a creative seed.

With recent advances in transformer-based generative learning algorithms for language modeling and text generation (Radford et al. 2019), we knew that lyric generation had the potential to produce rich sentence constructions both profound and amusing. We made this central to our process, giving the track meaning, albeit having fun with the quirks of language generation. Team members chose a set of words and short phrases suggestive of Eurovision themes to be used as seeds to the lyric generator.

Instead of automating harmonization, we made use of multiple generated phrases from the system to inspire harmonic layers.

Even though neural networks work well with lower-level melodic, harmonic, and rhythmic content, long-term structural arrangements are creative decisions that are still difficult to generate (Ji, Yang, and Luo 2023). We chose to leave structural and production decisions in the human domain, by producing or remixing the raw material of the machine output, as we would in typical production tasks.

Technical Goals

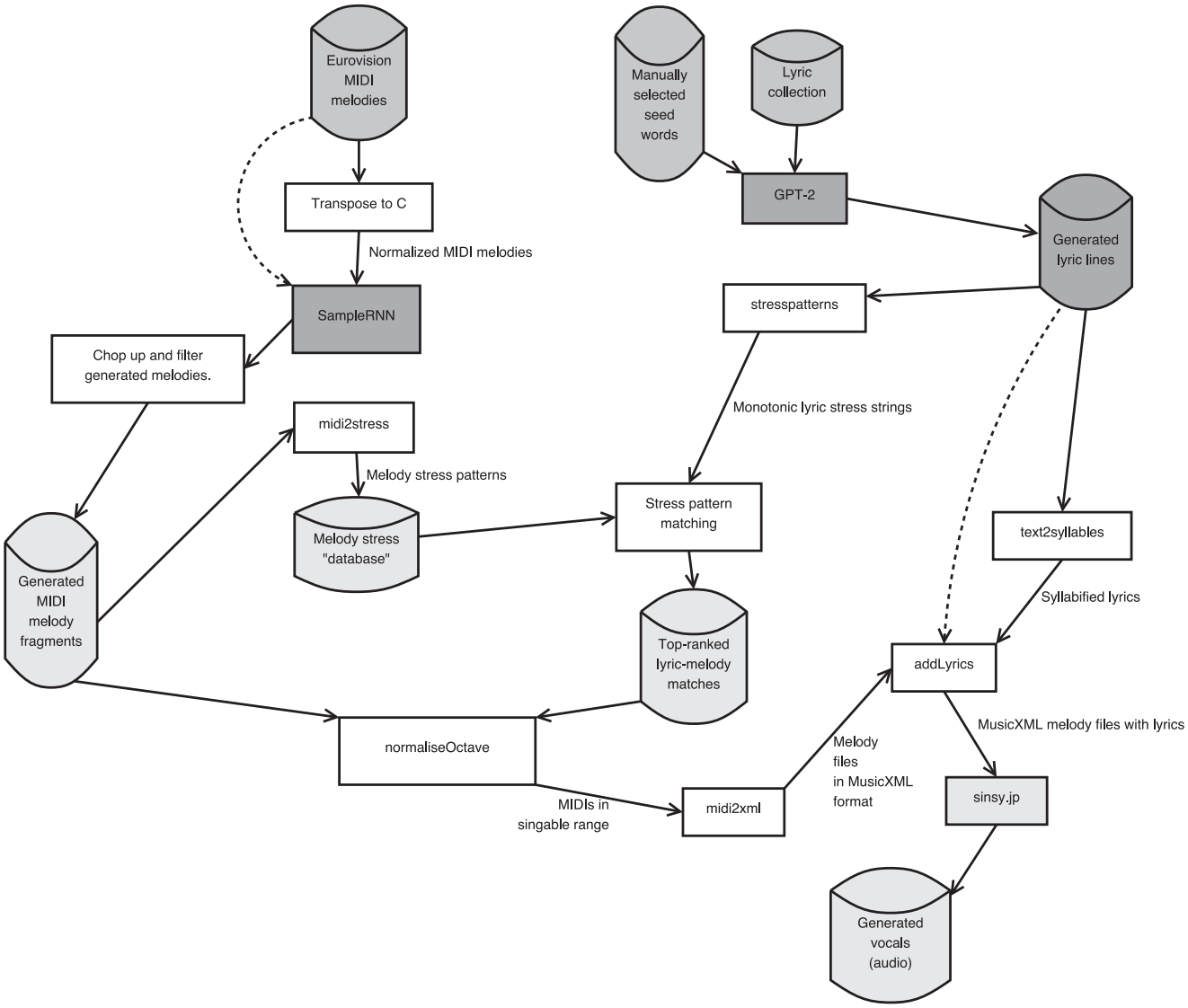
We have worked with a range of AI techniques and wanted to combine them:

1. Spleeter, for source separation (Hennequin et al. 2020);
2. A fine-tuned GPT-2, for lyric generation (Radford et al. 2019);
3. A modified sample-recurrent neural network (RNN), for melody generation (Mehri et al. 2017);
4. Differential digital signal processing (DDSP), for audio gesture and melody mapping (Engel et al. 2020); and
5. Convolutional representation for pitch estimation (CREPE), for melody and pitch detection (Kim et al. 2018).
6. Sinsy, for vocal generation (Oura et al. 2010)

We wanted to use AI techniques not only in composition but in production. Thus we planned to use the DDSP machine learning model, trained on audio samples of Australian animals, to allow us to map melodies into their voices. This tool helped us move beyond the symbolic domain of MIDI generation by incorporating audio-level production tools.

Prior experience with the sinsy.jp online service for vocal synthesis (Uitdenbogerd 2019) led to the use of automatically rendered vocal lines. These were

Figure 1. Process for creating the sung melody snippets. The dotted arrows indicate the process before all modules were implemented.



used both for evaluating generated song snippets as well as for the final production.

One novel approach was to automatically match generated lyrics to generated melodies, and then rank the results. A technical goal of the project was to develop a stress-based pattern-matching technique to find melodies and lyrics that would go well together. This would allow the output from both melodic and lyric generation to be filtered significantly, leaving the team to select from a smaller, hopefully more useful, output set.

Techniques

Figure 1 illustrates the various components of the system that produced the sung melody snippets

for consideration by the production team. In this section we describe the techniques used.

Lyric Generation

Lyrics were generated directly using a single learning algorithm featuring a standard GPT-2 model architecture (Wolf et al. 2020). A pretrained model was used to ensure basic language-modeling ability, with further training on a text dataset comprising a nearly complete set of Eurovision lyrics of about 80,000 lines for fine-tuning. The set of lyrics provided by the contest organizers was appended, so that these songs were effectively used twice as often in training as the other songs, in some cases in both original and English-translated form. Some preprocessing was performed to remove unnecessary special characters, punctuation, and header information. A more general set of lyrics (Smith, Zee, and Uitdenbogerd 2012) was also trialed for fine-tuning.

We implemented the model in PyTorch (an open-source ML framework providing tools and libraries for developing and training neural network models, cf. Paszke et al. 2019), with the pretrained GPT-2 language model and tokenizer, using an input block size of 512. Fine-tuning was performed using the Adam optimizer (Kingma and Ba 2015). This is an efficient general-purpose algorithm for gradient-based optimization of stochastic objective functions, such as those used during the training of neural networks via back propagation. We used a learning rate of 5×10^{-5}, no rate decay, epsilon at 1×10^{-8}), a batch size of four (limited by available GPU RAM), and trained for a total of ten epochs over the dataset (the complete dataset of Eurovision lyrics).

The trained model was then used for lyric generation, where a list of 20 seeds, consisting of single words (or, in some cases, short phrases such as "ding a dong") was used to prompt model inference, yielding generated lyrics with a maximum output character length varying between 50 and 85 characters, depending on the run, breaking at end-of-line characters, and performed in batch to return 64 generated sequences per seed. The set of seed words, chosen by the team, was used to seed the model generation on a line-by-line basis, and minor output filtering was performed on the generated song lyrics to ensure adequate length (i.e., more than two words).

Melody Generation

The model for melody generation was trained directly from MIDI files, transcribed from about 200 Eurovision songs with an average length of about two minutes each, read directly during training without any timing normalization (all the source data had the same number of ticks per beat).

The symbolically encoded melodies were generated with an implementation of the SampleRNN model architecture (Mehri et al. 2017), which we modified to produce symbolic data instead of audio. Our model used three hierarchical frame tiers featuring RNNs at each tier above the base (sample-level) multilayer perceptron layer. We wrote programs to convert MIDI melodies into sequences of monophonic MIDI note values for input into the model and vice versa for use in the matching process. Temporally, these sequences contained one data sample per MIDI tick (at 1,024 ticks per quarter-note beat), with silence encoded as zeros. Sequences were entered into the model in rolling segments of approximately one second, the top tier reaching up to about 30 seconds.

The modified SampleRNN model (MelRNN) implemented in PyTorch featured an input bit depth of seven (128 MIDI notes) passed as a single one-hot-vector (i.e., a vector in which all elements are set to zero except for a single element set to one) per time step, with an input frame-scaling factor of four between each tier, and with dual hidden-layer RNNs of dimension size 256. Weight normalization was used, and an initial hidden state was learned during training, using the Adam optimizer (learning rate 1×10^{-5}) and an input sequence length at MIDI-level of 64 with a batch size of 32, obtaining melody generation results sufficient for our purposes after a total of 100 epochs. Less training yielded unusable outputs, whereas further training tended to bias the model towards sparse outputs that significantly

lowered the hit rate without noticeably improving the quality of melodies generated.

Few modifications were made to the base SampleRNN model, but the input and output stages were heavily tuned to melody timing rather than audio sample rate (e.g., 48 kHz).

To improve the tonality of the generated melodies, the entire dataset was "key-normalized" (i.e., transposed to C major or A minor) at the input stage, which was found to greatly improve generative results without the need for postprocessing.

Output generation was performed from an initial silent state as seed, with a learned initial hidden-state vector, and propagated one data sample at a time for up to 30 seconds, producing the generated melody. The resulting batch of generated melodies was segmented around long silences and large jumps in pitch (defined as an interval larger than twelve semitones), to maintain internal consistency, and finally filtered to ensure that overall length, pitch range, and note count were suitable for vocal alignment. After segmentation and filtering, and a final conversion to MIDI format, we obtained the complete melody dataset. Thus, from a set of 1,024 outputs of up to 30 seconds in length, a total of 576 MIDI outputs with an average length of 5 sec ± 70 percent remained after the segmentation and filtering process, being approximately 10 percent of the original output.

Hardware Resources

Training time for learning algorithms (after training data had been preprocessed) was about one hour for the GPT-2 lyrics model and about one and a half hours for the Melody RNN model on consumer-grade hardware (Nvidia RTX 2070 8Gb OC). Optimization and testing of models and training protocols for generating melody and lyrics, however, exceeded five hours each (excluding time for development and configuration of novel aspects of model architecture).

Matching Lyrics to Melodies

Combining melody and lyrics was achieved by indexing hundreds of generated melody fragments for matching based on lyric pronunciation stress and melodic stress. To match text and melody stress, both need to be mapped to the same representation for scoring on similarity. In this section we describe two mappings.

Table 1. Lyric Stress Encoding

Syllable Stress	*Code*	*Example*	*Pattern*
High	1	black	1
Medium	2	everything	142
Low	4	the	4

Stress patterns are strings of digits 1, 2, and 4, with one digit per syllable.

Lyric Stress Patterns

Each word in the generated lyric lines was automatically converted to a syllabic stress pattern according to the Carnegie Mellon University Pronouncing Dictionary, CMUdict (http://www.speech.cs.cmu.edu/cgi-bin/cmudict), which provides a three-level stress code. For example, the word "beautiful" is represented in the dictionary by:

```
B Y UW1 T AH0 F AH0 L
```

The digits occur only where there is a syllable vowel, and for this example there are three. The first syllable (shown as `UW1`) indicates a strong stress, whereas the other two syllables have weak stress. Only a small proportion of words have a third level of stress. An example that occurs in our song is "everything." Words with only one syllable are mostly given a high stress digit, except for some short function words that tend to be unstressed in speech, such as "the." See Table 1 for examples.

Extracting the stress codes for each word, and using the digits 1, 2, and 4 for stress level (with 1 representing the strongest stress), resulted in stress strings for each lyric line, which could then be used for matching. For example, the stress pattern for "everything is black and white together" is represented by the string 1421141414.

Table 2. Melodic Stress Encoding

Note Length in Beats	*Stress Code*
> 1	1
$= 1$	2
< 1	4

Melody Stress Patterns

Musically, stress is defined in several ways. For Western classical music in common time, the default stress pattern for the four beats of a meter is defined as Strong-Weak-Medium-Weak. In performance, notes that fall on those beats receive that level of stress, and notes on half beats and smaller will have progressively smaller stress weights—see, for example, the entry on metre in the *Oxford Companion to Music* (London 2011). Changing this default stress is achieved with accents on notes or syncopated rhythms.

In the absence of defined measures, a probable stress pattern can be deduced from the relative durations of notes, with long melody notes having a higher stress than short ones. In our case, the generated melodies did not conform to a regular meter, being an unstructured stream of notes, so each note in the melodic data was given a stress score based on its length.

To allow matching against lyrics, the same three-level representation was used, as defined in Table 2.

Pattern Matching

For a more natural-sounding sung melody, stressed syllables should occur on melody notes that have higher stress and unstressed syllables should be on unstressed notes. Where the melody stress symbols are the same as the lyric stress symbols in the representations defined above, the ranking of the symbol pair should be higher than where they differ.

Another aspect of the melody-to-lyric matching is that there need to be enough melody notes for the lyrics to fit. A first attempt at matching used *local alignment*. This tries to find the best local match within the strings, regardless of the content of the remainder of the strings being matched. This led to the top-ranked results being partial matches. For example, four syllables of a ten-syllable lyric line may have matched four notes of a melody line. For our application, this was less useful than a full match.

To enforce matching of the entire lyric line, scoring was therefore based on the *global alignment* of stress strings to produce a ranked list of candidate melody matches for each lyric. We used a simple scoring scheme in which matching characters received a score of 1, mismatches −1, and −2 for *indels* (i.e., insertions or deletions). Therefore there was a strong penalty for strings of differing length, at −2 for each extra character.

For example, the stress string 12412, when matched against 1242, would align from the first character of each, as shown below.

```
1 2 4 1 2
| | | - |
1 2 4 - 2
```

Because both strings must be entirely consumed during global matching, there will be at least one indel due to the difference in string length. The resulting score would be 2, from four matches and one indel.

Scores can easily result in negative values with this scoring scheme. For example, the score for a ten-character string aligned with an exactly matching five-character substring (e.g., 1244412421 to 12421) is −5.

After matching, we had a ranked list of melodic phrases for each lyric line based on stress.

Audio Generation: Voice

Rendering the lyric-melody pairs required allocating syllables to notes. Code written in Python allocated words to notes of a melody in turn, and if there were more notes than words, the words would be cycled through again from the start. Although this may seem like odd behavior, compared to how people write songs, it was due to Sinsy not producing clear notes when no syllable is present, and the original code being written to allow a word or short phrase,

such as "la" or "amen" to be easily repeated across a melody.

Although there was information from CMUdict allowing us to calculate how many syllables existed in each line and the syllable stresses for each word, the exact divisions of the words into syllables was not available. The first set of sung melodies did not divide words into syllables, and therefore polysyllabic words, such as "happiness" were allocated to a single note, and leftover notes received the words from the start of the lyric phrase. Later, a rough implementation divided words into their syllables by adding hyphens. Both sets of output were considered in constructing the song. Melisma (syllables sung over multiple notes) was implemented later but was not used in the submitted song.

The top two lyric–melody pairs for each lyric line were automatically combined into MusicXML files and rendered using the sinsy.jp online singing-synthesis service, which uses hidden Markov models to generate sung text in English. Thus, hundreds of musical fragments, consisting of audio files of artificially sung phrases, were generated for human evaluation and selection.

These sung snippets were not used in the final recording. Once a complete song was composed from the snippets, however, a higher-quality rendering of the sung components of the song was generated with Sinsy, with octave and tempo shifts as required for the final work.

Audio Generation: Koala Synth

We created a dataset of the sounds of some iconic Australian animals to be source material for DDSP-based timbral style transfer, producing a playable synthesizer based on the sounds of Australian wildlife. It was a way to include a homage to the bushfires and our native animals. It is estimated that about 4,000 koalas died in the New South Wales bushfires (Van Eeden et al. 2020), and the population in 2021 was down about 40 percent from 2018 levels (AKF 2021).

Initially tested with koala sounds, additional animal sounds were later included in the model. Table 3 lists the animals and the duration of the source audio data for each. Excerpts from each animal dataset are available, as indicated in the Appendix.

Table 3. Audio Source Data for Koala Synth

Animal Source	*Duration (mm:ss)*
Koalas	10:58
Kookaburras	15:00
Tasmanian devils	00:44

We used this synthesizer to play some of the significant melodies from the melody-generation step, most distinctly in the instrumental break after the first chorus. An excerpt of this is available, as explained in the Appendix.

Audio Generation: Extracted Vocals

The SampleRNN model was also used for generating audio from vocal stems from the past five years of Eurovision song entries, extracted using Spleeter, placed end to end and then sliced into eight-second increments.

Production

We selected the best melodic phrases by hand, based on their earworm quality, by listening to audio representations of the melody and lyrics. These were then arranged into a typical song structure. Human production, including human-played accompaniment and arrangement, was based around chordal movements suggested by the melodic fragments, much like "Push Button Bertha" (Ames 1987).

We approached the production as we would any other song to be produced for a specific purpose. When remixing or reversioning a song by an artist, one may simply be presented with a melodic line and attached lyrics. Everything else, such as production style, chords, and rhythms, is subject to the creativity of the producer and the needs of the commissioning party. In that sense the goal was to produce a Eurovision-style track, using the bare

bones provided to us by the AI; that is, lyrics and melody.

The vocals in the final track were a blending of the synthetic Sinsy-generated voices and human singers. The key was, first, to match the note transition and pitch accuracy of Sinsy with Melodyne—a tool often used to produce vocals in professional studios. The producer listened closely to the way a Sinsy line was constructed, then used Melodyne to precisely adjust the pitch, pitch transitions, and timing of the recorded live vocals. Each vocal melody line consisted of four tracks, that is, the melody was sung by the human singer and recorded four times. These tracks were closely tuned in terms of timing and pitch, to emulate and enhance the slightly robotic sound. The producer then layered the Sinsy vocal on top of the chorus vocal mix, which, having already been tuned and timed, melded well with the Sinsy vocal. Finally, various effects were applied, such as reverberation, tempo delay, harmonic distortion, equalization, and compression.

Evaluation

The process of generating the parts of the song was an interactive one, with humans "in the loop." There was a continuous evaluation of the outputs. It was not a requirement that the output be convincingly human; on the contrary, catchy output giving the impression that only a machine could have written it was considered favorably by the team. An informal criterion for judging the success of the song was "can you sing it around the campfire with a guitar?"

Given the goals to produce a competitive Eurovision-influenced song and develop new tools, our methods of evaluation can be summarized as: (1) apart from the critical reflection of the creative process, the composition's success was evaluated, based on the results of the competition; (2) the developed tools were analyzed based on an assessment of the output produced; and (3) use of the tools by the team as an instance of human–AI cocreativity was analyzed qualitatively via reflective practice (Candy 2019).

Results

In this section, we report observations regarding the generated melodies and lyrics, an analysis of the matched outputs used in the song, the outcome regarding other generated audio, the contest results, and how the techniques used in the song differed from other competition entries.

Melody Generation

We tested various values for the output resolution against the fixed input resolution, and therefore the beat timing, of the MelRNN model. We found our final selection to be a good balance between variability and coherence of the resultant output melodies, providing a greater number of usable source melodies to pass further into the production process.

It was observed that higher resolutions required more postprocessing to achieve a reasonable output, owing to frequent short jumps between notes, whereas lower resolutions tended to remain stagnant over long periods, with pauses exceeding a reasonable duration, and again requiring significant postprocessing.

This balance was also motivated by a desire to have MelRNN output be used in as raw a form as possible, with minimal postprocessing or filtering, and therefore more directly involved in the creative process.

Given more time for experimental optimization and testing, it is possible that a better solution could have been devised. We found this solution satisfied the requirements of our study, however.

Raw Materials for the Song

Rather than independently choosing lyrics and melodies, the two processes were intertwined via the matching process. Once the melody-, lyric-, and audio-generation systems were producing usable outputs, the team listened to the snippets of sung audio to find any that stood out as suitable "hooks" for the song. This then influenced the direction of

the song and the lyric–melody pairs that would be used.

Lyrics

The main lyric source material consisted of 980 lyric lines generated from 19 seeds. Lines varied in length from three words totalling ten characters ("love to me") to a truncated 91-character string ("oh ah la, oh ah la la, la la, la la la la la la la la la la la la la … la la la la la l" from the prompt "oh ah la"). An additional curated set of 68 lyric lines of interest from previous lyric-generation runs, some with different seeds, was also processed.

Some lines were repeatedly generated. For example, with the seed "open," "open the door" was generated four times on its own as a whole line, out of 62 lines. In contrast, the longest line for the same seed was "open up your heart, come on with me and live my life like a butterfly," which demonstrates the lack of coherence typical of many lines. Some selected seeds were not standard words in an English language dictionary and yielded the least coherent results. The seed "lulu la" produced phrases in several different languages, including Spanish (*lulu la gente pluto sincero*, rendered by Google Translate as "lulu sincere pluto people") and French (the somewhat more plausible *lulu la peut-être les visages* and the amusingly ungrammatical construction *lulu la loi je douze points*). But English-like phrases were also generated, such as "lulu la seduce: 'I'll give you one good night to stay and be with me' ").

Lyric–Melody Pairs

The melody database consisted of 461 generated melody snippets, ranging from eight four-note sequences to one that was 15 notes in length. The most frequent stress pattern among the melodies was a succession of seven short notes, occurring 30 times in the database. All lyric lines were posed as queries to the database of 461 melodies on the basis of their stress patterns. This matching process restricted the set of outputs to 171 melodies, with melodies being matched between 0 and 135 times to lyric lines in the dataset. Thus, team members could listen to an example of each melody to find the most promising candidates, and had multiple lyric lines from which to choose, based on the matching process. The lyric–melody pairs selected for the first verse and chorus all used melodies that were retrieved exactly twice from the lyrics query set. (The Sinsy audio output for five of the lyric lines are listed as part of the Appendix.) The second-top matched melody for each lyric line was also rendered, to give an alternative melody for audition.

Lyric lines varied in length from 3 to 36 syllables. Given that the longest melody was 15 notes, this prevented one-note-to-a-syllable matching in some cases. The possibility of breaking up a lyric line to send to the database for matching was discussed during development but not pursued. The opposite problem of too few text syllables was not a concern, as melismas could be used.

As the generated sequences of notes were not aligned to measures, they were placed into an artificially imposed, long measure length in the MusicXML format prior to being sent to Sinsy. This caused Sinsy to lengthen the final note to the end of an incomplete measure, making the resulting melodies more like typical sung melodic phrases, in that they were more likely to end in a held note (Tierney, Russo, and Patel 2011).

The selection of lyric lines from the generated output was entirely a human process, with the team constructing lyrics from the many options available. For example, the lines "flying from this world that has gone apart" and "flying in fear but love keeps on coming" were selected, whereas "flying like a trumpet" was not. As mentioned elsewhere, selection was not necessarily based on what was grammatical or convincing, but on humor and the concepts that were part of the team's vision.

Matching

Table 4 shows the lyric stress patterns for all lyrics from the final song that were presented to the matching algorithm, along with the stress patterns of the best matching melodies and the actual melody used in the final song. Lyrics are listed in the order that they first appear in the song.

Table 4. Lyric and Melody Stress Patterns

	Stress Patterns		
Lyric	*Lyrics*	*Melody*	*Comments*
Flying in fear but love keeps on coming	1441111114	4444444444	
Dreams still live on the wings of happiness	1111411144	4444424244	
Flying from this world that has gone apart	1411111141	4444444442	Used melody from first line
Open up the heart I ain't scared no more	1414111111	4444224442	
Welcome home oh welcome home oh oh oh	1411141111	4444444444	
The world is beautiful	411144	444244	
Ding a dong sweet song thank you darling	141111114	444444444	
Open the door no it won't open the door (door)	144111114411	444444244444	
The world we are the things you see in your eyes	41114111411	44244444442	
Dreams you know that you'll never know			
[Peace you know, then you never know]	11111141	4442441	
The world would be a better place	41114141	4442441	Used different melody
Dreams are cheap they can lead me crazy	111111114	442244444	
Break your heart! Their flames are vivid candles of hope	111111141411	444444442442	Generated melody truncated
The power of fire, the power of fire			
[Power and power of the fire]	144141414	444442444	
Dreams can save us			
[Dreams and dreams that could save us]	1411111	4444441	
The music of the earth has arrived	414141141	444442444	Spoken
Lulu la love love	14111	4414	Used different melody
Everything is black and white together			Used different melody
[Everything are black and white together]	1421141414	4424444444	

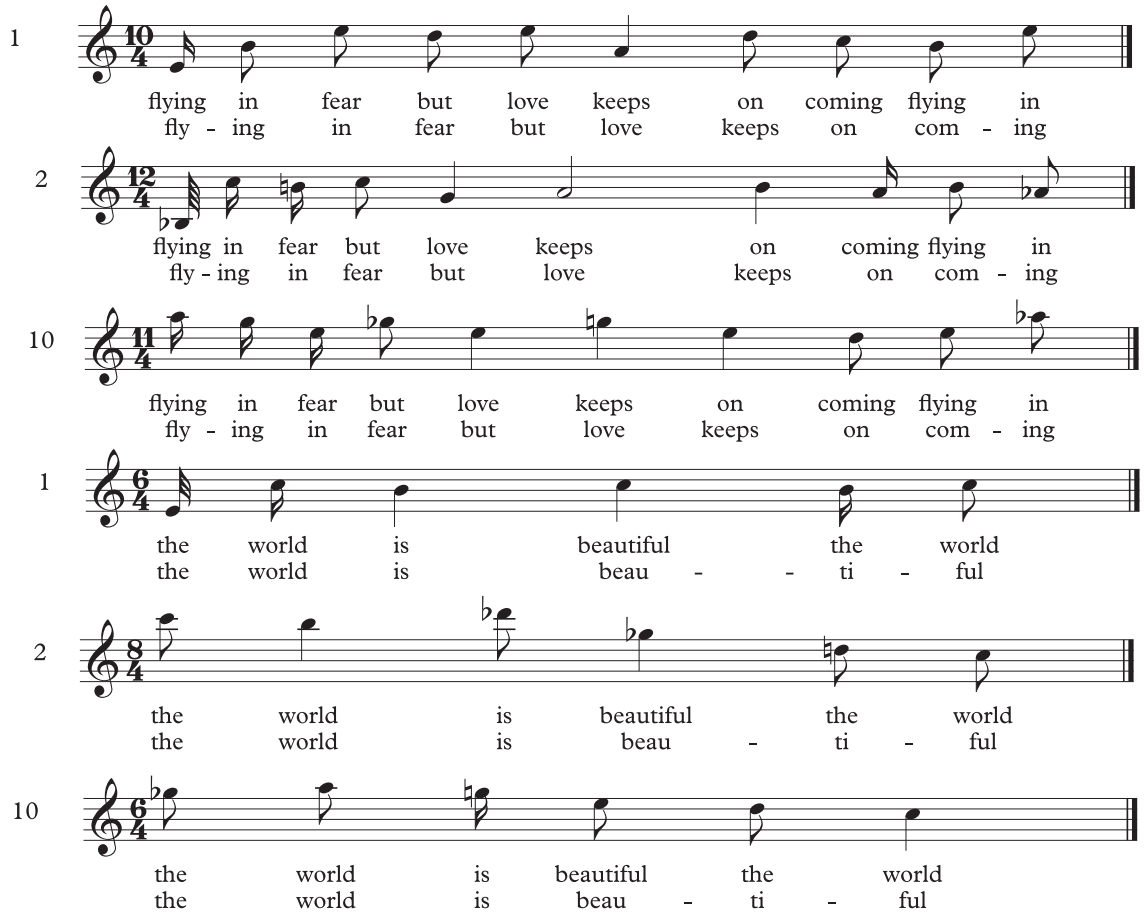
Over half of the lyric lines of the song were used as matched by the alignment algorithms. Some lyrics used other generated melodies (as indicated in the comments). Some lyrics were slightly modified, shown in italics above the unmodified original (in brackets), but kept the generated melody, except where noted.

The matched patterns clearly differ from each other greatly, apart from their length. Lyric lines of the same length were matched to different melodies. Some of these would have had the same match score. We did observe during the matching process (and prior to final selection of lyrics and melodies) that melodies would be matched to multiple lyric lines, which would allow some easily generated repetition—a feature of most popular music.

The generated melodic stress patterns tended to have a predominance of symbol 4 whereas the lyric stress patterns were dominated by symbol 1. This indicates that most notes were shorter than one beat. Indeed, although it was rare to hear a generated melody that was really slow, some melodies were too fast for vocals to have clear diction at the default tempo. On inspection, of the 461 melodies that constituted the final melody stress database, only 9 contained the symbol 1 at all.

Figure 2 shows the melodies that the matching algorithm ranked at positions 1, 2, and 10 (with 1 being the best), for two phrases that the team selected for the final song. The top line of lyrics shows how the incomplete algorithm allocated lyrics to notes, including the reuse of the first two words to ensure that all notes received words. The bottom line shows how they would have been allocated according to the stress-based matching algorithm.

Figure 2. Examples of algorithmically matched melodies, ranked 1 (best), 2, and 10. The first line of lyrics shows how lyrics were added by the incomplete algorithm, and the second line shows how they were automatically matched based on stress patterns. (Note that the time signatures displayed are arbitrary, chosen mainly so that each snippet could be notated in a single measure.)



Despite the limitations of this implementation, snippets were generated that inspired the team. We knew immediately when we heard the line "welcome home, oh welcome home, oh oh oh" sung by the computer that this was the hook and the theme of the song.

Other Audio

The koala synth was used for instrumental parts in the song, most notably in the lead break. As for the audio generated from the extracted Eurovision vocals, the quality of output in the separation of components, and resultant fidelity, was not high enough to satisfy our creative direction. So this aspect was abandoned.

Contest Strategy

Our strategy of using a few innovative techniques to address the judging criteria, combined with high production values to produce an audience-friendly song, resulted in high scores from both judges and voters. Table 5 shows the score breakdown for our song, "Beautiful the World," as well as the average and highest results across all competition entries. Audience votes were high across all criteria and the judges awarded the song the second highest score, equaling that of the entry by Can AI Kick It.

Burgoyne and Koops (2021) analyzed the voting in the competition, due to the concern about the effect of the large number of "groupie" votes, defined as voters who gave perfect scores to one song and did not vote for other songs. The reasons behind

Table 5. Average Votes

		All Songs	
Category	*Our Entry*	*Average*	*Highest*
Audience vote			
Song	2.6	1.67	2.6
Lyrics	2.2	1.52	2.2
Originality	2.3	1.79	2.3
"Eurovisionness"	2.7	1.54	2.7
Total	9.8	6.78	9.8
Judges' score	10	6.4	12

Audience votes were categorized as listed, the panel of judges gave only a single total score to each entry.

such votes can be blindly voting by friends of the team or voting once for the track that is most liked. Our track had the most groupies, based on that definition. However, Burgoyne and Koops noted that a corrected model that ignored the groupie votes did not change the rankings of the top-ranked songs.

Comparison with Other Entries

Huang et al. (2020) analyzed the approach taken by, and attitude of, each competition entrant. In the broad sense, none of the choices made by our team were unique, in that other entrants made similar choices. As with other teams that included professional musicians, a large proportion of our song components were human content. The most distinctive aspects of our approach were the use of stress-based matching of melody and lyrics and the use of DDSP to create a new instrument.

The marriage of lyrics and melody was handled differently by Team KTH/KMH+Doremir in that their system attempted to generate both components simultaneously, with each note leading to a duration prediction, which in turn would predict a text syllable (Bolcato 2020). As documented by Huang et al. (2020), the Doremir entry was one of only three, including ours, that did not manually align lyrics and melody. The third team conditioned their melody generation on lyrics.

Our process differed from the Algomus team, who had a more structured approach to creating their song. This consisted of the application of models to structure, harmony, and melody; from these, the team created subsequent short lists, and the final selection was decided by rolling dice, albeit with some aesthetic modifications as required (Micchi et al. 2021). Their initial lyrics were based on bigram frequencies in the Eurovision lyrics dataset, similar to the text-selection process of the algorithmic choral composition "World Cloud" (Uitdenbogerd 2019). However, the Algomus team then used the verses constructed from the bigrams as a seed for GPT-2 to create the rest of the lyrics, which then received minor tweaking. Sung melodies were composed by humans, but the instrumental hook was based on a model.

Another apparent difference found by Huang and coworkers was in attitude. Our team expressed exhilaration, as opposed to finding the process "painstaking, or similar to a difficult puzzle" (Team 1, as quoted in the study by Huang and colleagues). It is hard to say whether this was due to reducing tedium by automating the lyric-melody matching, our team's excitement in anticipating quirky output "gold," our delight in exploring the generated material, or a combination of factors.

Iteration and Changes in Direction

Due to the short timeframe, both pragmatic and aspirational decisions were made. Where some outputs did not match our vision, either algorithms were modified or techniques discarded. Evaluation during this process involved listening to and reading sample outputs, which led to more preprocessing of training data so that better quality output would be produced. The following changes occurred during development:

1. The key of input data to the melody SampleRNN was normalized to C major (or A minor), after initial dissatisfaction with the generated melodies.
2. Melody-lyric matching first used local alignment of word and melody stress pattern

strings. This led, however, to substring matches that were considered too short. So global alignment was used instead, to ensure there would be enough notes for lyric allocation.
3. Some generated melodies were not in a comfortable singing range, so automatic transposition based on average pitch was implemented.
4. Early runs of the melody-lyric snippet generation did not split words into syllables, despite matching being based on syllables. This created some interesting effects that were retained for the final song.
5. A method of audio generation involving vocal tracks from source-separated Eurovision songs was abandoned, because the results did not match our vision for the song.

As described above, there was iteration in much of the process of creating the song. Some subprocesses were added after earlier iterations showed their need, such as the Transpose-to-C process. It was clear that other processes were needed, such as splitting text into syllables, but they were not implemented until later iterations.

Discussion

We now look at the impact of the limitations of some on-the-fly design decisions, and we provide an analysis of the process in terms of cocreativity.

Melody Generation and Rendering

Several factors caused the generated melodies to diverge from conventional note durations and also to vary greatly in the range of durations. One was the high granularity of MIDI ticks used for input. Another was the variation in song tempi in the training data. According to the metadata provided to entrants, the tempi ranged from 60 to 194 bpm, with an average at around 110 bpm. One can imagine that the slower tempi songs may have had smaller note duration values than would be expected for a faster song. The renderings by Sinsy for audition were all at 100 bpm. Despite this variation in input tempo, individual generated melodies tended either to consist mostly of short notes, or to have a more relaxed sequence of notes. So some context seems to have been learned by MelRNN.

The free tempo in the rendered melodies meant that applying them to the final song required some interpretation in terms of how they should fit in a four-beat-per-measure song structure. Given more time, we might have explored the use of quantization to reduce the resolution and to observe its effect on outputs. Instead, pragmatic solutions were found to meet the immediate needs for our song.

Matching

The matching process succeeded in finding melodies of the same length as lyric lines, but the stress patterns were not similar. Initial thoughts during the preparation of the song for the contest were that a larger collection of melodies was needed for matching. Indeed, a set of, say, 10,000 melodies might have provided greater variety and a higher probability of lyric and melody stresses matching more closely. The predominance of the symbol 4 in the melody stresses suggests, however, that greater value could have been gained from the existing generated melodies by either slowing them down or changing the threshold for unstressed notes to, say, a half beat, or by creating multiple versions at different thresholds. Alternatively, the stresses could be defined relative to other notes within the melody. For example, it might be a more meaningful stress pattern to say that an eighth note is a strong note compared to 16th notes. Similarly, in a melody made entirely of half notes and whole notes, the half notes would be a weak stress.

The lyric stresses themselves were an initial approximation to their normal stress pattern when spoken. Since the implementation considered only the word in isolation, based on the CMUdict pronunciation dictionary, there was no relationship between the words within the phrase. For example, as seen in Table 4, the pronunciation pattern for "dreams are cheap" does not capture that "are"

has a weaker stress than the words surrounding it. Although the word "are" could have received a blanket rule of weak stress, there are situations where it should have a strong stress.

Initial thoughts on modeling how lyrics are normally set to melodies led to the idea of compiling a corpus of sung music in symbolic form for training. This was abandoned due to lack of time, but the approach might have allowed for a more realistic matching of lyrics and melodies based on general practice, beyond stress representations.

Despite the output not quite being at a "human" level of accuracy, the quirks became features of the song. Syllabification of lyrics was only partially implemented during the competition, and early outputs that did not divide lyrics into syllables prior to the application to melodies became key features of the final song. Both lines of the chorus had their polysyllabic words allocated to a single note, with the first words of the phrase being repeated to fill the remaining notes of the melodic phrase, leading to the distinctive phrase "the world is beautiful, the world."

Design Directions to Support Cocreativity

Given our longer-term aim of developing workable AI assistants to support the composition and production process, the evaluation of the final output must be complemented with an evaluation of the creative process itself. This was conducted in-house through a reflective process, with coauthor Bown—who was not directly involved in the AI or creative production process—gathering and summarizing reflections from the other authors.

We have considered above whether the AI had a significant role in the creative outcome, and, although the role was complicated and highly constrained, it is possible to make this claim. From the perspective of cocreativity (Bown 2014; Yannakakis, Liapis, and Alexopoulos 2014; Kantosalo et al. 2015; Jordanous 2017) not only do we look at the division of creative input between humans and algorithms we find in the final output, but we also seek an understanding of how creative interaction takes place in the production process.

Was the Interaction "Dialogic"?

A *dialogic* interaction is one that is iterative and mutually influencing (Bown et al. 2020). For a creative AI system to be dialogic, there should be some evidence that the system influences the creative outcome, but that it is also itself adaptive during the creative process and positively reinforcing. Naturally, this concept of dialogic interaction stems from our experience of human collaborative creation, where a human participant is typically able to adeptly reconceptualize goals and representations, mediated through language. Bown et al. (2020) make the argument that dialogic interaction with AI systems need not resemble human dialogue, but might still achieve positive mutually reinforcing creative adaptation through more traditional interfaces such as GUIs or APIs.

In this project, we acknowledge that there is no form of adaptation on the software side that would allow this two-way dialogue. The system follows a standard operation-based interaction paradigm: It is controlled by simple commands and produces outputs.

Could it easily be adapted to introduce dialogic elements? The most obvious way would be to allow a user to give feedback to the outputs, such as in the form of additional text prompts, which would modify future generation. This is an area that is being intensively explored in new products such as OpenAI's DALL-E 3 (https://openai.com/dall-e-3). It should be reiterated that at the time of this work, in 2020, such methods were not yet well developed.

Was the Interaction Exploratory?

A more commonly found quality of creativity support tools is the ability to support blind, exploratory search effectively (Simonton 2011; Stanley and Lehman 2015). Qualities of good exploratory interfaces include offering a quick diversity of novel options and allowing iteration and refinement, narrowing in on a design. On this front, our approach conformed to a user experience for creative coding in which the output was scanned, and either elements were selected or the algorithms (or training set) were modified for further generation. We could

call this a "generate-audition-tweak" cycle. This supports exploratory search in basic ways, but it is evidently not geared towards making the best use of this process.

The process of matching melodic and lyric outputs is, however, a form of additional filtering that can be seen to support exploratory search. For example, the matching results in a match score, and the user has the choice of exploring the space of possible scores as well as experimenting with a more complex configuration of the system, combining generative model parameters and the parameters of the matching process. This is more typical exploration of an interface, as we experience with standard music tools such as synthesizers, and enables the user to combine control with the potential surprise and original contribution of the system. Such interaction can be thought of as "pseudodialogic."

What Was the Hit Rate?

Also interesting in this regard is the question of the hit rate of outputs: the distribution of outputs deemed good, inspiring, usable, typical, and so on. As has long been debated in computational creativity, the nature of a system's space of outputs describes the different kinds of creativity that it can exhibit, taking into account the distribution of typicality and novelty in the generated dataset (Ritchie 2007; Jordanous 2012).

The final melodic generation output was relatively conservative and low risk, as the input data had all been normalized in terms of key. Previously, melodic output had been based on a model trained on data of any key, and the output had had a higher proportion of musically "incorrect" results. Thus the final output set was a relatively flat space of "safe" melodic ideas that could easily be sifted through for favorites, which in turn could easily be concatenated, the result still being richly complex. By contrast, the space of lyric output was much more erratic. Predictably, much of the output was nonsensical and arbitrary, but with strands of delightful meaning and word association involved. These results were more salient (or "stood out"), by the nature of the lyrical medium.

Rapidly scanning and selecting both melodies and lyrics was easy and creatively stimulating. The additional step of stress-matching lyrics and melodies had a high hit-rate of lyric-melody combinations that "felt right." It was possible to manually select preferred melodies and then match, and also just to match on a larger number of unfiltered inputs. The latter was preferred for being quicker, but also because both lyrics and melodies were transformed in value through the matching process. For example, the song chorus, "the world is beautiful, the world" stood out to the producers when auditioned in its melody-matched context.

Was the Interaction Operational?

By operational, we mean, "could the system be adeptly operateed?"

The team worked largely with code APIs, with three experienced programmers working largely in Python, and did not produce or use custom user interfaces. The affordances available to control the system corresponded to a standard ML generation pipeline: varying the training set, varying the training data representation, varying the ML architecture, and varying the ML algorithm parameters. In addition, by using standard command-line tools, there were options to vary the representation of the outputs, including filtering and ordering results. Evidence of the team operating these parameters can be seen as a combination of one-off developer activities, leading to an improved tool, and ongoing iterative exploration (i.e., the tool already existed, and the development work was part of a creative process). For example, choosing to normalize the key was a decision that might then be locked in for further iterations of this work, or it could have been considered an on-the-fly decision to be reconsidered at any moment further on. With the GPT-2 algorithm, initially only Eurovision lyrics were used in the fine-tuning dataset, but the option of adding additional sources to diversify the output was explored. This would be considered an on-the-fly creative decision based on reflection on the output.

More specifically, the question of "operationality" refers to how expertly one can operate the system, with a reasonable expectation of what will

happen. For example, a producer adjusting a filter cutoff value on a low-pass filter knows exactly what might be expected in terms of the underlying DSP (though aesthetically there might still be some surprises). Adjusting training datasets and parameters, such as the "temperature" parameter in an ML algorithm, may correspond to clearly understood and articulated intentions ("make it more jazzy" or "give me more wild variety"), but the result may still not come out as expected. Given the rapid bricolage nature of the project (McLean and Wiggins 2010), we did not have an opportunity to more systematically study how coherent the system was to use, but we witnessed enough determinism and operational possibilities that we might expect a strong degree of coherence.

What Was the Experience? What Was Frustrating? What Was Stimulating?

Overall, the team found that the quality of ML generative systems now available has significantly lifted the quality of the experience, with higher hit rates and more interesting outputs—as well as far more accessible APIs provided by third parties, meaning that relative nonexperts can access this technology. The overall sentiment in the team was that if the barrier to accessing these generally high-quality generative outputs is low enough, then casual use of generative outputs in professional production has a reasonable benefit, especially in the way that it can stimulate new perspectives and ideas, continuing a long-standing tradition of using chance and opportunistic adaptation in creative practice, and well grounded in theories of creativity (such as those of Simonton 2011).

Remarks Regarding Context

It should be noted that the context for the project was 2020, and there have been many developments since then. Several online systems have been published that have made the creation of music with AI tools more accessible to nonexperts, such as MusicGen (Copet et al. 2023). They are fundamentally different to the process used and presented in this study, however.

Conclusions

In this article we sought to understand how the approach used in creating our entry in the AI song contest supports cocreativity. We used a practice-based research approach to the case study, in which we considered the context of the project and the team members, the techniques applied to the song's creation, and an analysis of the associated collaborative process.

We set out to use AI to generate key musical fragments and to use stress matching to discover interesting cohesive combinations of lyrics and melodies, an essential quality in a strong song. We feel that system provided plenty of good raw materials with which to work. Production processes were situated much more on the human side, but the AI output strongly influenced the final piece of music and forced us to be creative in different ways during production.

We put considerable effort into production, but this part took far less time than gathering data, working on code, and testing generation methods. Although some techniques were imperfectly implemented, the resultant song was highly successful, and exploited the peculiar outputs to create something that was both accessible to a wide audience and yet surprising. As a result it was highly rated by both judges and voters.

Future Directions

Our experience during the competition has suggested possibilities for future work. The concept of automatically matching melodic and lyric outputs based on stress patterns is a promising one, with much scope for improvement in its ability to work in a similar manner to preferred songwriting guidelines. Rather than enforcing a specific match, the ranked melodies for a lyric phrase can be explored, to align with the goal of supporting creativity rather than replacing it.

Since the closure of the 2020 AI Song Contest, OpenAI released the Jukebox system, which generates song audio (Dhariwal et al. 2020), and the GPT-3 language model (Brown et al. 2020). Both

these systems, which demand huge computing resources to train and to use, simulate human output to a remarkable degree. For example, the voices of famous singers, such as Frank Sinatra, are easily identified by lay listeners in the Jukebox output, and ChatGPT is able to produce coherent pieces in a particular style or genre without fine-tuning. Rather than using these and subsequent systems to create complete works, it will be interesting to explore how they could be used in human–AI cocreativity. A few current examples are MusicGen (Copet et al. 2023) and Riffusion (https://riffusion.com/about), which produce audio given a text description, one of many ways to allow humans to "rage with (rather than against) the machine."

Acknowledgments

We thank Sally-Ann Williams and Julia Zemiro for their input, Antigone Foster for her singing, Ash Watson for the music video, and the many people who spread the word and voted. We also thank Google's Creative Lab, who collaborated with us on the DDSP work in the project. Code for working with sinsy.jp was based on an student project at the Royal Melbourne Institute of Technology by Arthur Choung (code monkey), Edward Mumford (project manager), and Joseph Johnson (documentation and testing).

References

Ackerman, M., and D. Loker. 2017. "Algorithmic Songwriting with ALYSIA." In *Computational Intelligence in Music, Sound, Art and Design: 6th International Conference*, pp. 1–16.

AKF. 2021. "Koala Population Estimates: National Overview 2021." Report. Brisbane, Queensland, Australia, Australian Koala Foundation. Available online at www.savethekoala.com/wp-content/uploads/2021/09/KoalaEstimates2021.pdf. Accessed November 2023.

Ames, C. 1987. "Automated Composition in Retrospect: 1956–1986." *Leonardo* 20(2):169–185. doi:10.2307/1578334

Ariza, C. 2011. "Two Pioneering Projects from the Early History of Computer-Aided Algorithmic Composition." *Computer Music Journal* 35(3):40–56. doi:10.1162/COMJ_a_00068

Bolcato, P. 2020. "Concurrent Generation of Melody and Lyrics by Recurrent Neural Networks." Master's thesis, KTH Royal Institute of Technology, Department of Computer Science and Engineering, Stockholm.

Bown, O. 2014. "Empirically Grounding the Evaluation of Creative Systems: Incorporating Interaction Design." In *Proceedings of the International Conference on Computational Creativity*, pp. 112–119.

Bown, O., et al. 2020. "A Speculative Exploration of the Role of Dialogue in Human–Computer Co-Creation." In *Proceedings of the International Conference on Computational Creativity*, pp. 25–32.

Brown, T., et al. 2020. "Language Models Are Few-Shot Learners." *Advances in Neural Information Processing Systems* 33:1877–1901.

Burgoyne, J. A., and H. V. Koops. 2021. "We Are Not Groupies … We Are Band Aids: Assessment Reliability in the AI Song Contest." *Transactions of the International Society for Music Information Retrieval* 4(1):236–248. doi:10.5334/tismir.102

Candy, L. 2019. *The Creative Reflective Practitioner: Research through Making and Practice*. Milton Park, UK: Routledge.

Carr, C., and Z. Zukowski. 2018. "Generating Albums with SampleRNN to Imitate Metal, Rock, and Punk Bands." In *Proceedings of the International Workshop on Musical Metacreation*. Available online at musicalmetacreation.org/mume2018/proceedings/Carr.pdf. Accessed November 2023.

Collins, N. 2016. "A Funny Thing Happened on the Way to the Formula: Algorithmic Composition for Musical Theater." *Computer Music Journal* 40(3):41–57. doi:10.1162/COMJ_a_00373

Copet, J., et al. 2023. "Simple and Controllable Music Generation." In *Proceedings of the Conference on Neural Information Processing Systems*. Available online at nips.cc/virtual/2023/poster/70671. Accessed November 2023.

Davis, N. 2013. "Human–Computer Co-Creativity: Blending Human and Computational Creativity." In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pp. 9–12.

Dhariwal, P., et al. 2020. "Jukebox: A Generative Model for Music." arXiv preprint arXiv:2005.00341. Accessed November 2023.

Doornbusch, P. 2004. "Computer Sound Synthesis in 1951: The Music of CSIRAC." *Computer Music Journal* 28(1):10–25. doi:10.1162/014892604322970616

Edwards, M. 2011. "Algorithmic Composition: Computational Thinking in Music." *Communications of the ACM* 54(7):58–67. doi:10.1145/1965724.1965742
Engel, J., et al. 2020. "DDSP: Differentiable Digital Signal Processing." In *Proceedings of the International Conference on Learning Representations*. Available online at openreview.net/forum?id=B1x1ma4tDr. Accessed February 2024.
Fernández, J. D., and F. Vico. 2013. "AI Methods in Algorithmic Composition: A Comprehensive Survey." *Journal of Artificial Intelligence Research* 48:513–582.
Ghedini, F., F. Pachet, and P. Roy. 2016. "Creating Music and Texts with Flow Machines." In G. E. Corazza and S. Agnoli, eds. *Multidisciplinary Contributions to the Science of Creative Thinking*. Berlin: Springer, pp. 325–343.
Hadas, E. 2021. "How Computational Literature Interacts with Text Analysis." In E. Navas, O. Gallagher, and x. burrough, eds. *The Routledge Handbook of Remix Studies and Digital Humanities*, chapter 19. New York: Routledge, pp. 289–301.
Hennequin, R., et al. 2020. "Spleeter: A Fast and Efficient Music Source Separation Tool with Pre-Trained Models." *Journal of Open Source Software* 5(50):Art. 2154. doi:10.21105/joss.02154
Herndon, H. R. 2019. "Proto." PhD dissertation, Stanford University, Department of Music.
Hiller, L. A., and L. M. Isaacson. 1958. "Musical Composition with a High-Speed Digital Computer." *Journal of the Audio Engineering Society* 6(3):154–160. Originally presented at the Annual Convention of the Audio Engineering Society, New York City, 9 October 1957.
Huang, C.-Z. A., et al. 2020. "AI Song Contest: Human–AI Co-Creation in Songwriting." In *Proceedings of the International Conference on Music Information Retrieval*, pp. 708–716. doi:10.5281/zenodo.4245530.
Ji, S., X. Yang, and J. Luo. 2023. "A Survey on Deep Learning for Symbolic Music Generation: Representations, Algorithms, Evaluations, and Challenges." *ACM Computing Surveys* 56(1):Art. 7.
Jordanous, A. 2012. "A Standardised Procedure for Evaluating Creative Systems: Computational Creativity Evaluation Based on What It Is to Be Creative." *Cognitive Computation* 4(3):246–279. doi:10.1007/s12559-012-9156-1
Jordanous, A. 2017. "Co-Creativity and Perceptions of Computational Agents in Co-Creativity." In *Proceedings of the International Conference on Computational Creativity*, pp. 159–166.
Kantosalo, A. A., et al. 2015. "Interaction Evaluation for Human–Computer Co-Creativity." In *Proceedings of the International Conference on Computational Creativity*, pp. 276–283.
Kim, J. W., et al. 2018. "CREPE: A Convolutional Representation for Pitch Estimation." In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 161–165.
Kingma, D. P., and J. Ba. 2015. "Adam: A Method for Stochastic Optimization." In *Proceedings of the International Conference on Learning Representations*. Available online at arxiv.org/abs/1412.6980. Accessed April 2024.
Klein, M. 1957. "Syncopation in Automation." *Radio-Electronics* 28(6):36–38.
London, J. 2011. "Metre." In A. Latham, ed. *Oxford Companion to Music*. Oxford, UK: Oxford University Press. doi:10.1093/gmo/9781561592630.article.18519.
McLean, A., and G. A. Wiggins. 2010. "Bricolage Programming in the Creative Arts." In *Psychology of Programming Interest Group Annual Workshop*. Available online at www.ppig.org/files/2010-PPIG-22nd-McLean.pdf. Accessed February 2024.
Mehri, S., et al. 2017. "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model." In *International Conference on Learning Representations*. Available online at openreview.net/pdf?id=SkxKPDv5xl. Accessed March 2024.
Micchi, G., et al. 2021. "I Keep Counting: An Experiment in Human/AI Co-Creative Songwriting." *Transactions of the International Society for Music Information Retrieval* 4(1): 263–275. doi:10.5334/tismir.93
Oura, K., et al. 2010. "Recent Development of the HMM-Based Singing Voice Synthesis System: Sinsy." In *Proceedings of the ISCA Workshop on Speech Synthesis*, pp. 211–216.
Paszke, A., et al. 2019. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pp. 8024–8035.
Pearce, M., D. Meredith, and G. Wiggins. 2002. "Motivations and Methodologies for Automation of the Compositional Process." *Musicae Scientiae* 6(2):119–147. doi:10.1177/102986490200600203
Radford, A., et al. 2019. "Language Models Are Unsupervised Multitask Learners." Blogpost 20 February 2023. Available online from academic.jyunko.cn/publications. Accessed November 2023.
Ritchie, G. 2007. "Some Empirical Criteria for Attributing Creativity to a Computer Program." *Minds and Machines* 17(1):67–99. doi:10.1007/s11023-007-9066-2
Simonton, D. K. 2011. "Creativity and Discovery as Blind Variation: Campbell's (1960) BVSR Model after the

Half-Century Mark." *Review of General Psychology* 15(2):158. doi:10.1037/a0022912
Smith, A. G., C. X. S. Zee, and A. L. Uitdenbogerd. 2012. "In Your Eyes: Identifying Clichés in Song Lyrics." In *Proceedings of the Australasian Language Technology Association Workshop*, pp. 88–96.
Stanley, K. O., and J. Lehman. 2015. *Why Greatness Cannot Be Planned: The Myth of the Objective*. Berlin: Springer.
Tierney, A. T., F. A. Russo, and A. D. Patel. 2011. "The Motor Origins of Human and Avian Song Structure." *Proceedings of the National Academy of Sciences* 108(37):15510–15515. doi:10.1073/pnas.1103882108
Uitdenbogerd, A. L. 2019. "World Cloud: A Prototype Data Choralification of Text Documents." *Journal of New Music Research* 48(3):253–263. doi:10.1080/09298215.2019.1606255
Van Eeden, L. M., et al. 2020. "Impacts of the Unprecedented 2019–2020 Bushfires on Australian Animals." Technical report. Ultimo, New South Wales, Australia, World Wildlife Fund, Australia. Available online at assets.wwf.org.au/image/upload/v1/website-media/resources/WWF_Impacts-of-the-unprecedented-2019-2020-bushfires-on-Australian-animals. Accessed November 2023.
Wolf, T., et al. 2020. "Transformers: State-of-the-Art Natural Language Processing." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45.
Yannakakis, G. N., A. Liapis, and C. Alexopoulos. 2014. "Mixed-Initiative Co-Creativity." In *Foundations of Digital Games: Proceedings*. Paper 37. Available online at www.fdg2014.org/papers/fdg2014_paper_37.pdf. Accessed February 2024.
Zukowski, Z., and C. Carr. 2018. "Generating Black Metal and Math Rock: Beyond Bach, Beethoven, and Beatles." doi:10.48550/arXiv.1811.06639.

Appendix

The following sound examples are available at http://direct.mitpress.edu/comj/10.1162/COMJ_a_00674.

The song itself is followed by Examples 2–5, which were part of the audio data set used as training data for the DDSP Koalasynth. These raw samples were not directly used in the song. Examples 6–10 are examples of the Sinsy output for automatically paired lyric lines and melody snippets (with initial and final silence manually removed). Example 11 shows an earlier idea that was explored using melody–lyric pair output, rendered using Sinsy. This was later abandoned in favor of the "Welcome Home" song idea. Example 12 is a contrasting interpretation of the song, inspired by medieval musical techniques instead of pop.

Example 1 The song "Beautiful the World," as submitted to the 2020 AI Song Contest.
Example 2 Excerpt of the song, highlighting the lead break that uses the Koalasynth.
Example 3 Excerpt from the koala calls data set.
Example 4 Excerpt from the kookaburra calls data set.
Example 5 Excerpt from the tasmanian devil calls data set.
Example 6 Original lyric–melody matched Sinsy rendition of verse 1 line 1 ("Flying in fear but love keeps on coming").
Example 7 Original lyric–melody matched Sinsy rendition of verse 1 line 2 ("Dreams still live on the wings of happiness").
Example 8 Original lyric–melody matched Sinsy rendition of verse 1 line 4 ("Open up the heart I ain't scared no more").
Example 9 Original lyric–melody matched Sinsy rendition of chorus line 1 ("Welcome home oh welcome home oh oh oh").
Example 10 Original lyric–melody matched Sinsy rendition of chorus line 2 ("The world is beautiful").
Example 11 An earlier idea developed from other melody–lyric pair outputs.
Example 12 A different interpretation of the song, with all vocals rendered by Sinsy, and with minimal effects on the Koalasynth lead break.