(Manufac) Turing Tests for Music

François Pachet and Pierre Roy
Sony CSL
6, rue Amyot
Paris, France, 75010

Abstract

AI systems passing a music Turing test have long been proposed in computer music research. Many AI music systems have been developed, some of them very sophisticated. However, evaluating artificially generated musical artefacts remains a most controversial issue. One of the reasons is that, contrarily to language, music does not have semantics, i.e. a shared, commonly accepted way to associate external references to musical constructs. In that light, we review here some pratical difficulties in defining a convincing music Turing test and make some proposals.

Language versus Music

The original Turing test involves a *conversation* in *natural language*, through a medium that *bypasses* the issue of spoken language rendering. The transposition of this test to music is not straightforward for several reasons: there is no such thing as a *musical conversation* (in the literal sense), there is no such thing as a *musical language* or common-sense, and *audio rendering* (including the issues of sound, timbre, voice expression, etc.) is crucial in music so it cannot be easily bypassed.

On the other hand it is extremely easy to generate *imitative* music: take a random walk on basic order-1 Markov chain estimated from a corpus of scores, for instance. It is, however, extremely difficult to generate *convincing* music. As a consequence, methods for assessing the musical quality of artificial productions have been proposed for a long time and are still debated today.

Critical Issues with Music Evaluation

The definition of a Turing test for music generation systems seems easy at first. A question like "Is this music composed by a human?" seems natural, and easy to implement in practice through various kinds of listening tests. In reality, a number of problems arise as soon as one tries to perform such an experiment. Based on our experience in building music generation systems and in numerous interactions with the community of computer music research and computational creativity, we list below the most difficult ones, from our point of view.

1. No *common-sense music* knowledge. The original Turing test involves a dialog held in basic English through a neutral medium such as keyboard/monitor. Such dialogs are possible indeed only if the system has sufficient common-sense knowledge about the world, which makes the test interesting and challenging, because we can assume that most humans share approximately the same common-sense knowledge. In the case of music, there is no clear equivalent of common-sense knowledge: music is listened to and appreciated in vastly different ways depending on the skills, culture and interests of the listener, and on the music genre. As a consequence, universal tests involving "'basic musical skills"' are difficult to define.
2. *Multi-dimensionality*. Music is awfully multi-dimensional, to the extent that it is often impossible to define the essence of a music production: is it the melody? the rhythm? the sound? the quality of the singing voice? the quality of the accompaniment? the tempo? Consequently a fantastic melody could perform poorly in tests because of a low quality rendering. Conversely, bad tunes could become hits if sung by the right person (see Sabrina Salerno's famous Boys boys boys). Eventually it is not clear what is being tested because music cannot be easily split into independent dimensions. Unlike with language, a *reductionist* approach does not work.
3. *Variety* expected. In order to rule out systems that are able to generate only very specific artefacts (e.g. small variations around essentially the same tune) a convincing test should involve not one production but a large number of them. In practice this is difficult to do because assessing a music production takes at least the duration of the production itself, so assessing a sampling of the system takes a time that grows linearly in the number of samples. Few people want to listen to hundreds of computer-generated stuff.
4. Genre *specificities*. Generating a convincing guitar accompaniment in Bossa Nova style raises problems wich are fundamentally different from, say, generating a sitar improvisation on a raga or producing one hour of lounge music for a bar on the beach. Each in of those cases, the challenges are of completely different nature. In bossa nova accompaniment the dimensions of interest are harmony (chord substitution) rhythm (subtle temporal off-

sets to create the groove), distribution of patterns (regularity in the rhythmic patterns, but also variety), sound. For lounge music, the key issue is the structure, e.g. how to build up slowly by developing a motive and how to end the tune, etc. There is no evidence so far that generating music in all these different genres and contexts have anything in common, conceptually, musically and technically. In other words, "'music"' does not really mean anything.

5. Biases against *computer-generated* stuff. Recent studies (Moffat and Kelly 2006) tend to show that there is a natural bias against computer-generated music (Moffat). Of course such a bias is cultural, and could evolve in time. But it only adds-up uncertainty in any effort to evaluate artifical systems.
6. Expecting *unexpectedness*. Even if one restricts himself to a specific music genre and context (say, songwriter's songs), there are different kinds of music, quality-wise and consequently different expectations. Do we expect "radio-friendly", flat tunes (see Jon Lajoie's (Jon) hilarious example of a perfect and boring tune), or "surprising" ones like Paul McCartney's classics ? Here again, the problems are fundamentally different.

Challenging Musical Situations (Turing-wise)

Based on the caveats mentionned above, we propose here *contextualized* experiments, as opposed to tests involving some sort of *universal* musical skills. These tests pose well-defined and challenging problems in specific music genres and address audiences loosely interacting with a musician/machine and eventually asked to tell them apart.

1. The *jazz gig* problem. Several professional jazz musicians play together, e.g. a pianist, bassist and drummer without looking at each other. They improvise on a theme (a jazz standard). Eventually they have to say whether each other musician was a machine or a human. Such a context allows human musicians to try to fool the system, by using musical devices commonly shared by jazz musicians (and see wether they are being catched up), such as doing breaks, playing patterns repeatedly, double time (transforming a 2/4 into a 4/4), etc.
2. The *singer accompaniment* problem. In a specific genre, such as Bossa nova, generate automatically a guitar accompaniment for any well-known tune of the repertoire (say "'Girl From Ipanema"'), at a given tempo, and for a given human singer. The problem involves choosing the right chords, rhythm patterns, alternating between these patterns in a coherent way, following the tempo of the singer, and of course rendering all that with an acoustic guitar sound. Singer can ask for accompaniments in the style of well-known guitarists (e.g. João Gilberto, Djavan).
3. *Interactive piano-bar* or non-metric improvisation on a theme problem. Many improvisation systems have been developped in the computer music community, but so far no system was able to produce a free-form improvisation on a given theme. There are indeed free-form improvisation systems which do not follow any structure, or at the other end, tightly constrained solo generation systems (a la Band in a Box) which follow strictly a given lead-sheet. In real improvisation settings, the musician (often a pianist) does follow a tune, but takes a lot of liberty here and there. It is still a challenging task to model such a production. The setting could involve a tester giving queries through a keyboard such as "'play Solar in the style of Monk for 4 minutes"' or "'Play Autumn leaves in bossa-nova style, 3 minutes"'.
4. *Catchy motive composition* problem (Cf. 2). Compose an 8-bar (or so) musical motive (single monodic line with chord labels) that is as catchy as the motives composed by Morricone, McCartney or the likes. The notion of catchiness is in itself challenging as the system must not know ony about music but also about what people like. This issue is related to the issue of music common-sense.
5. The *metro busking* problem. Busking in the metro (e.g. the Paris metro) is one of the most challenging task for a performing musician. Audience is very demanding, and the goal is to catch the attention of passerbys so that they stop and possibly give some money. People should stop more often with the system than with humans. In the mercantile version, more money should be collected with a system.
6. The *hit-generation* problem. Can a machine compose, arrange and produce completely autonomously a song that becomes a hit ? This may not be strictly speaking a Turing test, but is probably more challenging and interesting. Several works have addressed the issue of predicting hits (so-called "'Hit Song Science"' (Pachet 2011)) but generation has been ignored so far.

conclusion

Transposing the original Turing test to music is not straightforward for several reasons listed here. However, music raises many challenging problems, especially when produced in interactive, real-time settings.

Acknowledgments

The Flow-Machines and Lrn2Cre8 projects both address fundamental issues in music generation raised in this paper.

References

Jon Lajoie radio-friendly tune. `https://www.youtube.com/watch?v=A0Gs4xGw1Eg`. Accessed: 2014-10-22.

Moffat, D., and Kelly, M. 2006. An investigation into people's bias against computational creativity in music composition. In Colton, S., and Pease, A., eds., *The Third Joint Workshop on Computational Creativity*, ECAI 2006. Trento, Italy: Universita di Trento.

Pachet, F. 2011. Hit song science. In Tao, T. . O., ed., *Music Data Mining*. Chapman & Hall, CRC Press. chapter 10, 305–326.