Mid-term Update on the Somax System

Joakim Borg

February 1, 2022

1 Introduction

Somax is an interactive system which improvises around a musical material, aiming to provide a stylistically coherent improvisation while in real-time listening to and adapting to input from a musician. The system is trained on some musical material selected by the user, from which it constructs a corpus that will serve as a basis for the improvisation. The main idea is that Somax should serve as a co-creative agent in the improvisational process, where the system after some initial tuning is able to listen and adapt to the musician in a self-sufficient manner.

The Somax system is an ongoing development project that has already been described in a number of reports [7], [8] and more recently in [5] and [6]. This report will give a brief summary of the advances made in the design and development of the Somax system during the past two years.

2 The Somax Front-end

The Somax system consists of two main components: the back-end server, which is implemented in Python, and the front-end which is implemented in the Max programming language [1]. The front-end has gone through a number of changes during the past two years, where the most important one is the wireless paradigm described in [5]. With this update, the system has gone from supporting a single co-creative agent to being able to support an entire ensemble of co-creative agents, each of them improvising over their own musical material, where the user may specify how the agents should listen to each other and/or to the musician (or musicians) interacting with the system. This is possible thanks to a fully parallelized implementation of the back-end server, where each of the parallel agents communicates with a single, centralized scheduler to ensure that the musical timing and tempo is maintained by all agents.

The front-end has during this period also been completely redesigned with a number of improvements in the human-computer interaction domain in general, for example visual feedback to provide the musician with better understand on which bases each agent makes its decisions, interactive documentation (so called maxhelps, see [5] for details), tutorials to help new users getting started, and many other things.

3 Transformations

A co-creative agent in the Somax system improvises over the musical material on which it has been trained by essentially recombining short segments of the material in an order that is musically coherent with what the musician (or any other agent it listens to) is playing. But this also means that the system – at least originally – could not play anything that does not exist in the original material. The recently introduced concept of transformations [6] changes this fact: by utilizing pitch shifting and time stretching algorithms, each agent is now able to improvise over transposed and/or temporally altered versions of its musical material, thereby greatly increasing the number of possibilities, which in turn results in a higher coherency between the agent and its listening sources.

4 Model Improvements

A number of improvements have been made to the Somax model in order to provide the user/musician with a higher degree of control over the agents and overall increase the quality of the musical improvisation performed by each agent. Among these are a number of parameters to control which segments in the corpus the agent may use at a given time (taboo, auto-jump and regional masking – see chapter 6 in [6]), a number of adaptive algorithms to control various musical domains of the generation (loudness, duration, orchestration, tempo – again see chapter 6 in [6]) as well as a number of algorithms to control how each agent selects its output at a given time (quality threshold, probabilistic output selection [6]).

5 Embedded and Distributed Somax

In collaboration with HyVibe, a first step towards integrating the Somax system in the HyVibe Guitar has been taken. A number of optimizations of the Somax system has been made to facilitate the future integration of the system into the microcontroller used in the HyVibe Guitar (which does not have the same processing power as a modern laptop, which Somax normally runs on). An intermediate prototype has been constructed, where the back-end server of the Somax system is running on a separate microcontroller and communicating directly with the HyVibe Guitar as well as the Max front-end on a separate computer.

For this particular purpose, distributed versions of the server and front-end were implemented, which in turn allows multiple machines to communicate with a single server over a network. This means that several agents may exist on multiple machines and communicate with each other wirelessly and thereby allowing Somax to work as a fully distributed system. The original architecture, as described in chapter 4 of [5], was already designed as a local back-end server communicating with a local front-end client over a network-compatible protocol (OSC [2]), with the intention of being able to move to a fully distributed system at some point. With this in mind, no large architectural changes were needed to adapt Somax from a local system to a fully distributed one.

In terms of optimizations, the most expensive runtime operations in Somax revolve around the aspects of creating, shifting and scaling peaks (see chapter 3 and 4 of [6] for details). Here, a number of strategies to handle the peaks along a discretized time axis were implemented, in order to reduce the computational cost of the above mentioned operations. The source code for the distributed Somax architecture is available at [4].

6 Somax for Composition and Research

While Somax primarily is a system for real-time improvisation, it could just as well be used as a co-creative agent in the compositional process, which would require a slightly different workflow. A number of features have been added to facilitate this use case, for example tools for recording the improvisation and exporting the content into a score (where bar number references to the original content is maintained).

Finally, a distribution of the Somax system oriented towards offline generation of content is now available on PyPI [3], and can be used for compositional and/or research-oriented purposes.

References

[1] Cycling74 - Max. `https://cycling74.com/products/max/`. Accessed: 2022-01-31.

[2] Osc. `https://opensoundcontrol.stanford.edu/`. Accessed: 2022-02-01.

[3] Somax - PyPI. https://pypi.org/project/somax/0.1.0/. Accessed: 2022-01-31.

[4] Somax2 at dev-hyvibe. https://github.com/DYCI2/Somax2/tree/dev-hyvibe. Accessed: 2022-01-31.

[5] The Somax Software Architecture. Ircam-STMS, Technical Report, 2021.

[6] The Somax Theoretical Model. Ircam-STMS, Technical Report, 2021.

[7] Laurent Bonnasse-Gahot. An update on the SOMax project. Ircam-STMS, Tech. Rep, 2014.

[8] Joakim Borg. Somax 2: A Real-time Framework for Human-Machine Improvisation. Internal Report - Aalborg University Copenhagen, 2019.