An Investigation Into ChordRipple

Zuyu Chen

Samuel Missner

Yurui Wu

1. Introduction

ChordRipple is a chord recommendation system created by a team at Harvard University with the aim of allowing novice composers to more easily create novel chord progressions in their music. The system is based on a previously created algorithm called Chord2Vec, which is based on a popular language processing algorithm of the similar name Word2Vec. Within this paper discussing ChordRipple, the team evaluated their system through way of a user study to determine a novice composer's satisfaction with using the ChordRipple recommendations compared to more simple recommendation systems as well as how novel were the chord progressions produced by these systems.

The conclusion that was determined in the paper was that, while the ChordRipple system was not rated higher than simpler recommendation systems in terms of user satisfaction, the chord progressions created were more novel than those of the simpler system; however, the validity of these results could still be called into question since this study was done on a small sample size of nine participants, and thus, the p-values of the resultant data was considerably low.

Our aim with this reproduction is to use the ChordRipple system, along with the already created web-app that was used for the user study, and recreate the study for a new set of users in order to either increase the evidence of significance for the original results, or to bring evidence that the results gathered by the original authors is not reproducible enough to be considered valid.

2. ChordRipple – Model and System

To model the chords for recommendation, the authors train a chord embedding model, chord2vec, using the architecture of word2vec, a neural-network model that learns word embedding. The repository contains individual scripts for training skipgram, bigram and unigram model but it's not clear how the models are compared, what tests and data are used to evaluate different models, and most importantly, how the final model is trained and determined. It's confusing that although the paper claims they use skip-gram for training chord2vec, the plots of embeddings for visualization are made for bigram. And neither of the datasets used for embedding visualization is the one used for chord2vec. Even if we find the way to train the model as intended by the App, there is no metric to check how close the model we would have trained is to the model they trained.

The other issue arises from the recommendation system. The paper says the single substitution is generated by either querying the chord2vec embedding or by inward conditioning on the current context. There is no clue which query method is used in the App. The system can also provide substitutions for the existing contexts. In addition to preserving the original structure it incorporates a way to account for the similarity of the chords in the regenerated context to that of the original. However, we didn't find the code for this feature in the repository.

The lack of objective evaluation of the embedding model and of the implementation of the additional feature for the recommendation system led us to the formal evaluation presented by the paper – User Study.

3. The Repository

The original authors provided a GitHub repository for running the web application used for the user study. This repository contained a pre-trained Chord2Vec model which was used in generating the chords in the ChordRipple recommendation system. The repository also contained the web application itself, along with some scripts for analyzing user data.

This repository is where most of our problems arose with setting up the experiment. For one, most of the codebase was written back in 2014, and nothing has been updated since 2016, this meant that some of the packages,

specifically Gevent which was used for server-to-user communication, had not been updated or supported in several years. Another artifact of the code's age was that it was written for Python2, so we had to run the code out of a new virtual environment, which took a significant chunk of time to configure.

Another major problem we ran into was the poor documentation regarding package versions that were being used for this code. The text file containing the package versions was hidden within a subfolder, so we did not happen upon it until far into our investigation. On top of that, there were packages missing from the requirements, so we had no idea which version was being used. This was most important with the package Pandas, which was used extensively throughout the codebase, but without knowing what version to use, we suspect this is the reason that we couldn't extract any data from the program, which we will write about later.

Overall, we felt that this repository was not very well organized. The lack of documentation regarding running and understanding the code, which notes the lack of commenting that existed in the scripts, led to an overall difficulty understanding of what the code was doing, or how to fix the numerous problems that arose during our investigation. We did ultimately get the web application to run after much struggle, but we still encountered errors while working through the user study, which we just elected to ignore as much as we could.

4. The Web Application

The web application included 1 tutorial page followed by 3 experiment pages. Each page contains a picture, the recommendation part, and the feedback part.

The recommendation part includes a provided 8-chord progression. When clicking on any of the chords, recommendations for the currently activated chord will be generated. To help users get an idea of how the chords sound like, recommendations can be played alone, or within the context which includes chords before and after the recommendations. The feedback part asks users about their experience interacting with the application.

The tutorial page serves to help users be familiar with how to generate recommendations and use them. The three experiment pages are backed up with three systems, singleton-typical system, singleton-adventurous system, and ripple system, in random order. The order is printed to the terminal as soon as the web page is opened. The singleton-typical system generates one-chord substitutions that are typical in the current context. The singleton-adventurous system generates one-chord substitutions that are less typical, while the ripple system generates three-chord substitutions, which includes the adventurous chord as the middle chord.

As we tested with the web application, we found that it does not support inputting personalized chord progression. Which is to say, the application is tailored to experiment 2 in the paper, preventing us from reconducting experiment 1. However, for experiment 2, the application does not output novelty evaluations due to unknow reasons, so we chose to calculate progression's novelty with our own scripts.

Besides, the feedback part does not allow the users to rate their satisfaction with the finalized chord progression. The questions are not related to what the experiment is evaluating, and any answers typed in do not show in the terminal. Therefore, we decided to ignore the whole feedback part and let users rate their satisfaction orally during the experiment process.

It is not a painful process rerunning and testing the web application after the environment is configured properly. However, as pointed out in the previous part, the lack of documentation should be responsible for most of the problems we encountered, including confirming the application's functionality. The readme file in the repository hardly gave us any introduction and instruction related to the web application, except for how to start it.

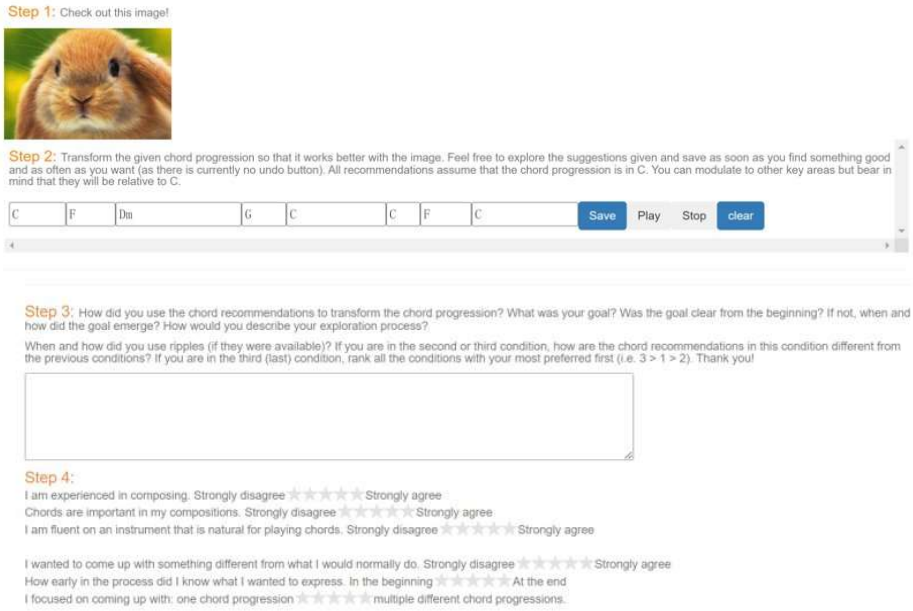

Figure 1. Screenshot of first page of the web app. Instructions are given to the user on each page for what they need to do.

5. User Study

We have recruited 7 participants for the experiments, and all of them have a music background. The participants used our experiment laptop, by either controlling it remotely through online meeting app or coming to our lab directly, to finish the experiment. In this way, we could avoid the process of reconfiguring the environment on user's own devices.

For the experiment process, we followed the guideline provided in the paper. Firstly, we gave a brief introduction of the experiment, and then some instructions on how to interact with the web app. After the participant finished the tutorial, we let them modify the provided chords freely. Then, each time when they felt satisfied with the result, we collected their satisfaction rating. The rating is scaled from 0 to 5 and can have one decimal place. Finally, after the experiment is done, we would answer any questions the participants have.

Apart from the required evaluations, we received other feedback on some features that may affect user's satisfaction with their chord progression. For example, two of our participants considered the app's playback system having a negative effect when testing with different chords. The playback system features a blur and sine wave-like timbre. Besides, chords' pitch varies greatly sometimes. These two participants could not get a clear idea of how the progressions sound like, and they would prefer the chord to be played through other instruments, such as guitar or piano. We wonder if the research teams received any similar feedback and how would they handle them.

Another special case we encountered is mainly due to the experiment procedure setup. One participant gave the first system's result a full score. However, she found that she liked the second system's result even better, and she wanted to give it a 5.1 score. This situation is possible because we were collecting each time when users finished with a system, and they did not have any reference when rating the first system's result. Again, we can never know if the research teams encountered similar situations and their solution to it. Since the order is randomized, this effect on user's rating should be eliminated if our sample size is large enough. However, considering our current sample size of 7, and the sample size of 9 in the paper, we cannot ignore it for now.

6. Evaluating Data

Our first attempt at analyzing the data from the user studies was to use the analysis tools provided for us by the repository; however, we were unable to get this to work. As mentioned before, we believe that it could be a problem

with not knowing what package versions were being used causing the program to not store any of the user data, but the scripts that analyze the data were not running, and due to the messy nature of the code, we could not figure out a way to get the preexisting code to work for our purposes; instead, we elected to analyze the user data ourselves by writing custom scripts for the statistical evaluation.

The biggest challenge with evaluating the data ourselves was knowing how the data points were originally made. For example, the equation shown in the figure below(Figure 2) shows the equation for calculating novelty as shown in the original paper. This equation can be read as the novelty is the sum of the inverse of the number of occurrences of a chord from the user's progression within the chord database, which is said to be the ROCK corpus database in the paper, for each chord in the progression; however, we don't know which version of the dataset was being used from the repository, so it's possible they were only comparing to the training set or the testing set instead of the entire set, which is how we calculated it.

Similar problems occurred in calculating the statistical values of our data since we couldn't decipher exactly how these values were being calculated. The paper references using a Friedman Square test for obtaining a comparative test statistic, and the evaluation code from the repository had functions for computing this, but we weren't exactly sure what the inputs and outputs were supposed to look like for these systems. We ended up using Scipy functions to evaluate these statistical values, but yet again, without being able to use the original code, knowing that everything works the same way is essentially impossible.

$$novelty(c_{0\ldots T-1}) = \sum_{i=0}^{T-1} \frac{1}{N_{ci}}$$

Figure 2. Function for calculating chord progression novelty.

7. Results

First, although only two participants are missing, the standard deviations among our user data are much greater than those of the paper, especially the one for Novelty. The rank of our user ratings across 3 conditions is completely different than the paper. Ours ranked Ripple > Atypical > Typical. The paper's user ratings ranked Typical > Ripple > Atypical. But whether it's our user study or the paper's, both indicate a novelty rank as Atypical > Ripple > Typical. The cause for the big novelty value in our user data could be the paper and our team used two different datasets. The paper may have a bigger Rock corpora dataset to calculate novelty than the dataset we found and used for our study.

```
df.groupby("Type").Ratings.describe().iloc[:, 1:3].T
```

Type	0	1	2
mean	3.957143	4.071429	4.242857
std	0.937829	0.826064	1.239432

```
df.groupby("Type").Novelty.describe().iloc[:,1:3].T
```

Type	0	1	2
mean	0.758948	1.482698	1.292712
std	0.595136	0.999969	1.036617

Figure 3. Mean and standard deviation of user ratings (upper) and novelty (below) on their best-ranked chord sequence across three recommendation types (0: Typical, 1: Atypical, 2: Ripple) in our user study

Measure	SINGLETONTYPICAL	SINGLETONATYPICAL	RIPPLE
Self-rated satisfaction	4.5 (0.707)	4.06 (0.682)	4.222 (0.939)
Novelty	0.021 (0.035)	0.147 (0.091)	0.063 (0.085)

Figure 4. Mean and standard deviation of user ratings (upper) and novelty (below) on their best-ranked chord sequence across three recommendation types in the paper's user study

However, the Friedman test performed on our user data yields a way big p-value in both ratings and novelty. It's too big to be considered simply due to the small sample size we have. So, we lean to accept the null hypothesis that chord recommendation type has no significant effect on the user ratings and the novelty of their best-ranked chord sequence.

```
import scipy.stats as sts
sts.friedmanchisquare(table_ratings.iloc[:,0], table_ratings.iloc[:,1], table_ratings.iloc[:,2])
```

```
FriedmanchisquareResult(statistic=1.4615384615384506, pvalue=0.48153843340713176)
```

```
sts.friedmanchisquare(table_Novelty.iloc[:,0], table_Novelty.iloc[:,1], table_Novelty.iloc[:,2])
```

```
FriedmanchisquareResult(statistic=0.2857142857142776, pvalue=0.8668778997501851)
```

Figure 5. The Friedman Test results for user ratings and novelty data in our study

8. Conclusion

The world of statistical machine learning is one that has grown immensely in the past couple of years, with many experiments, including this one, being made in attempts to create something new that could provide the user with novel tools to perform various technical tasks in various fields, in this case, that means music composition; however, due to the rapid progress of this field, many projects can find themselves without upkeep, and it only takes a small handful of years for an experiment to end up at a point where recreation becomes extremely difficult.

ChordRipple is one such project that seems to have befallen that fate. The original paper was only able to find a small significance for the use of this program to help amateur composers create novel music that was to their liking. The low value of the project, combined with what I'm sure were fruitful and widespread careers by those involved in this project, has left ChordRipple to be a somewhat forgotten technology. Sure, the original developers probably took skills and tools that they discovered along the development into their future work, but this specific effort seems to have been somewhat forgotten,

Our results from running our own user study were inconclusive in an effort to prove the original hypothesis. While this may be due to the lack of cooperation of the code to perform the statistical analysis the exact same way they did it, we can never know for sure, and we can just assume that the original experiment had some sort of bias to show what significance they did find.

ChordRipple would have been a novel idea from the time of inception, but since then many similar chord recommendation algorithms have been created, some more recent ones most likely being more powerful due to the advancing of machine learning techniques, but this paper and the Chord2Vec algorithm that inspired it will still be seen as a step in the advancement of this corner of music technology.