SpotiBot—Turing Testing Spotify

Prof. Pelle Snickars
Department of Culture and Media Studies / Humlab
Umeå University
pelle.snickars@umu.se

In mid May 1951, Alan Turing gave one of his few talks on BBC's Third Programme. The recorded lecture was entitled, "Can Digital Computers Think?" By the time of the now lost broadcast, a year had passed since the publication of Turing's (now) famous *Mind*-article, "Computing Machinery and Intelligence", with its thought provoking imitation game. Computers of his day, in short, could not really think and therefore not be called brains, Turing argued in his lecture. But, digital computers had the *potential* to think and hence in the future be regarded as brains. "I think it is probable for instance that at the end of the century it will be possible to programme a machine to answer questions in such a way that it will be extremely difficult to guess whether the answers are being given by a man or by the machine", Turing said. He was imagining something like "a viva-voce examination, but with the questions and answers all typewritten in order that we need not consider such irrelevant matters as the faithfulness with which the human voice can be imitated."

The irony is that Alan Turing's own voice is lost to history; there are no known preserved recordings of him. The written manuscript of his BBC lecture can be found at the collection of Turing papers held at King's College in Cambridge—partly available online. As Alan Jones has made clear, Turing's radio lecture was part of a series the BBC had commissioned under the title "Automatic Calculating Machines". In five broadcasts during spring 1951, an equal number of British pioneers of computing spoke about their work. Then again, Jones was only able to examine surviving *texts* of these broadcasts. Consequently, there is no way to scrutinze or explore Turing's oral way of presenting his arguments. His intonation, pitch, modulation etcetera are all lost, and we cannot conceive the way Turing *actually* spoke.

My point to be made is that audiovisual sources from the past often tend to be regarded as textual accounts. By and large audiovisual sources have also been used by historians to a way lesser degree than classical (textual) documents. Sometimes—as the case with Turing—archival neglect is the reason, but more often humanistic research traditions stipulate what kind of source material to use. In many ways, however, the same goes within the digital humanities. Even if digitized and born-digital audiovisual material today amounts to a steadily increasing body of data to research (and do research with), it is still relatively poorly represented in the field of DH. As is well known, the focus on the modality of text is (and has remained) strong. Textual scholarship has been the discipline's core concern—from Busa's concordances, computational linguistics and the automatic analysis of vast textual corpora, to various text encoding initiatives (such as TEI) and distant *reading*.

As of lately, however, there seems to be an increased scholarly DH-interest in the media modality of sound—and a search for "sound" or "music" in this conference programme seems to suggest an expanded attention within DH. My presentation is perhaps then part of a contemporary trend of including a greater variety of media modalities within DH-research. Be that as it may; my purpose here is to provide some initial findings from an ongoing research project that deals with various experiments, interventions and the reverse engineering of Spotify's algorithms, aggregation procedures, and valuation strategies. The key idea of this research project is to 'follow music files'—rather than the people making, using, collecting or listening to them—on their distributive journey through the Spotify streaming ecosystem. Building on the tradition of "breaching experiments" in ethno-methodology, our project tries to break into the hidden infrastructures of digital music distribution in order to study its underlying norms and structures. The interventionist approach in many ways resembles the simple way a postman would follow the route of a parcel—from packaging to delivery. The setting for studying such processes include the distribution and aggregation of self-produced music/sounds through Spotify, 'monetary interventions' with calculated musical tinkering, the documentation and tracing of Spotify's history through constantly changing interfaces—and importantly, the programming of multiple bots to act as research informants.

In essence, a substantial part of our Spotify project has been about fine tuning the both highly influential and widely criticized classical Turing test. By focusing on the *deceptive qualities of technology*—particularly regarding the difference between man and machine—a number of the notions proposed in Turing's essay "Computing machinery and intelligence" have never really lost their relevance. Basically, the so called SpotiBot experiments we have conducted resemble a repetitive Turing test—i.e. our bots interact with the Spotify system, which tries to decide (via various unknown fraud detection tools) if the interaction is human or machine based. Consequently, we have asked ourselves what happens when (not if) streaming bots approximate human listener behavior in such a way that it becomes impossible to distinguish between them. Streaming fraud, as it has sometimes been labeled, then runs the risk of undermining the economic revenue models of music streaming services as Spotify.

The imitation game, Turing once stated in his 1950 essay, "is played with three people, a man (A), a woman (B), and an interrogator (C)". The object of the game was for the interrogator to determine "which of the other two is the man and which is the woman". Already at the beginning of his essay, Turing however, asked what would happen if "a machine takes the part of A in this game?" As N. Kathryn Hayles famously put it, *gender* hence appeared at the "primal scene" of humans meeting with their potential evolutionary successors, the machines. Still, following her interpretation of Turing, the 'gender', 'human' and 'machine' examples were basically meant to prove the same thing. Aware that one of the two participants (separated from one another) was a machine, the human evaluator would simply judge natural language conversation (limited to a text-only channel) between a human and a machine—designed to generate human-like responses. If the evaluator could not reliably tell the machine from human—the machine was said to have passed the test. It might then be termed artificially intelligent.

Towards the end of Turing's article, the initial question, "Can machines think?" was consequently replaced by another: "Are there imaginable digital computers which would do well in the *imitation game*?" Naturally, Turing thought so—and only 15 years later, the computer scientist Joseph Weizenbaum programmed what is often regarded as the first *bot*, ELIZA.

She (the bot) had two distinguishing features that usually characterize bots: intended functions that the programmer built, and a partial function of algorithms and machine learning abilities responding to input. ELIZA, Weizenbaum stated, "appeared capable of understanding what was said to [her] and responding intelligently, but in truth [she] simply followed a pattern matching routine that relied on only understanding a few keywords in each sentence." ELIZA was hence a mock psychotherapist—and the element of artifice programmed into her again testifies to the deceptive qualities of technology which the Turing test underlined. In fact, ever since, *fraudulence* (in one form or the other) seems to be a distinguished part of an evolving bot culture constantly capitalising on advancements in artificial intelligence. Bots *appear* to be human—which is why they are interesting.

&

Now, established in Sweden in 2006, Spotify is today the dominant player in the streaming music market. With more than 75 million global listeners, and 30 million monthly paying subscribers, Spotify has emerged as the giant within the streaming music business. Even outcompeting Apple Music, Spotify's market share currently lies above 40 percent. One success factor is arguably the ease by which a listening account can be set up at Spotify. It is, in short, extremely simple to sign in—both for humans and bots. No CAPTCHA is for example needed, and obviously, one should in this context remember what the CAPTCHA abbreviation originally stands for: "Completely Automated Public Turing test to tell Computers and Humans Apart". As is well known, a CAPTCHA is a program that protects websites against bots by generating a simple test that humans can pass but computer programs cannot. Since no CAPTCHA is needed, scripted bot users can easily be programmed to register, and within our research project we have used and deployed multiple bots to study the explicit and implicit logics of the Spotify web client. We have even been able to automate account registration (for fast bot setups). Departing from the idea that software is normative and regulatory—and, hence, that the streaming architecture of Spotify promotes and materializes certain world views (and not others)—our bots have been programmed to explore the socio-technical protocols that endow music files with cultural meaning.

One major research issue we have struggled with, is the type of knowledge that can be gained (and gleaned) from working with *bots as informants*. Are they to be trusted? Can they produce valid empirical data? We do think so. Yet, as a kind of virtual informants our bots do not interactively and explicitly *collect* information, rather they are designed and set up to acquire and log certain data via the 'actions' and different functions they are programmed to perform. Nearly all of our bots have been Spotify 'freemium users'. The setup has involved a few similar steps; firstly, bots are named, and each given certain specified (or random) characteristics (age, nationality, gender etcetera). Secondly, the bots—in the form of virtual users—are programmed to do specific tasks and hence act as research informants within the Spotify web client. Within our research project these have varied depending on what scholarly issues or tasks we have been interested in.

Essentially, the bots we have programmed are scripted algorithms that exhibit human-like behavior (in one way or the other) when 'listening' to music. Implemented in the Python programming language, and using a web UI testing frameworks, our SpotiBot engine has been able to automate the Spotify web client by simulating user interaction within the web interface. In the implementation to conduct experiments with the Spotify web client we have used a system framework originally designed for automated tests of web pages. Normally its purpose has been to validate correct behaviour of software. Consequently, our bots have been designed to program user activities and—importantly—log and record output of these. Initially, our research project designed and developed a rudimentary virtual machine (or run time engine) with the capability to execute about 15 high level machine instructions such as "register_account", "login", “logout”, “play_media” and “follow_artist”. The selected instruction set corresponded to the most common user interactions (in a web interface), and the set also included primitives for data capture (screenshots, playlists, video), as well as loops and conditional execution. Finally, the virtual machine has used the popular Web testing framework, Selenium for access and control of various web browsers.

Now, one of the major contemporary controversies regarding the transition to streaming music platforms involves payouts to artists. The sometimes heated discussion has, in short, been centered around the issue *if* streaming music will be able to generate a sustainable income for musicians—or not. Within the music industry this has led to considerable debate; Taylor Swift decided to remove her entire back catalogue from Spotify, Adele rejected streaming her new album etcetera, and lesser known artists have experimented with different music-hacks or pranks. These have ranged from the funk band Vulfpeck, and their conceptual album, “Sleepify”—containing five minutes and 16 seconds of pure silence; asking fans to stream the album on repeat (while sleeping)—to hacks by the band Ohm & Sport and their application Eternify, were for a (very) short time one could enter the name of a favorite artist and play songs on repeat for economic support in 31-second intervals.

&

At Humlab we therefore set up an experiment—the SpotiBot —with the purpose to determine if it was possible to provoke, or to some extent undermine, the Spotify business model in a similar manner. Royalties from Spotify are usually disbursed to artists (or more precisely, record labels) once a song or track is *registered* as a play, which happens after 30 seconds. The SpotiBot engine—with the ability to run a multiple of pre-programmed user bots—was hence instructed to play a single track repeatedly for more and less than 30 seconds, and sometimes simultaneously with different accounts. Tracks consisted both of self-produced music (from our research project; the artist Fru Kost with the song "Avplock") and Abba's "Dancing Queen". The fixed repetition scheme ran from 100 to *n* times. Basically, the SpotiBot engine was programmed to stick to this schedule—with three testing rounds—resulting in quite noisy computation!

From a computational perspective the Spotify web client appeared as a black box; the logics that the Spotify application was governed by was, for example, not known in advance, and the web page structure (in HTML) and client side scripting quite complex. It was not doable within the experiment to gain a fuller understanding of the dialogue between the client and the server. In addition, since our bot experiments violated (some of) Spotify's user agreements, a VPN connection was used that hid the running clients behind a public proxy IP outside of the university network.

This chart gives a graphic estimation of some results from our experiments: the SpotiBot engine was able to play Fru Kost's track, repeatedly for 25 and 35 seconds. The bot "selenium57" for example played the track 229 times, and the "selenium_bot" as many as 1,141 times repeatedly—that is, after 35 seconds of "Avplock", the bot started the song again, and again, and again. Similarly, in the second experiment, "selenium_bot37" was able to repeatedly play Abba's "Dancing Queen" for 35 seconds, at repeated intervals of 16 times, 208, 30, 1,141, 19, 20 times etcetera.

Apart from the possibility of *actually* being able to automatically (and repeatedly) play tracks on Spotify via bots, one preliminary result indicate that there was no major difference between our bots playing artist like Fru Kost *or* Abba for 25 *or* 35 seconds. For the Spotify system both artist were simply content. Our hypothesis was that playing tracks repeatedly for 25 seconds would (at least in theory) not be a problem, since such plays are not regarded as a registered play by the Spotify system. This was also true, as is evident in this chart. On one occasion the bot "selenium51" (to the left) played Abba 550 repeated times for 25 seconds. Then again, we did not discover *any* statistical difference if the same songs were played repeatedly during 35 seconds—even if they were then registered as a play (and subsequent royalties were registered). A second preliminary result was also that our third experiment (using a large number of parallel bots to play Abba's "Dancing Queen") could not be executed on available hardware since it would have required investment in a lot of new machines. In a lighter setup, however, we used more than 20 bots running in parallel on two computers (virtual machines) each interacting with a Spotify web client, repeatedly playing the same Abba track hundreds of times—and the presented chart gives a graphic estimation of some of the results.

In theory—and if we had the financial abilities—we definitively believe it would be possible to perform a massive 'bot setup' with hundreds of clients running in parallel on a larger number of (possibly cloud based) servers. Our simple experiments, in fact, indicates the ease in which hundreds of bots can be setup in parallel. In fact, this type of massive music hack (seems to) have been executed previously. The music journalist William Bedell, for example describes how he decided to "prototype a robot with an endless appetite for music to see if Spotify could detect what it was doing". Performing his hack Bedell did not "encounter many Turing tests … There wasn't even a CAPTCHA or email verification when creating accounts." His conclusion was hence similar to ours: “The barriers to entry are clearly minimal.” In addition, Bedell’s hack resembles a similar one made in 2013 by Peter Filmore. As a payments security expert, Filmore wanted to test the robustness of music-streaming services, and particularly if they had any fraud detection systems in place—which it turned out, they hadn’t.

Basically, the results from our experiments with the SpotiBot engine, confirm Bedell’s and Filmore’s hacks. The defence mechanisms used by Spotify to prevent our experiments have been insufficient—or remained unknown. In fact, it has proven more or less impossible in advance to predict how, or when, different kind of fraud detection systems have been activated (or not). It is worth stressing, however, that our bots are more advanced than the ones Bedell and Filmore programmed. They appear to have been ‘fixed programmed’ with the purpose to *only* play songs—not register and log any outcomes, nor ‘interact’ with a web client or be able to perform different tasks. Filmore’s bots were programmed in Bash, a command-line interface in Linux—indeed scalable, yet without any form of web interface interaction.

One of the core assumptions in our research project has been to use bots as informants in multiple ways. As a consequence, they have been designed as ‘programmable bots’ with the ability to receive instructions to perform different task (depending on the purpose of the intervention or experiment). Then again, even if our programmable bots were arguably more sophisticated than Bedell's and Filmore's, an important result from our SpotiBot interventions is still that a huge number of deviations did interrupt the Spotify web client, causing a number of our bots to stop playing. The SpotiBot setup was based on an *ideal* bot usage flow, and *all* deviations would in practice interrupt the client execution. This is the main reason (we believe) why so many of our bots did not perform the exact amount of repeated plays they were programmed to perform. If truth be told, nearly *all* of our bots—within the three rounds of experiments—stopped at random occasions. Most frequently deviations were caused by wrong behaviour by the bot, due to lack of knowledge of client logic. A lot of the interrupts were, in addition, caused by synchronization problems, where a bot tried to access parts of the user interface not yet loaded (or not yet visible).

Still, even though the used framework, Selenium, had a good support for these kind of errors, the software didn't always behave as we expected. To summon up our problems, the bots we programmed to listen to the Spotify web client were not only disobedient—they were inserted into a non-compliant system full of latent errors. This led to a situation where much more supervision and program correction than anticipated was needed. In addition, lots of interrupts in our experiments occurred off hour, when the SpotiBot engine was without supervision at Humlab. All in all, our bots weren't really 'battle-tested' before our experiments began, and a lot of bugs were constantly found that needed fixing.

Then again, even if we encountered a number of problems and random deviations that interrupted client execution, the general results from our SpotiBot setup do indicate that it is possible to automatically play tracks for thousands of repetitions that exceeds the royalty rule at Spotify. Listening via the Spotify web client, our SpotiBots repeatedly passed the Turing test. Admittedly, a more robust setup, with frequent and repeated testing—based on increased knowledge around client logic—would have made our bots less disobedient, and way more successful in their listening habits. Still, since our experiments do raise a number of research ethical issues, such a resilient intervention infrastructure would all likely have increased our manipulative hesitations. In the end, Abba become a tiny fraction wealthier—and indeed we also made (a very, very small amount of) money ourselves. Research can be rewarding.