Towards an Automated Composer Of Popular Country Music

Jim Suruda, Norman Carver
Department of Computer Science, Southern Illinois University, Carbondale IL, USA

Abstract— *In the context of a software tool that can generate novel melodies in the popular country genre, this article describes the development of such a system. Algorithmic composition systems typically generate musical works and leave the evaluation of the output as a task for a human listener. This article describes an attempt to use machine learning techniques to evaluate the quality of the generated melodies. We describe the architecture of a prototype melody generation and evaluation system, which uses extracted features from MusicXML to judge the quality of machine generated melodies. Experimental results, including rankings of generated songs compared to currently charting country songs, are reported.*

Keywords: machine learning, composition, music, artificial neural networks, genetic algorithms

1. Introduction

The first computer program ever written might be one that calculates Bernoulli numbers, written by Ada Lovelace for Charles Babbage's computing engine. In her notes on the computing engine, Lovelace speculated about what further problems a more powerful computing engine might solve:

> "[computers] ... might compose elaborate and scientific pieces of music of any degree of complexity or extent" [4]

So, interest in computer music composition music begins with the first computer science research paper ever published in English. Researchers continue to explore new algorithmic composition techniques every year, from jazz solo construction to real-time accompaniment systems to AI applications that generate concertos. But researchers rarely conduct machine evaluation of generated music. A recent and exhaustive book on the subject of algorithmic composition makes only a brief one-paragraph mention of the evaluation of generated material, and cites no research in evaluation at all. [10] Despite the lack of mention by Nierhaus, there has been some work on the evaluation of generated work, [8] [13], and many more avenues of exploration still beckon.

Music generation and evaluation is a naturally appealing subject for research because even a prototype system can produce novel and interesting compositions, giving an exciting glimpse at the future of AI, when intelligent systems might begin to emulate and expand beyond human's creative abilities. More practically, efforts that uncover generation rules or aid in feature extraction from the highly structured but well-defined data of music could have applications in related fields. The areas of medical data, imaging, and natural language all have dense and complex data sets in which important features are not always obvious.

We chose music generation as the domain for our testbed system because a working prototype of a generation and evaluation system makes an excellent springboard for further research in machine learning. Judgements about the quality or catchiness of music are naturally imprecise and fuzzy, and lend themselves well to soft computing and AI techniques. Working within an existing genre simplifies compositional complexity, and our geographic and cultural proximity to Nashville Tennessee made popular country music a compelling choice for genre.

This paper describes the initial implementation of ACME (The Automated Country Music Engine), a testbed we have developed to explore computer generation and evaluation of popular country music. The initial implementation has demonstrated the feasibility of using a simple stochastic context-free grammar to generate a variety of novel melodies within a genre, and the possibility of using extracted features rank the fitness of the generated works.

2. Related Work

Researchers had been developing algorithms for generating melodies and producing harmony accompaniment long before computers existed. One of the earliest examples of melody generation is the Musikalisches Wurfelspiel, published in book form in 1792 and attributed to Mozart. The book contained small sections of music that the user would select by rolling dice. Users would play the concatenated sections as a single, novel work of music. Algorithms for harmony accompaniment date from 1725, when Johann Fux published a system of rules for creating counterpoint accompaniment to existing melodies. [5]

Within years of the development of the first electronic computers, researchers were writing programs to generate music. One of the first to do this was Henry F. Olson, who in 1950 extracted first and second order Markov models of note transitions from the works of Stephen Foster and used them to generate novel melodies in the style of Foster. [11] Also in the 1950s, Noam Chomsky's work on generative grammars for natural language inspired music researchers to develop generative grammars for music. Lerdahl and Jackendoff built on Chomsky's work in the 1980s to develop an ambitious grammar designed to generate any type of tonal music [9] Since then, researchers have applied techniques from nearly

every area of AI and machine learning to the creation of music works: cellular automata, transition networks, genetic algorithms, and rule-based systems. But few have attempted to quantify the quality of the generated works.

Klinger and Rudolf [2006] used an artificial neural network to rank the fitness of computer generated melodies. [8] They extracted a small set of features from melodies and used those along with user-supplied evaluations to train a feed forward neural network to rate the melodies. They had difficulty creating a large enough body of training data because they relied on users to listen to and rank each training example, which was time consuming.

3. Problem Definition

When country music songwriters write songs, they often create and reject a number of melodies in the process of writing one song. Writers repeatedly create and reject series of notes until they find something "catchy" or interesting to incorporate into a song. [3] Of all the songs they write each month, writers only submit a few of the best to their publishers. [7] Publishers select a few of the most promising songs to pitch to artists, who may choose some of the songs to record for an album. Of all the thousands of albums released each year, only a few songs get airplay on radio stations. So for each song that makes it to the radio, a huge number of songs are created, and only a few of the "fittest" songs are selected from the pool for recording and popular rotation. This is the process we modelled for our prototype music generation system: create a large population of songs, then use an evaluation system to select the best compositions. It is those songs that are undeniably excellent that are in demand, not those that are reasonably good. Tom T. Hall once said, on the subject of "pretty good" songs "... if we could make a career out of writing pretty good songs there'd be a lot more people in the business than there are." [7]

When a publisher chooses a song that they would like to present to a performer, they create what is called a "demo" of the song. The demo is a sample recoding of the song usually performed by studio musicians and a vocalist, created to showcase the song to a performer. The studio musicians play from a lead sheet, which contains only the melody notes, lyrics, and chord symbols. [3] During the demo session the studio musicians and producer create the song arrangement; the harmony, bass parts, backup vocals, and other instrumentation. Thus, the essential deliverable produced by the songwriter is the lead sheet: melody, lyrics, and chord symbols. Since our system does not attempt to create lyrics, the output will be melody notes and chord symbols. Figure 1 shows an example of a lead sheet for a popular country song.

MusicXML is a standard format for lead sheets and is supported by a variety of scorewriting and music editing tools, so we chose to have the project produce and evaluate songs in MusicXML format. The system should be able to generate a large number of unique and original songs and evaluate them

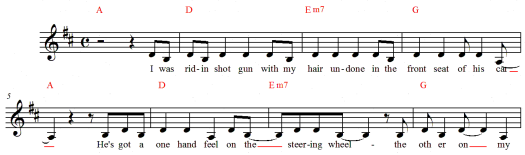

Fig. 1: Section of a Lead Sheet.

based on a heuristic that has some relation to the evaluation that occurs when human-composed songs compete for radio airtime. If our project is successful, the top-rated songs the system generates should be of equal quality to those composed by a human expert. This success is dependant on the ability of the generation system to produce successful songs and on the ability of the evaluative system to successfully recognize the best compositions. The acme of success for our prototype would be for an artist to record one of the computer-generated songs, and for the song to garner radio airtime.

4. Project Goals

We implemented our system, called the Automated Country Music Engine (ACME), to provide the following capabilities:

1) Generate melodies
2) Add chord accompaniment
3) Write MusicXML
4) Read existing MusicXML
5) Extract features from MusicXML
6) Train a neural network to rank songs
7) Evaluate generated songs

Our goal for this iteration of ACME was to create a complete, working system. Breaking the project up into subcomponents based on the capabilities listed above made it possible to attack the problem in independent sections. We chose to implement ACME in C#, because the ergonomic IDE facilitates rapid prototyping, because of its strong XML and string manipulation capabilities, and partly due to philosophical preference for a statically-typed language.

5. Implementation

Figure 2 illustrates the high-level architecture of ACME. In the following sections, we will briefly describe the implementation of each major capability.

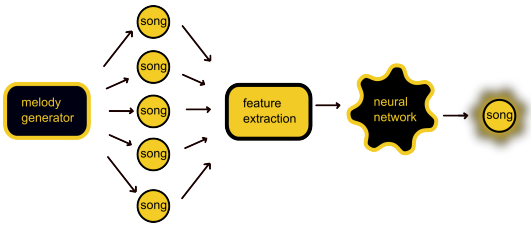

Fig. 2: ACME System Overview.

5.1 Generating Melodies

If our composition system is capable of successfully evaluating melodies, the melody generator does not need to be especially good on average, as long as it can generate a large variety of unique melodies within the popular country genre. The space of all possible musical compositions is obviously enormous, but the characteristics of typical country music compositions are relatively tightly constrained, so the subspace of reasonable country songs is much smaller. It is worth considering how large this subspace is to get a sense of the difficulty of the problem of generating popular country music.

A representative country song might consist of only three main sections (verse, chorus, and bridge) that are repeated. Each section could be about 8 bars of music, with 8 notes or rests possible per measure. Using a scale of eight notes plus a rest, there are 9 possible tones available. So, the search space of all possible country songs is roughly 9^{3*8*8} unique melodies. If a system could evaluate a million of the songs in the search space per second it would take $1.7 * 10^{55}$ years to process all possible songs. Enumeration of all possible works is not a practical generation strategy. Other, less constrained genres would have even larger search spaces, so popular country is still an attractive choice for an experimental domain.

For our initial implementation, we settled on a generative grammar to produce melodies since this made it easy to represent the types of rules found in songwriting books. Instead of enumeration, we settled on a simple generative grammar to produce melodies. Generative grammars were introduced by Noam Chomsky in the 1950s [2] as a way to describe the hierarchical structure of the human language. They can also be used to model the meta-structure of music, and to serve as a production guide for generating melodies. This hierarchical deconstruction of music was actually first proposed by Heinrich Schenker in the 1930s and was a basis for the hierarchical system for the construction of music later proposed by Lerdahl and Jackendoff. [9]

Although an implementation of Lerdahl and Jackendoff's grammar might produce a wide variety of novel melodies, they would certainly not be limited to popular country melodies. At the opposite end of the spectrum, popular country songwriting books often provide no algorithmic or generative rules for producing melodies. For instance, despite its title, Tom T. Hall's book "How I Write Songs", contains no description of methods for creating melodies. [7] Rather than devise our production rules *a priori*, we based ACME's generative grammar on those described by Stephen Citron in his book "Songwriting." [3] Although written for the human songwriter, Citron's melody generation rules sketch the outlines a stochastic context-free grammar that forms the core of ACME's generation system.

The terminal symbols in our musical context-free grammar are notes and rests. ACME limits itself to notes from a single major or minor scale for each composition. Although the conventions of popular country music allow accidentals and modulation, for simplicity ACME does not currently use notes outside the scale, and does not change the tonic of the within the melody. Assembling a series of notes from the lead scale into what we call a melodic cell is Citron's first production rule for melody construction.

To select a sequence of tones for the melodic cell, we use the generating techniques for white, brown and 1/f melody described by Gardner.[6] ACME composes brown melodies by choosing notes that are at most two steps offset from the previous note, and white melodies by choosing notes randomly, without regard to preceding tones. We implement Gardner's algorithm for 1/f melodies, which produces tone sequences that fall somewhere between white and brown. At this point ACME also assigns durations to the notes, picking lengths randomly from eighth to whole note.

Because the first note in a melodic cell can be important in determining the successive notes, we had ACME search a corpus of popular country music to determine the likelihood of a phrase starting with each scale tone. Phrases tended to begin on the tonic or dominant, and ACME uses its analysis of starting note frequencies to stochastically pick the beginning tones for melodic cells.

After assembling a melodic cell of 1-4 notes as described above, ACME follows the rules described in [3] to combine the melodic cells into motives:

- Repetition: Repeat the melodic cell exactly.
- Translation: Repeat the melodic cell at higher or lower intervals.
- Division: Repeat a portion of the melodic cell.
- Modification: Repeat a portion of the cell with modified intervals.
- Introduction: Begin a novel melodic cell.

ACME chooses a series these operations stochastically: for this prototype the each rule has equal likelihood of being applied when extending a melodic cell. Adjustment of these probabilities could be one way to improve generation in future prototypes, based on feedback from the melody evaluation.

The series of modified melodic cells forms a motive. The same production rules are then applied to motives to form a phrase. This simple method of modifying and repeating the same or novel sections seems to follow the conventions of popular country music. For example, the phrase shown in Figure 3 from a popular country song written by Taylor Swift suggests a similar construction pattern. Swift's motive repeats a two-note melodic cell twice, then repeats a portion of the cell twice. This motive, M, is stated, then repeated twice again slightly modified form, M'. The phrase finishes by introducing a new motive, R, which resolves to the tonic.

After creating a phrase, ACME repeats and modifies the phrase using the same production rules outlined in of [3], to generate a period. ACME concatenates these periods together to function as the verse, bridge or chorus of a song.

ACME's system of melody generation is basic, and there are many established songwriting rubrics that might improve ACME's compositions. From Fux's rule that after a long jump

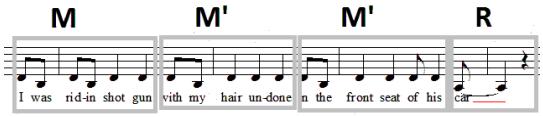

Fig. 3: Melodic Motives.

in one direction melodies ought to take a small step in the other direction [5] to Paul Simon's advice to construct the bridge from notes used infrequently in the verse and chorus [14], there are many heuristics we could incorporate into ACME's generation system. These rubrics suggest that context is important in producing quality melodies, and so a weakness of the context-free grammar is that it is unable to capture interactions between song sections that determine in some way the quality of the melody.

A context-sensitive grammar might produce better average quality, especially when combined with other machine leaning techniques, such as weight search. But it seems unlikely that weights could be found that would generate only good songs. Based on the model of human songwriters, it seems that even a fine-tuned generator will still produce many low-quality songs. For the purposes of providing input to our evaluative system, ACME does the job of creating a large volume of melodies for lead sheets. If the universal search space is the set of all possible melodies, our generation system produces a much smaller set of compositions, because the production is limited to tonal music in a set melody scale, with a fixed length and some repetition of phrases. Further fine-tuning of the generation rules might reduce the search space, at the risk of excluding some portion of the set of country melodies.

5.2 Chord Generation

ACME is able to generate melodies, which are one component of a lead sheet. Lead sheets also have chord symbols for rhythm guitar accompaniment that are used by the arranger to guide creation of harmony parts. A chord is a set of notes that sound at the same time, often strummed on guitar or played on a keyboard. Each chord contains notes from a scale that may be different from the melody scale. Generally a chord is defined by three tones, the chord triad.

Chord changes usually occur at the beginning of measures [12], and tend to draw from scales close to the melody scale on the circle of fifths. [3] We used these two assumptions to simplify ACME's chord generation task: we only allow chord changes on the first beat of a measure, and use only major and minor keys adjacent to the melody scale on the circle of fifths. So for a song in the key of C major, our palette of available chords is C, F, and G major, along with their relative minors A, D and E minor.

Although ACME is able to narrow down the set of possible chords for a phrase and their potential positions, arranging the chord changes in order is a more difficult problem. Texts on harmonizing tend towards subjective evaluation, although there are some common themes. The relation of chords of the sequence of chords is important: Citron suggests beginning with the tonic and ending with the dominant. [3] Piston stresses that chords should change from measure to measure, but not always, and that the distance between keys should be a factor in choosing chords. [12]

The relation between notes in the measure and the associated chord is also important. Chord tones that are dissonant with the notes of the melody are discouraged, and melody notes that are part of the chord triad are encouraged. [12] Agreement of chord tones with longer melody notes and notes at the start of the measure also are considered significant. [1]

Although we lack an algorithm for harmonizing an existing melody, we can describe features that might play a part in determining the suitability of a particular chord accompaniment. The features describe both the relation of notes to their accompanying chord change, and the relation of chord changes to each other:

- Triad Ratio: The ratio of notes governed by the chord change that are part of the chord triad.
- Scale Ratio: Ratio of notes played during the chord that are tones in the chord scale.
- Nonharmonic Ratio: Fraction of notes that are passing tones in the accompanying chord scale.
- Accidental Tones: Ratio of notes that are not members of the chord scale.
- Key Centeredness: 1 if the phrase starts with the tonic chord, 0 otherwise.
- Resolution: 1 if the phrase ends on the dominant chord, 0 otherwise.
- Freshness: Ratio of measures that contain change in chord.
- Mode Freshness: Ratio of chord changes that are different in mode from the previous chord.
- Harmonic Movement: Average distance in fifths between successive chords.

Given a set of candidate chord changes for a melody, ACME can calculate these features for each measure. If we could determine a set of weights to assign to the features, ACME could use that heuristic to rank its own chord changes.

We used a genetic algorithm (GA) to find weights for our chord sequencing heuristic. As training data we used a set of MusicXML lead sheets of contemporary country songs obtained from Wikifonia. ACME created a population of individuals initialized with randomized starting weights, and calculated the fitness of each individual as the linear combination of weights and features each generation. We considered the chord accompaniment in the training set to have an optimal chord fitness of 100, and defined the fitness of individuals as the squared difference of their ranking of the chord changes from 100. Using a population of 1000 individuals for 1000 generations the GA kept the fittest 2/3 each generation, and randomly picked 10% of the remaining individuals for crossover and re-stocked the rest of the popu-

lation with mutation.

After 400 generations the GA produced individuals that had a minimal error against the training data. Figure 4 is a chart of the average and best fitness over 1000 generations.

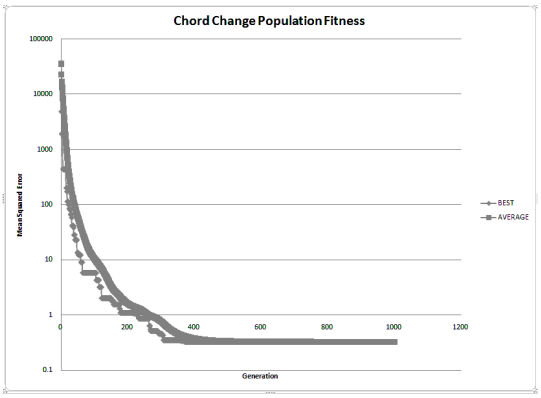

Fig. 4: Chord Weight Population Fitness.

Using the weights of the most-fit individual from the GA as a heuristic, ACME is able to score the fitness of a chord accompaniment to a melody. Given a melodic phrase, ACME generates a set of permutations of chords for the melody. Then, ACME ranks the chord accompaniments and chooses the chord progression with the highest score. Using this method ACME generates chord accompaniments that sound pleasing, while the lowest scoring accompaniments are jarring in their dissonance. As part of the next iteration of ACME we plan to play a selection of low and high scoring chord accompaniments to human listeners and see if their judgements agree with the system's heuristic.

5.3 Reading and Writing MusicXML

MusicXML is an XML-based music file format, introduced in 2004. MusicXML was an attractive format compared to MIDI or PDF because MusicXML is widely supported, can be stored as human-readable text, and appeared to have a reasonable set of examples available online. Free players to edit and export MIDI from MusicXML are also available. The main disadvantage of MusicXML for ACME is that contemporary country songs are generally not available for free. Creating MusicXML samples involved typing in songs by hand from sheet music.

Coding a parser for MusicXML was not difficult, as it is an XML format and C# contains robust classes for navigating XML. We created packages to read and write MusicXML and to allow ACME to save its generated lead sheets as MusicXML files.

5.4 Basic Feature Extraction From MusicXML

Humans, even without formal training, have a strong innate sensibility for music, and can make value judgements on the quality and structure of music. Most people can recognize a variety of musical features: repeated motives, song structure, dissonance, and the ineffable "catchiness" of well-crafted songs.

ACME is able to extract a set of features from a melody stored as MusicXML, and uses those features to emulate a human's ability to make judgements about music. The feature set uses 11 features described in [13] with additional features based on [3], for a total of 80 features. The 11 features that follow from [13] include melodic features, tonality features, melodic contour features, and rhythmic features:

- Pitch Variety: Ratio of distinct pitches to notes.
- Pitch Range: The difference in half-steps between the highest and lowest tones in the melody.
- Key Centeredness: Ratio of dominant or tonic notes (important tones for asserting the melody's key).
- Dissonant Intervals: Ratio of notes whose preceding note is seven semitones distant (intervals of a seventh are dissonant).
- Contour Direction: Overall trend of the melody to rise or fall.
- Contour Stability: Tendency of the melody to continue moving in the same direction.
- Brownian Steps: Ratio of intervals that are a single scale step.
- Leap Restraint: Ratio of large steps that are followed be a small step.
- Note Density: Average number of notes per beat.
- Rest Density: Proportion of beats that are silent.
- Repeated Duration: Proportion of notes that have the same duration as the previous note.

5.4.1 Feature Extraction From Motives

The features described in the previous section all provide aggregate information about the melody as a whole. But, as illustrated in Figure 3, there can be motives in country music that repeat and develop the theme of the melody. The characteristic riff or hook of a song can be recognized by a human listener, and ACME is also able to pick out features from the prominent motive of the melody. ACME finds the most commonly repeated series of notes that recur in the melody: the main motive. By extracting the 11 features described earlier separately for the notes of the motive, ACME has information about the range, direction, variety, and other features of the main motive. In addition, ACME also creates features for the length and number of repetitions of the motive.

Songs can have different motives in different song sections, and motives may not be repeated verbatim. The ability to extract multiple motives and be flexible in recognizing variations would certainly produce a richer feature set. But at present the system has at least some insight into some features of the most reiterated motive.

5.4.2 Feature Extraction From Note Frequency

The base feature set contains one feature, *Key Centeredness*, which describes the combined proportion of the I and V tones in the melody. ACME extends this feature to provide individual proportions for all 7 tones in the melody scale. These 7 scaled values are added to the existing feature set. Having calculated these values, the system is able to create a visualization of the usage of scale tones in the melody. By laying out a histogram of the frequency of note usage, ACME can create human-readable data about the melody during its feature extraction. Figure 5 is a tone frequency histogram generated by ACME during feature extraction.

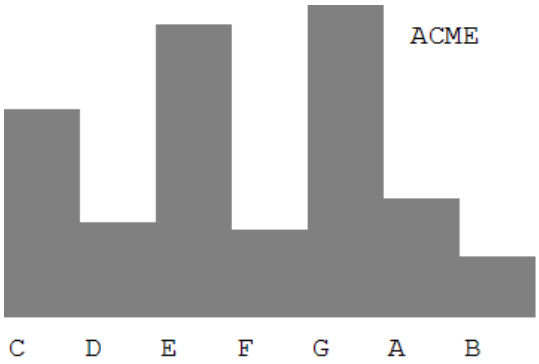

Fig. 5: Tone Frequency Histogram.

5.4.3 Feature Extraction From Note Transition Model

The likelihood of moving from any single scale tone to another can be represented by a 1st-order Markov model. Some of the earliest attempts at algorithmic composition relied on solely Markov models to generate melodies in the style of a corpus of music [11]. To make use of these features, ACME creates a 1st-order Markov model of the scale tone transitions in a melody. The generation system composes using major and minor scales of seven tones, so a 1st-order Markov model of the note transitions has 49 values, which ACME adds to the existing feature set. When these features are plotted graphically, they provide a visual representation of likelihood of transitions from scale tones to other scale tones in the melody. Figure 6 is an example image plotted by the system of the Markov transition features of a generated song. Darker squares represent more common transitions, while lighter squares represent less frequent transitions. The most common transition in the melody represented in Figure 6 is the transition from F# to E.

5.5 Training The Evaluation Neural Network

ACME is capable of extracting features from training examples to use as input for a neural network. For training data, we used a set of several dozen country music songs in MusicXML format. Since the goal of the system is to create songs that might be successful on the radio, ACME uses the

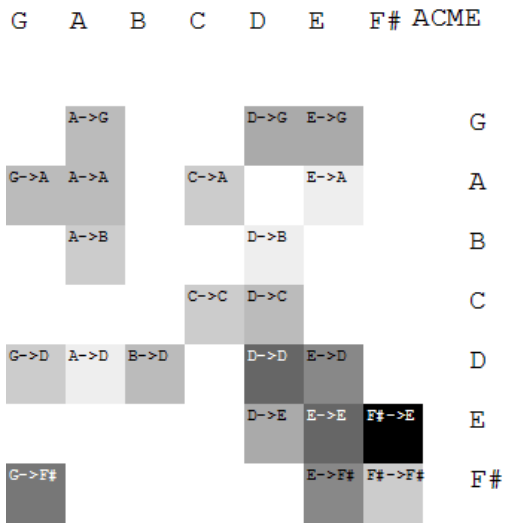

Fig. 6: Note Transition Visualization.

song's Billboard chart ranking (1 being the best and 100 the worst ranking) as the expected value of the examples. We had some difficulty in obtaining contemporary country music lead sheets. There is no shortage of lead sheets, but they are not generally available for free. We were able to find some lead sheets on Wikifonia.org, and entered more manually using a score editor.

To provide negative examples, the system created duplicate, degraded versions of the positive training examples. ACME randomly nudged intervals, reordered notes in measures, split and combined notes, and inserted and deleted rests to existing melodies. These modified examples sound undeniably execrable, and form our set of negative examples. The system assigns a very poor expected value of 1000 to these negative examples.

Given a set of labelled training examples and the ability to extract a set of features, ACME is capable of training a backpropagation neural network. After some experimentation, we obtained satisfactory results from a fully-connected network with 80 neurons in the input layer (one for each of the the 80 features), two hidden layers of 40 neurons each, and an output layer of one neuron. Figure 7 shows the reduction in mean squared error over 300 iterations over the training examples.

5.6 Evaluation of Generated Songs

ACME can generate a few thousand songs per hour, and evaluation with the neural network can keep pace with generation. So ideally, we could sort through 100,000 original songs in a few days and select the best-ranked melody. Given our simple generation algorithm, we would expect that the majority of generated lead sheets should rank poorly. As expected, the neural network does classify the majority of the output as poor. In one run ACME generated and evaluated 5000

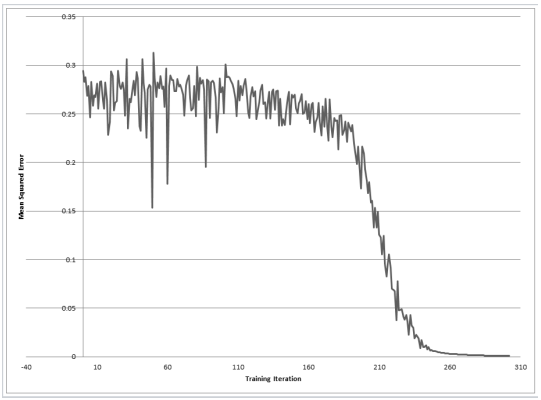

Fig. 7: Mean Squared Error Over 300 Iterations.

lead sheets, with the classification representing the putative chart position of the generated song: 0 being the best possible ranking and 1 the poorest. Figure 8 shows the ranking for the 5000 songs, grouped by ranking. Most of the generated melodies fall into the poorest category, while only a small number garner a high rating.

To our ears the poorly rated melodies do not sound pleasing, and although the highly ranked melodies are noticeably better, the best ranked melodies are not of exceptional quality. The high ranked songs are not of equal quality to those produced by a human expert, so the evaluation system could use improvement in discrimination. More features might be needed, and it is also not clear that a neural network is the best choice for evaluation of generated melodies. An alternate technique, such as Bayesian learning, might produce better results and could help illuminate which features or combination of features are most critical for judging the popularity of country music.

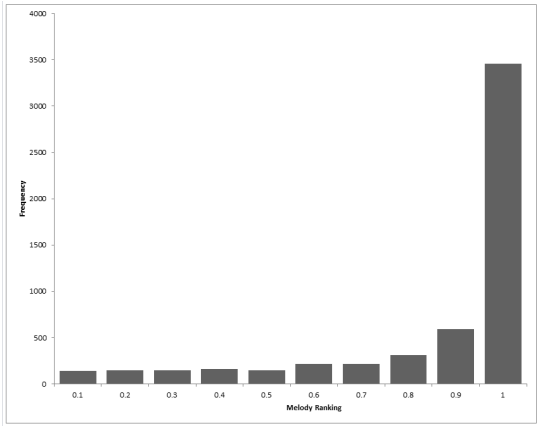

Fig. 8: Generated Song Rankings.

6. Conclusions and Future Work

ACME's results demonstrate that it is possible to generate a large volume of novel melodies and use existing machine learning techniques to evaluate and select those that are "better". From this proof of concept we plan to consider a variety of techniques to improve our results. We will consider adjusting the rules and learning weights for the stochastic context-free grammar to increase the proportion of higher-quality melodies. We plan to investigate alternative generation techniques, such as Markov Models, Bayesian Networks, or a context-sensitive grammar. Creation of such models will of course require a larger database of popular country songs. A larger database can also improve the reliability of training the evaluation component.

Even with a feature set of size 80 the system still is lacking insight into overall song structure (such as the location and arrangement of the verse, chorus and bridge), motives other than the single most common one, and information about the variation and development of motives. So, there seems to be room for improvement in feature extraction. It is likely that not all existing features that are relevant to a song's popularity are known, so we will investigate the possibility of automatically discovering new features.

As the system becomes more mature, we intend to assemble a panel of human listeners to determine if their rankings of generated melodies agree with those of alternative evaluation architectures. We also intend to run the system for an extended period to see if the generator is able to produce a work of high quality. Although our work is at an early stage, we are encouraged that ACME might be able to recognize a musical gem inside a heap of low-scoring rubble.

References

[1] Musicarta.com, *Songwriting Techniques*, http://www.musicarta.com/songwriting-techniques.html , 2011.
[2] Noam Chomsky, *Three Models for the Description of Language*, IRE Transactions on Information Theory, 1956.
[3] Stephen Citron, *Songwriting: A Complete Guide to the Craft*, William Morrow and Company, New York, 1985.
[4] Ada Lovelace, *Sketch of the Analytical Engine Invented by Charles Babbage, Esq*, Scientific Memoirs, Vol 3 (1842)
[5] Johann Joseph Fux, *The Study of Counterpoint*, Translated and edited by Alfred Mann, Norton, New York, 1971.
[6] Martin Gardner, *White and Brown Music, Fractal Curves and 1/f Fluctuations*, Scientific American, 234(4): 16-32 April 1978.
[7] Tom T. Hall, *How I Write Songs*, Chappell Music Company, New York, 1976.
[8] Roman Klinger, Gunter Rudolf, *Evolutionary Composition of Music with Learned Melody Evaluation*, Proceedings of the 5th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics, 2006.
[9] Fred Lerdahl, Ray Jackendoff, *A Generative Theory of Tonal Music*, The MIT Press, 1982.
[10] Gerhard Nierhaus, *Algorithmic Composition*, SpringerWein, New York, 2009.
[11] Henry F Olson, *Music, Physics, and Engineering*, Dover Publications, New York, 1967.
[12] Walter Piston, *Harmony*, W. W. Norton & Company, New York, 1962.
[13] Micheal Towsey, Andrew Brown, Susan Wright, and Joachim Diederich *Towards Melodic Extension Using Genetic Algorithms*, Educational Technology and Society, 4(2) 2001.
[14] Paul Zollo, *Songwriters on Songwriting*, Writer's Digest Books, Cincinnati, 1991.