MusicTXT: A Text-based Interface for Music Notation

Kelian Li, Wanwan Li

MusicTXT: A Text-based Interface for Music Notation

Kelian Li[1]*, Wanwan Li[2]*

[1]Center for Music Technology
Georgia Institute of Technology

[2]Department of Computer Science
George Mason University
* Joint First Authors

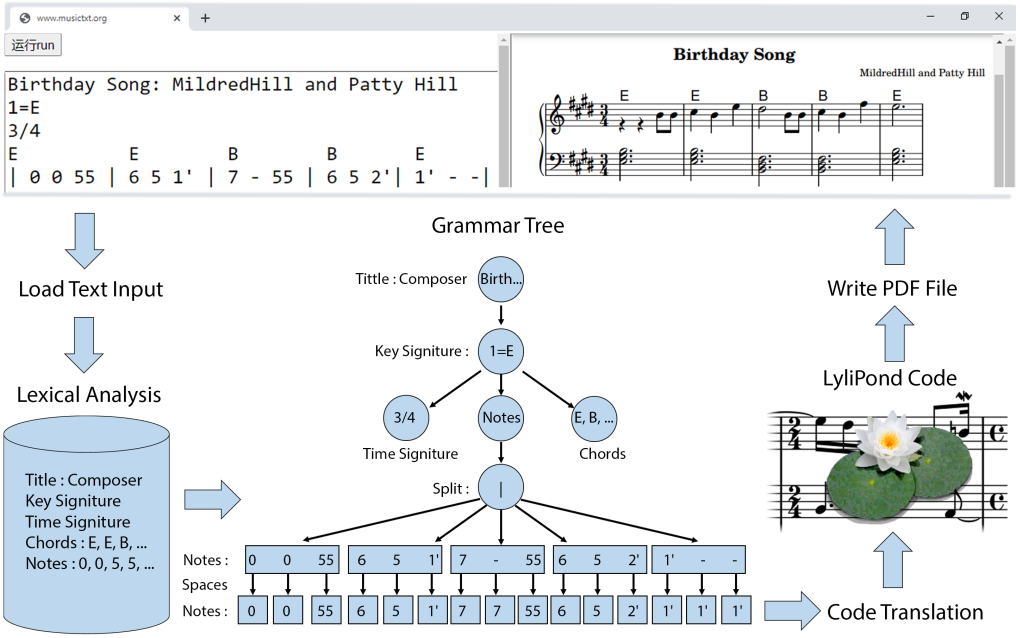


Figura 1. Pipeline of MusicTXT user interface.

Abstract. *For most music notation software, due to the complex nature of modern staff notation, extra mouse interactions and manual efforts are highly demanded. According to this observation, we propose MusicTXT, a plain text-based user interface for music notation that is almost mouse interaction-free. Based on our easy-to-learn online user interface, users are able to notate music by typing a paragraph of numbers and alphabets as plain text. We validated the music notation efficiency of our interface by comparing it with another popular online software-NoteFlight. According to statistical analysis, we prove there is a significant improvement in music notation efficiency by using MusicTXT.*

1. Introduction

As music production becomes democratized[Galuszka and Brzozowska 2017], more people, including nonprofessionals, step pace into the music production industry. Typically, music production speed seriously depends on music notation efficiency[Mauch et al. 2015]. Especially, musicians tend to prefer the software which can deliver their musical expression with very little time and manual effort. Unfortunately,

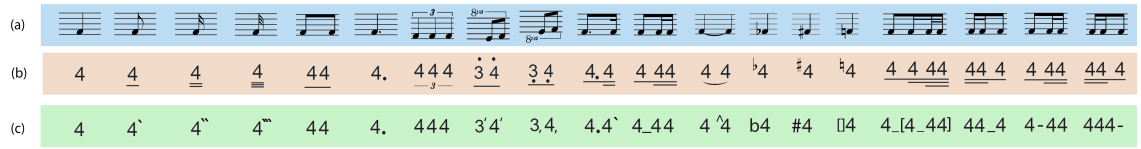


Figura 2. Comparison among the (a)modern staff notation, (b)numbered musical notation, and (c) our plain text-based notation.

most of the existing music notation software[Byrd 1994] share the common drawback that it takes extra manual efforts to notate the music due to the heavy mouse interactions.

Therefore, such inconvenience of those existing music notation software lifts the learning curve for most music composers as beginners, especially for nonprofessionals. These high demands of complicated interactions even strangled some of the beginners' interests in music production. Especially, in music education programs, younger children cannot be able to start the hands-on practice with those music notation software in a timely manner[Gudmundsdottir 2010].

Inspired by the numbered musical notation which is firstly invented by Jean-Jacques Rousseau[Rousseau 2009], people realize that using numbers instead of using notes increases the efficiency to express the music in reading and writing[Winangsit and Sinaga 2020]. In China, numbered musical notation, also called "JianPu", successfully turns down the beginners' learning curves for playing the Chinese musical instruments such as Guzheng[Gaywood 1996], Pipa[Myers and Myers 1992], and Erhu[Stock 1993], and "JianPu"is widely accepted by Chinese musician as a standard music notation system.

According to this observation, we design MusicTXT, a novel efficient text-based user interface for music notation that is easy-to-learn and easy-to-input. Our user interface takes advantage of such numbered musical notations and move a step forward by introducing a novel plain text-based music notion system. With our music notation interface, musicians are able to get rid of heavy mouse interaction by simply typing a paragraph of numbers and alphabets as plain text. The contributions of our work include:

- We design a novel plain text-based music notation system that extends the existing modern numbered musical notation.
- We devise a computational algorithm to automatically convert the plain text music notation into the standard music score as a PDF file and playable audio MIDI file using the LilyPond library[Nienhuys and Nieuwenhuizen 2003].
- We develop an online website-based Graphics User Interface (GUI) for our MusicTXT platform which can be reached at `http://www.musictxt.org/`.
- We have validated the music notation efficiency of MusicTXT by comparing it with another popular web application-NoteFlight[McConville 2012].

2. Numbered Musical Notation

With different cultural backgrounds, musical notation has its different expression forms. For example, one of the earliest musical notions is found by the Ancient Near East in about 1400 BC known as Music of Mesopotamia [Duchesne-Guillemin 1984]. Within

thousands of years of progression, modern music notation is most widely accepted as Modern Staff Notation (The standard notation)[Rastall 1983]. Modern Staff Notation[Gerou 1996] employs the staff lines upon which pitches are indicated by placing oval note heads on or between the staff lines. As another popular form of musical notation, numbered musical notation, also known as Ziffernsystem[Klassen et al. 1959], is widely accepted within China or some other countries such as Japan, Indonesia, Australia, and Ireland, etc. As a natural way to indicate the pitches using numbers, numbered musical notation gained wide popularity among beginners or none-professional music composers[Jiang et al. 2006]. Also, text-based notations, such as LATEX[Suyanto 2019], have been proposed for representing numeric music notation systems. Besides, lots of existing software provides a convenient GUI for users to type their numbered music score by dragging and clicking such as JP-Word[Word] enables the user to notate the numbered music score like using a word. However, due to the limitation of numbered music notation's nature, most of the exciting software needs lots of unnecessary mouse interactions so that be able to create the correct numbered music score. Those limitations of the numbered music notation theoretically make it impossible for the user to compose music without too many mouse interactions. Therefore, we made a further step forward to adjust the standard numbered musical notation into a novel plain text-based notation which is similar to the standard one but needs fewer mouse interactions as it contains only plain text. We provide a user-friendly web page-based GUI (Graphical User Interface) as an easy-to-use platform for music notation. Our website can be visited through this link `http://www.musictxt.org/`, our demo video can be viewed through this link `https://youtu.be/yFTbm7tpLII`, more usage details can be found in Section 4 under the subsection of Code Translation.

3. Text-based Musical Notation

Plain text-based music notations are always interesting topics to explore[Read 1987]. As the basis of our MusicTXT interface, we design a novel plain text-based musical notation system that takes further improvements upon the standard numbered musical notation. As shown in Figure 2, the comparisons among the modern staff notation, numbered musical notation, and our plain text-based notation are listed. Users who are using our tool can learn such notation and create their own music work with only typing letters and symbols on the keyboard just like write a paragraph of sentences in a Word document. As our proposed text-based music notation roughly follows the styles that the numbered musical notation possesses, it is easy to master for those beginners who have any knowledge of the stand numbered musical notation which is nowadays widely accepted by both professionals and non-professionals. Given the fact that we hope to reduce the mouse interactions during the music notation process, as such mouse interactions seriously slow down the music notating process, we introduce some differences between the traditional numbered musical notation and our proposed notation. Those differences are listed below:

- Eighth note is represented as the half duration of a quarter note (a single number, say fa, 4) by appending the number with one back quote (4‘). Similarly, two back quotes for sixteenth note (4“) and three back quotes for thirty-second note (4“‘).
- Beamed notes are connected with eighth notes that there are no spaces between numbers. We separate different beats using spaces. All numbers concatenated without spaces results in one beat. For example, three numbers connected without spaces represents a triplet.

- Octave signs are represented by appending the number with one single quote (') or one comma (,) where single quote (') represents one octave higher and comma (,) represents one octave lower.
- Common accidentals are represented by symbols appears before a number including b (Flat), # (Sharp) and [] (Natural).
- Beats subdivision are represented by underlines (_) which equally divide one beat into two half beat (Recursive method) or are represented by dash lines (-) such as if a note's duration is prolonged twice it can be followed by a dash line (Iterative method). More details will be clarified in Section 4.
- Other symbols such as a tie (∧) which indicates that the two notes are joined together, repeat signs are represented as (|: :|), volta brackets are represented as ([1]... [2]...) and chords are represented by alphabets such as (C, G, Am, ...).

4. MusicTXT Interpreter

As the essential algorithm to convert users' plain text input into readable music score documents such as PDF files, we develop a computer program as the interpreter[Reynolds 1972] to achieve such a translation process. Our defined high-level language, as illustrated in Section 3, is transformed into the low-level data structure that is understandable by machine through this interpreter. Main pipeline of our interpreter includes: Lexical analyzing, grammar tree parsing, and Lilypond code translating. In this section, we will explain the details of our algorithm to implement the interpreter.

4.1. Lexical Analyzer

As the first step for an interpreter to work with, lexical analysis[Hanks 2013] is applied to the plain text input. Like a standard compiler, a lexical analyzer is used to identify commands, keywords, and special symbols when scanning the input text. In our proposed user interface of MusicTXT, we define a framework for lexical analysis which includes recognizing title, composer, key signature, time signature, measures and musical symbols. After the user's specifying that information at the beginning of a plain text, a couple of lines of strings are used to specify the notes of the whole music. During lexical analyzer's scanning of the whole content of input text, the user's information specified through keywords such as "Title : Composer", "1=C", and "4/4"are analyzed and stored in the interpreter. Afterward, each line of the string is separated into measures according to the symbols of vertical bars. Each measure is separated into several beats according to the symbols of spaces. Each beat will be separated according to two methods: iterative method or recursive method. After iterating each line in the plain text input, a group of words is created by the lexical analyzer. Through such words list, it can create a tree data structure called grammar tree to interpret the input text as music notations.

4.2. Grammar Tree

After the lexical analysis, a list of words is generated. In order to convert those words into sheet music, a tree data structure is created. In the compiler system, such a tree data structure is typically called a grammar tree[Kovács and Barabás 2011]. The grammar tree is efficient to capture and analyze any recursive expressions and is widely used in modern compiler systems[Grune et al. 2012]. According to an observation that music notation expression naturally possesses a recursive feature[Armand 1993], we use a grammar tree to parse the words analyzed through the lexical analyzer into the final sheet music.

As shown in Figure 3, a sentence in Ode to Joy by Beethoven is written in our music notation. After the lexical analysis, we extract the numbers in plain text. Then according to the grammar tree that we have built from such a sentence, we assign the duration for each number and convert those numbers into meaningful music notes. The note duration d_i calculations are achieved through a recursive mathematical formula $d_i = \frac{d_{i-1}}{n_{i-1}}$ where i is the depth of the node in the grammar tree and n_i is the total number of nodes in that depth. For example, if the sentence is: | 3 3 4 5 | 5 4 3 2 | 1 1 2 3 | 3. 2‘ 2 - | and as user-specified that there are 4 beats each measure and each sentence has 4 measures, therefore, the during a node at the root level is $d_0 = 16$ beats and number of nodes at the root level is $n_0 = 4$ measures. According to the above formula, the during for the nodes in the 1st level is $d_1 = 16/4 = 4$ beats. Similarly, we can have the nodes in the 2nd level is $d_2 = 4/4 = 1$ beat. This formula is flexible for analyzing the rhythms including both invariant and variant measures.

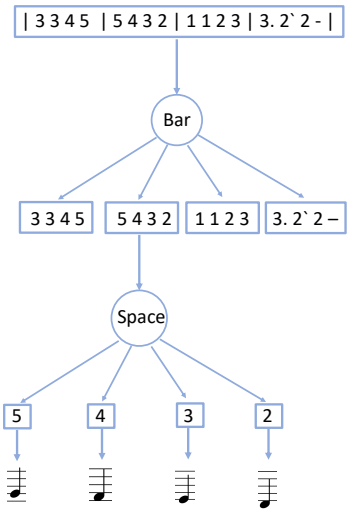


Figura 3. Grammar Tree.

With respect to the representation of beamed notes, we provide two separate interpreting methods of how to further subdivide one single beat into half or less. One being the recursive method while another being the iterative method. According to different users' preferences of reading the music sheet, the understanding of beamed notes are typically separated into two ways: some of the users are trying to subdivide one beat into two halves or further subdivide the half-beat while some other users are trying to subdivide one beat into many equal minor parts and count how much of the portions are occupied by each note. Hereby, we use the recursive method to satisfy the users who are reading the music in the first way while using the iterative method to attract users who prefer the second way. More detailed explanations are itemized below:

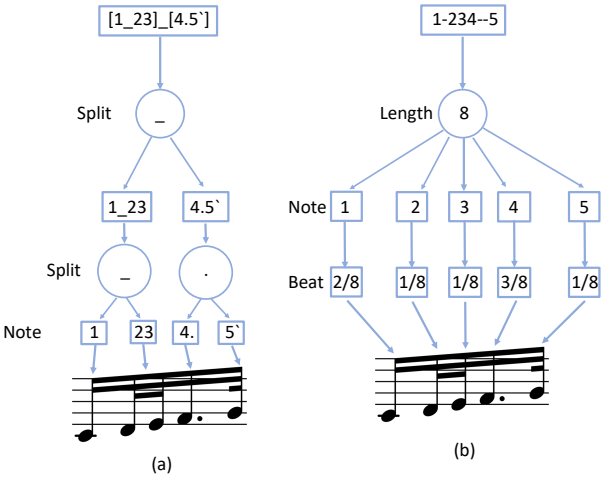


Figura 4. Beat subdivisions. (a) Recursive method. (b) Iterative method.

- The recursive beat subdivision is represented by underlines which equally divide one beat into two half beat. For example, in a measure noted as 32_4, 32 takes half-beat together, 4 takes half-beat itself, 3 and 2 take a quarter beat individually.
- The iterative beat subdivision is represented by dash lines. A note's duration is prolonged if it is followed by a dashed line. For example, in a measure represented by 32-4, 2 is prolonged and takes half-beat, 3 and 4 take a quarter beat individually.

Here we demonstrate a more complex example to show how are these two methods applied to our MusicTXT interpreter. As shown in Figure 4, we have one beat consisting of five notes which are 1, 2, 3, 4, 5 where 1, 2, and 3 take half-beat while 4 and 5 together

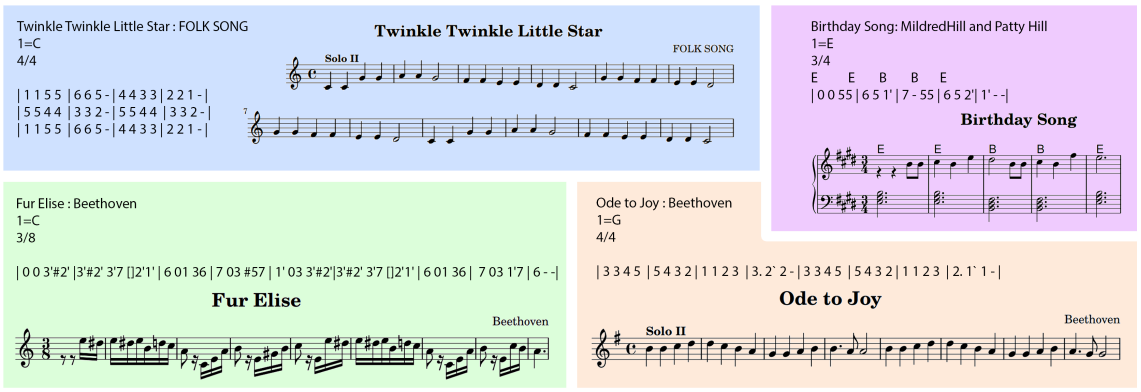


Figura 5. Examples: Inputs are the plain text-based notation of four songs and the outputs are the modern staff notation of the corresponding inputs.

take another half beat. In the recursive beat subdivision method, we use brackets [and] to specify the priority of subdivision order so that the notes in a bracket will be subdivided further. The subfigure in (a) is the recursive beat subdivision process while the notes are texted as [1_23]_[4.5‘] and (b) is the iterative beat subdivision process while the notes are texted as 1-234–5. These two representations result in the same music score in modern staff notations. As we can see that the recursive (first) representation is more structurally resembling the modern staff notation while the iterative (second) representation seems more convenient for the user to express their music in a straightforward way.

4.3. Code Translation

After the grammar tree has been built from the users’ input as text, we translate it into the LilyPond[Nienhuys and Nieuwenhuizen 2003] code according to the process and algorithms clarified above. LilyPond, as a part of the GNU project, is a well-known program to produce sheet music through a programming language, free to use and is widely used for online music notation applications[Solomon et al. 2014]. By virtue of this convenient software for music sheet encoding, we achieved the music notation process by translating our MusicTXT script into the Lilypond script. By invoking those API functions in Lilypond’s Python libraries, Lilypond’s code translation process is integrated into our web-based GUI interface. As shown in Figure 1, we develop a web-based GUI for users to input plain text and run their scripts. After users load the web-page and clean the website buffer by pressing "Ctrl+Shift+R", user can type their scripts of the music in plain text. To generate the corresponding sheet music, they can click the "Run"button. During the music sheet generation process, we first load the input text and apply the lexical analysis, then we generate the grammar tree data structure as mentioned above. In the end, we convert the grammar tree into a music sheet by invoking the LilyPond APIs, finish Lilypond’s code translation process, and generate the PDF file for display. We also provide the buttons for downloading the synthesized music in the formats of PDF file, midi file, and LilyPond script file. As shown in Figure 5, the input is the plain text-based notation of four songs and the output is the standard modern staff notation of the corresponding inputs. Different pieces of music are highlighted in different colors including blue (Twinkle Twinkle Little Star), green (Fur Elise), yellow (Ode to Joy), and pink (Birthday Song).

5. Experiments

To validate the efficiency of using our proposed user interface, we compared MusicTXT with another well-known music notion software NoteFlightn[Richmond 2015], which is an online music composing application that lets users create, view, print, and hear professional quality music notation through a web browser. NoteFlight is well accepted by lots of professional music composers as it is install-free, easy-accessible, free-to-use, and incorporated straightforward mouse-keyboard interactions that are easy-to-learn. Therefore, we choose such a successful music notation software of NoteFlight as our counterpart, which is able to prove the efficacy of our MusicTXT if it overperforms NoteFlight.

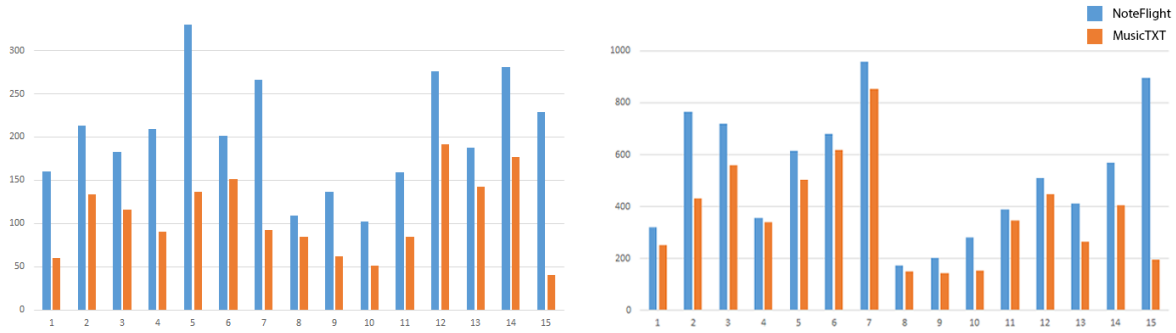
Participants. We recruited 15 users to compare the efficiency using MusicTXT with the one using NoteFlight. All of the users have some music background, most of them are from the music majors. All of them can read and write music score. They have background in reading music both using standard modern staff notation and the numbered notations. Some of them have a background in using NoteFlight. Before, the experiments, we give the users enough time to learn how to use NoteFlight and how to use MusicTXT. From our observations, users who are not familiar with both usually take about 10-20 minutes on average in learning MusicTXT, while takes 20-30 minutes in learning NoteFlight, most users tend to learn MusicTXT faster than NoteFlight.

Efficiency Test. The efficiency test is used to evaluate how much time needed to finish typing the same music using different interfaces. When the users are ready to start the efficiency test, the researcher will ask them to transcribe two pieces of music in the NoteFlight and MusicTXT respectively where the first task being the Twinkle Twinkle Little Star and the second task being the Fur Elise. For each task, the researcher will record the time in the beginning and the end, then the duration can be calculated to measure how long it takes to finish each task. During the experiments, half of the users are randomly chosen to use NoteFlight first while the remaining half use MusicTXT first. As we know that Twinkle Twinkle Little Star is much easier than Fur Elise to read and write, therefore, we test the interaction efficiency of our interface for both easy songs and difficult songs. Their efficiency test results will be explained and discussed in Section 6.

6. Results and Discussions

The efficiency test results are shown in Figure 6 where the time (in secs) that the users have taken to finish writing two songs in two different software are plotted. The numbers are the seconds that have been taken by the users to finish writing the music score of the song named Twinkle Twinkle Little Star (Task1) and Fur Elise (Task2) using NoteFlight (Colored in blue) and MusicTXT (Colored in orange) respectively. According to the descriptive statistics, for Task1, the average finishing time using NoteFlight and using MusicTXT are 202.8 sec and 107.6 sec respectively. For Task2, the average finishing time using NoteFlight and using MusicTXT are 523.2 sec and 377.2 sec respectively. Furthermore, we analyze whether there is any statistically significant difference among two different software (NoteFlight and MusicTXT), we applied two factors ANOVA tests (with replication)[St et al. 1989] to evaluate users' efficiency on both two tasks. Two factors null hypothesis includes:

- *Among groups*: there is no statistically significant difference between the users' efficiency using NoteFlight and using MusicTXT.



(a) Twinkle Twinkle Little Star (b) Fur Elise

Figura 6. The result of efficiency test. The numbers are the seconds that have been taken by the users to finish notating the music score of the song named (a) Twinkle Twinkle Little Star and (b) Fur Elise using NoteFlight (Colored in blue) and MusicTXT (Colored in orange) respectively.

- *Among columns*: there is no statistically significant difference between users' efficiency in finishing Task1 and Task2.
- *Interaction*: there is no statistically significant interaction between these two factors, namely, the different software and different tasks.

By setting $\alpha = 0.05$ (95% confidence interval), we get the ANOVA test result showing that among different software (NoteFlight and MusicTXT) $P_{\text{value}} = 3.66E{-}9 < 0.05$, among different columns (Task1 and Task2) $P_{\text{value}} = 0.005986 < 0.05$, and the interaction $P_{\text{value}} = 0.54 > 0.05$. Therefore, with 95% confidence, we reject the null hypothesis among groups that there is no statistically significant difference between different software. And additionally, with 95% confidence, we reject the null hypothesis among groups that there is no statistically significant difference between different tasks. Alternatively speaking, with 95% confidence, we conclude that the users using MusicTXT are much more efficient than using NoteFlight. At the same time, users finish the Task1 much faster than Task2. This result basically suggests two facts: (1) Compared with NoteFlight, MusicTXT is easier and faster to use when typing the same music and (2) Task2 (Fur Elise) is harder than Task1 (Twinkle Twinkle Little Star), which seems reasonable.

7. Conclusions

In this paper, we present MusicTXT, an innovative interface for online music notation using plain text as the input. Our algorithm is designed to converting users' input of numbered-notation-like plain text into readable music scores in standard modern staff notation, we develop a computer program as the interpreter to achieve such a translation process. Where the major steps of our interpreter include: Lexical analysis, grammar tree parsing, and Lilypond code translation. To validate the efficiency of using our proposed user interface, we compared MusicTXT with another well-known music notion software-NoteFlight. Given two different tasks, by measuring how much time needed for users to finish typing the same music using different software, we statistically calculated the average finishing time. After applied two factors ANOVA tests, we concluded that MusicTXT is easier and faster to use than NoteFlight and the second task is harder than the first task which seems quite obvious given to the first glance.

According to these inspiring findings, in the future, we will move along this direction and propose more interesting music notation interfaces that are based on text. Some functions are expected to be integrated such as enabling midi input which can be converted into plain text according to our notation rules, handling Fugue, adding lyrics, and supporting complex rhythms. Also, there can be some other exciting explorations in the near future. For example, it can be incredible if we can write and send a piece of music on Wechat just like writing a text message, the user on the other side receives such message as a standard music score or a playable sound. Also, by taking advantage of AI music composition technology, when people write their own music like writing text, the system can automatically suggest the chords beneath the text, this working process looks just like when someone is using the google doc there is another collaborator writing the comments at the same time. Rich text can be explored to simplify the current grammar by adding underlines or bold font to specify particular meanings. Also, given the rich text-based interface, custom-designed graphic notations can be introduced to help users define their own notations through graphical symbols designed by themselves. Further more, in the future our proposed grammar and syntax can be extended to be capable of coding contemporary music such as Xenakis and Ferneyhough, etc. Overall, there can be a couple of more directions to explore given to our interface of MusicTXT, this will cast a light upon the future as the text-based music notation is easier to use.

8. Acknowledgment

Thanks for the participants in user study and their feedback. Especially, we thank Zihan Xiong, Wei Xiao, Yihe Wang, and Syu Kevin for their participation.

Referências

Armand, J.-P. (1993). Musical score recognition: a hierarchical and recursive approach. In *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)*, pages 906–909. IEEE.

Byrd, D. (1994). Music notation software and intelligence. *Computer Music Journal*, 18(1):17–20.

Duchesne-Guillemin, M. (1984). *A hurrian musical score from ugarit: the discovery of mesopotamian music*. Undena Publ.

Galuszka, P. and Brzozowska, B. (2017). Crowdfunding and the democratization of the music market. *Media, Culture & Society*, 39(6):833–849.

Gaywood, H. R. A. (1996). *Guqin and Guzheng: the historical and contemporary development of two Chinese musical instruments*. PhD thesis, Durham University.

Gerou, T. (1996). Essential dictionary ofmusic notation.

Grune, D., Van Reeuwijk, K., Bal, H. E., Jacobs, C. J., and Langendoen, K. (2012). *Modern compiler design*. Springer Science & Business Media.

Gudmundsdottir, H. R. (2010). Advances in music-reading research. *Music Education Research*, 12(4):331–338.

Hanks, P. (2013). *Lexical analysis: Norms and exploitations*. Mit Press.

Jiang, Y.-n., Zhang, Y., and Zhang, S.-Y. (2006). Research of numbered musical notation recognition method. *Jisuanji Gongcheng yu Yingyong(Computer Engineering and Applications)*, 42(32):204–206.

Klassen, P. J., Bender, E., and Bender, H. (1959). Ziffersystem (numerical musical notation). *Global Anabaptist Mennonite Encyclopedia Online*.

Kovács, L. and Barabás, P. (2011). Experiences in building of context-free grammar tree. In *2011 IEEE 9th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 67–71. IEEE.

Mauch, M., Cannam, C., Bittner, R., Fazekas, G., Salamon, J., Dai, J., Bello, J., and Dixon, S. (2015). Computer-aided melody note transcription using the tony software: Accuracy and efficiency.

McConville, B. (2012). Noteflight as a web 2.0 tool for music theory pedagogy. *Journal of Music Theory Pedagogy*, 26:265–289.

Myers, J. E. and Myers, J. (1992). *The way of the pipa: Structure and imagery in Chinese lute music*. Kent State University Press.

Nienhuys, H.-W. and Nieuwenhuizen, J. (2003). Lilypond, a system for automated music engraving. In *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, volume 1, pages 167–171. Citeseer.

Rastall, R. (1983). *The Notation of Western Music: An Introduction*. London: JM Dent & Sons.

Read, G. (1987). *Source book of proposed music notation reforms*. Number 11. Greenwood.

Reynolds, J. C. (1972). Definitional interpreters for higher-order programming languages. In *Proceedings of the ACM annual conference-Volume 2*, pages 717–740.

Richmond, K. (2015). Noteflight: The future of music notation. *The American Music Teacher*, 64(6):69.

Rousseau, J.-J. (2009). *Essay on the origin of languages and writings related to music*, volume 7. UPNE.

Solomon, M., Fober, D., Orlarey, Y., and Letz, S. (2014). Providing music notation services over internet. In *Linux Audio Conference*, pages 91–96.

St, L., Wold, S., et al. (1989). Analysis of variance (anova). *Chemometrics and intelligent laboratory systems*, 6(4):259–272.

Stock, J. (1993). A historical account of the chinese two-stringed fiddle erhu. *The Galpin Society Journal*, pages 83–113.

Suyanto, Y. (2019). The dynamics symbol representation features of numeric music notation in latex. In *2019 5th International Conference on Science and Technology (ICST)*, volume 1, pages 1–6. IEEE.

Winangsit, E. and Sinaga, F. S. S. (2020). Writing music through parnumation 3.0 in the musical activities learning process. In *1st International Conference on Lifelong Learning and Education for Sustainability (ICLLES 2019)*, pages 31–34. Atlantis Press.

Word, J. P. J. Jp-word tutorial. `http://www.happyeo.com/intro_jpw.htm`.