NLP4MusA 2021

Proceedings of the 2nd Workshop on NLP for Music and Spoken Audio (NLP4MuSA)

12 November, 2021
Online

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-950737-19-2

Introduction

Welcome to the 2nd Workshop on NLP for Music and Spoken Audio. The aim of NLP4MuSA is to bring together researchers from various disciplines related to music and audio content, on one hand, and NLP on the other. It embraces the following topics.

- NLP architectures applied to music analysis and generation
- Lyrics analysis and generation
- Exploiting music related texts in music recommendation
- Taxonomy learning
- Podcasts recommendations
- Music captioning
- Multimodal representations

The workshop spans one day split into two days to accommodate an online format while preserving a timezone friendly schedule, which features both live and asynchronous presentations and Q/A sessions. The main topics covered in the accepted papers

The talks of our keynote speakers highlight topics of high relevance in the intersection between music, audio and NLP. The presentation by Yunyao Li discusses the challenges posed by the current *Wild West* of NLP research. Invited speakers cover different areas in at the crossroads between Music, Spoken Audi oand NLP, in particular: Longqi Yang focuses on goal-directed music recommendation; Markus Schedl describes different approaches for emotion-aware music exploration; Juham Nam provides a review on music auto-tagging; and finally, Anna Huang discusses a preliminary approach to "tuning" Music Transformer.

In total, we accepted 8 papers (47% of submissions), following the recommendations of our peer reviewers. Each paper was reviewed by three experts. We are extremely grateful to the Programme Committee members for their detailed and helpful reviews.

Sergio Oramas, Elena Epure, Luis Espinosa-Anke, Rosie Jones, Mohamed Sordo, Massimo Quadrana and Kento Watanabe

Online

November 2021

Invited Talks

Yunyao Li: Taming the Wild West of Natural Language Processing

Natural language processing (NLP) is becoming increasingly adapted in the real-world. To many, NLP is the new resource of growth and wealth. However, the NLP landscape is like the Wild West now: many and growing numbers of players, fast innovations, and limited oversight. In this talk, I will discuss the major challenges in taming the Wild West of NLP. I will present our work in recent years in in addressing these challenges. I will showcase some of the work in concrete domains (e.g. compliance). I will also share thoughts on a general approach towards adapting NLP to solve real-world problems.

Longqi Yang: Towards Goal-directed Content Recommendation

People's content choices (e.g., Podcast, music, etc.) are driven by their short-term intentions and long-term goals, which are often underserved by today's recommendation systems. This is mainly due to the fact that higher-ordered goals are often unobserved, and recommenders are typically trained to promote popular items and to reinforce users' historical behavior. As a result, the utility and user experience of content consumption can be affected undesirably. This talk will cover behavioral experiments that quantify the effects of goal-agnostic recommenders and algorithmic techniques to improve them.

Markus Schedl: Using NLP for emotion-aware music exploration, lyrics and playlist analysis

In this talk, I will showcase the use of NLP techniques for several music-related tasks, which are carried out at the Institute of Computational Perception of the Johannes Kepler University Linz. More precisely, I will briefly introduce our latest research on lyrics analysis, text-based playlist clustering, and emotion-aware music exploration and recommendation.

I will report findings of our studies on genre and temporal differences of song lyrics, and on uncovering the extent to which the sequential ordering of tracks in user-generated playlists matters for different playlist types identified by their title. Furthermore, I will briefly introduce EmoMTB, our emotion-aware music exploration and recommendation interface which adopts emotion recognition techniques from user-generated texts.

Juhan Nam: Music Auto-Tagging: from Audio Classification to Word Embedding

Music auto-tagging is one of the main audio classification tasks in the field of music information retrieval. Leveraging the advances of deep learning, particularly, convolutional neural networks for image classification, researchers have proposed novel neural network architectures for music to improve the annotation and retrieval performances. However, this classification approach has the limitation that the model can handle only a fixed set of labels that describe music and does not consider the semantic correlations between the labels. Recent approaches have addressed the issues by associating audio embedding with word embedding where labels are located in a vector space. This allowed the model to predict unseen labels in the training stage from music or retrieve music from any word query. This talk reviews the advance of music auto-tagging where research interests are moving toward combination with natural language processing techniques.

Anna Huang: Tuning Music Transformer

Music Transformer is an expressive language model for music, offering exciting potential for creative exploration. In the AI Song Contest, we see artists obtain a range of compelling results, by feeding it different musical fragments to elaborate. However, finding something novel and appropriate could take many iterations. If there's more control, then it could be possible to steer the exploration process. In this talk, I'll discuss preliminary work in taking both ML and HCI approaches to "tuning" Music Transformer towards users' creative goals, and also a common framework for evaluating progress in generative models and interfaces.

Table of Contents

Improving Real-time Score Following in Opera by Combining Music with Lyrics Tracking

Charles Brazier[1] **Gerhard Widmer**[1,2]
[1]Institute of Computational Perception, Johannes Kepler University Linz, Austria
[2]LIT AI Lab, Linz Institute of Technology, Austria
firstname.lastname@jku.at

Abstract

Fully automatic opera tracking is challenging because of the acoustic complexity of the genre, combining musical and linguistic information (singing, speech) in complex ways. In this paper, we propose a new pipeline for complete opera tracking. The pipeline is based on two trackers. A music tracker that has proven to be effective at tracking orchestral parts, will lead the tracking process. In addition, a lyrics tracker, that has recently been shown to reliably track the lyrics of opera songs, will correct the music tracker when tracking parts that have a text dominance over the music. We will demonstrate the efficiency of this method on the opera *Don Giovanni*, showing that this technique helps improving accuracy and robustness of a complete opera tracker.

1 Introduction and Contribution

Score following aims at aligning classical music performances with their corresponding scores (sheet music), in order to assign a score position at each time step in the performance. There has been constant progress in this domain, starting with the tracking of monophonic melodies in (Dannenberg, 1984), all the way to recent systems that can follow, under real conditions, complex orchestral works (Arzt and Widmer, 2015) in a completely autonomous process. This has led to the development of new applications such as automatic page-turning for pianists (Arzt et al., 2008), live performance visualization (Lartillot et al., 2020), or score viewing and automatic contextualization in orchestra concerts (Prockup et al., 2013; Arzt et al., 2015) to enrich the viewers' experience.

Tracking live opera performances would become an essential tool for all future opera halls, supporting functionalities like fully automatic subtitles display, or automatic camera control and video editing for live streaming services. However, and compared to previous existing works, operas are more challenging to track, due to the setup with a complete orchestra and singers that act and sing on stage, one or several at a time, for several hours, with various noises, acting breaks, intermittent applause, musical (sometimes improvised) interludes, etc.

First attempts at opera tracking (Brazier and Widmer, 2020b,a) use an *On-Line Dynamic Time Warping (OLTW)* algorithm (Dixon, 2005) to align complete performances with a *reference performance* (some other recording of the work in question) that has been aligned to the score beforehand and serves as a proxy to the score. This audio-to-audio alignment strategy is an elegant way to circumvent the unavailability of complete opera score files in symbolic format. Also, using a real recording is advantageous because the sounds in the reference are much more realistic and similar to what is to be expected in the real performance than anything one could synthesize from a score. Brazier and Widmer (2020b) combine alignment with three audio event detectors for music, speech/singing voice, and applause, which halt the tracking process during long silences, applause, or interlude passages that can occur in between the parts. Brazier and Widmer (2020a) further improve tracking accuracy by using two trackers working in parallel, one using audio features tuned on orchestral music (Gadermaier and Widmer, 2019), the other using features tuned on the *recitative* subset of one opera performance.

In this work, we propose to exploit an additional source of information: the *lyrics* sung or spoken in the audio recordings. We do not assume the written lyrics to be available in textual form. Rather, the idea is to train an acoustic phoneme recognition model that extracts phoneme sequence estimates both from the reference (off-line) and the live performance (on-line), and to align these in

Conductor	Place	Year	Duration	Role
H.v. Karajan	Berlin	1985	2:57:53	Reference
Á. Fischer	Vienna	2018	3:12:54	Target
A. Manacorda	Vienna	2019	3:07:09	Target

Table 1: Dataset used in this study.

real time, giving us a real-time lyrics tracking algorithm. More specifically, the acoustic model will predict, for each audio frame, a probability vector over a set of phonemes. For each part (aria, recitative, etc.) in the score, we assign a *voice on music ratio* value, calculated on the reference recording with the help of dedicated music/speech audio classifiers. The music tracker leads the alignment process. As soon as the score position corresponds to a voice-dominant part in the score, the lyrics tracker starts and we rely on its score position. When the score position reaches a music-dominant part, the lyrics tracker is stopped and the music tracker alone is used.

Acoustic model and lyrics tracker have already been presented in a recent publication (Brazier and Widmer, 2021b), but only evaluated on selected text-heavy *recitativo* passages. Here we demonstrate, for the first time, the benefit of combining lyrics with music tracking in an automated way.

2 Data Description

Score followers are evaluated by computing their alignment accuracy on audio performances that have been manually annotated to the corresponding score. As no such dataset exists for opera, we had to create our own. The dataset focuses on the opera *Don Giovanni* by W.A.Mozart. As the *reference*, serving as a proxy to the score, we selected a commercial CD recording conducted by Herbert von Karajan in 1985. As *target performances* that we want to align to the score in real time, we use two full live performances, with different casts and stagings, that have been recently recorded at and by the *Vienna State Opera*, one conducted by Ádam Fischer in 2018, the other by Antonello Manacorda in 2019. There are two parts in the reference that are not played in the two live recordings. For this study, we decided to remove these to align performances that follow the same score structure. Compared to the reference, the live performances contain applause, breaks, and interludes that can appear between parts. The dataset details are given in Table 1.

For each performance in the dataset, we manually affixed 5,304 bar annotations, 2,866 for the first act and 2,438 for the second, corresponding to the total number of bars present in the 500-pages score book. The annotations in the reference performance permit to link the complete performance to the score book. The annotations in the target performances serve for evaluating the alignment accuracy of our tracker. Thus, our dataset comprises more than 9 hours of opera recordings played and sung in real conditions by different orchestras and singers and recorded with different recording setups. It contains around 16,000 manual bar-level annotations assigned to the 530 pages score book, which is available online thanks to the *Mozarteum Foundation Salzburg*[1]. Precisely, annotating these 9 hours of music took about 300 hours of work.

3 Real-Time Opera Trackers

Operas are complex works that combine music, singing, and speech in complex ways. Most of the time, the piece is led by the music, with singers singing on top of the orchestra. However, operas also include passages, such as *recitativo* sections, where the dominant signal is the lyrics spoken or sung by the singers, with a sparse musical accompaniment that is played differently across performances (e.g., arpeggiated chords, not aligned to the lyrics, partly improvised, and played by different instruments). To tackle this, we propose to alternate between two trackers, one focusing on the music information and the other on the lyrics information. We first describe our *music tracker* that serves as a baseline in this study and that leads the tracking process. We then describe our *lyrics tracker*, and then propose one simple way of combining them for achieving a better global tracking accuracy. This combination strategy will be experimentally verified in the next chapter.

3.1 Music Tracker

The *music tracker* is based on an adaptive version of the On-Line Time Warping (OLTW) algorithm (Dixon, 2005) that has been successfully used in orchestra (Arzt et al., 2008) and also in opera tracking (Brazier and Widmer, 2020b,a). The OLTW algorithm updates an accumulated cost vector that has the length of the reference feature sequence, where the index of its minimal

[1]https://dme.mozarteum.at/DME/nma/

value corresponds to the score position given by the algorithm. Per audio frame, it receives as input a feature vector of 100 MFCCs (120 MFCCs are calculated from the audio sampled at 44.1 kHz, but the first 20 are discarded (Gadermaier and Widmer, 2019)), computed with a window size of 20 ms, and a hop size of 10 ms. The features of the reference audio are computed beforehand, while those of the target performance are computed in real-time. For each new incoming target feature, we compute the cosine distance between the feature and an interval of reference features of length c, centered around the expected score position (in practice c is fixed to 4000, corresponding to a context of 40 seconds of audio). Then, considering the previous score position sp, the previously accumulated cost vector D_{prev}, and the current distance vector d, we compute the value of the new accumulated cost vector D by first initializing its values by $+\infty$, and then applying the following recursive formula:

$$\forall i \in [sp - c/2 : sp_{j-1} + c/2] \,,$$

$$D[i] = d[i - (sp - c/2)] + \min \begin{cases} D_{prev}[i-1] \\ D_{prev}[i] \\ D[i-1] \end{cases} \tag{1}$$

To compare costs in D among themselves and not favor shorter paths over longer ones, we normalize them by dividing all values by their distance from the initial score position (i.e. by the sum of their index in the accumulated vector and an incremental counter representing the number of iterations since the beginning of the tracking).

Our target performances are performed under real conditions and thus include applause, breaks, or interludes that can be played in between the parts. We make use of the applause, music, and speech detectors detailed in (Brazier and Widmer, 2020a) to halt the tracking process when detected.

3.2 Lyrics Tracker

The *lyrics tracker* makes use of an on-line audio-to-lyrics alignment method that has been shown to robustly track the lyrics of different languages, in the genre of opera (Brazier and Widmer, 2021b). The tracker is composed of an acoustic model that generates, in real-time, *posteriograms* representing the frame-wise probability distribution over a set of predefined phonemes through time. Then, it employs the same OLTW algorithm described in Section 3.1, but in this case, aligning the posteriogram of the reference performance generated beforehand, and the posteriogram of the target performance generated online. This obviates the need for a text-to-phoneme tool to translate the written-out lyrics, as well as a manual alignment of the lyrics to the reference performance. It works without having the lyrics themselves and can track a language other than the language(s) the acoustic model was trained on, as shown in (Brazier and Widmer, 2021b).

The *acoustic model* is the core element of our lyrics tracker; its role is to estimate in real-time a posteriogram matrix from the audio recording. Its architecture is the CP-ResNet (Koutini et al., 2019), composed of convolutional layers with residual connections between layers, and has a receptive field of 57 frames in the input feature sequence centered around its time position, fixing the delay of the model to 28 frames. The model takes as input 80 MFCCs that are extracted from an audio window of 20 ms, sampled at 16 kHz, with a hop size of 10 ms; it outputs a vector every 40 ms. The output vector is of length 60, representing the classes of the 57 different phonemes that are included in the multilingual DALI dataset (Meseguer-Brocal et al., 2018) used to train the model. The dataset collects 275 hours of Western musical genres with lyrics annotations at the sentence, word, or note level, and includes English, German, French, Spanish and Italian languages. The phoneme representation permits to train a single model on different languages (Vaglio et al., 2020). The output vector also adds the space token, the instrumental token, and the blank token, essential to a Connectionist Temporal Classification (CTC) training (Graves et al., 2006) (the blank class will be ignored when applying Equation 1).

3.3 M&L Tracker: Combining Music and Lyrics Trackers

To exploit the complementarity between the two previously described trackers, we first classify each part of the opera in two classes (in the given reference performance): parts dominated by the music and parts dominated by the voice. To do so, we use the structure detailed in the Table of Contents of the Opera[2], and consider each title as an individual part. For each part, we use the mu-

[2] dme.mozarteum.at/DME/nma/nma_toc.php?vsep=68

sic and voice detectors (already used to halt the tracking process in between parts, as mentioned in Section 3.1 above) to calculate a *voice over music ratio* that is given by the percentage of voice along the part divided by the percentage of music. Thus, an instrumental part will have a ratio close to 0, whereas a part that contains more voice than music will have a ratio higher than 1.

The combination of the two proposed tracking models is delicate because they both work at a different pace (10ms for the music tracker, and 40ms for the lyrics tracker), the lyrics tracker has a delay of 280ms in its output due to its receptive field, and neither of them is able to track accurately full opera performances. More precisely, the music tracker is inaccurate when an improvised accompaniment is played during a part led by the lyrics, and the lyrics tracker is entirely lost during instrumental parts. Our approach is to use the music tracker continuously, along with the complete target performance. When the score position given by the music tracker corresponds to a part in the score that, according to our estimated voice/music ratio, is dominated by voice(s), we initialize the accumulated cost vector of the lyrics tracker by values of $+\infty$ everywhere, and a value of 0 at the score position given by the music tracker. We then use separately music and lyrics trackers but we rely only on the score position given by the lyrics tracker. As soon as the score position given by the lyrics tracker corresponds to a part in the score dominated by music, we stop the lyrics tracker and rely on the position given by the music tracker.

4 Experiments and Discussion

For our experiments, we compare three different tracking models. The first, *music*, reproduces the work in (Brazier and Widmer, 2020b) and uses the music tracker only (including acoustic event detectors to deal with interludes and other unexpected events such as applause and acting pauses). The second one, *musicP*, is the state-of-the-art opera tracker (Brazier and Widmer, 2020a); it uses two music trackers in parallel, one using the features detailed in Section 3.1, the other using optimized audio features that have been tuned on the recitative subset of the Fischer performance. Finally, the third tracker *M&L* is the contribution of this paper. The systems are evaluated by their alignment accuracies (Cont et al., 2007). We report the mean error in ms, as well as the proportions of bar boundaries (which reflect the precision of our ground truth annotations) that are detected with an error less than 1, 2, and 5 seconds. The results are given in Table 2.

Conductor	Tracker	Mean	≤ 1s	≤ 2s	≤ 5s
Fischer	music	811ms	91.8%	95.0%	97.3%
	musicP	373ms	93.4%	96.8%	99.0%
	M&L	**335ms**	**94.1%**	**97.3%**	**99.2%**
Manacorda	music	561ms	90.1%	94.5%	97.9%
	musicP	547ms	90.3%	94.7%	98.0%
	M&L	**410ms**	**91.6%**	**95.9%**	**99.0%**

Table 2: Tracking error of three trackers: music (Brazier and Widmer, 2020b), musicP (Brazier and Widmer, 2020a), and Music and Lyrics (M&L).

For both live target performances, the proposed *music & lyrics* tracker achieves the best accuracy, beating the *music* tracker, and also the *musicP* tracker whose features were tuned on the Fischer performance. The accuracy improvement on Fischer is relatively small, but no fine-tuning on features is done in our proposal. The improvements on Manacorda are more substantial, dropping the mean error to 410 ms and increasing all the 3 percentages by at least one point.

We tried to take into account the delay of the lyrics tracker, in adding an offset to the score position given by the tracker, but the best results were achieved in ignoring this delay.

5 Conclusion

We have presented a new state-of-the-art method for tracking full-length opera performances. The method makes use of an acoustic model that estimates the sung lyrics (phoneme probability vectors) over time. The final model combines lyrics and music information (without requiring the written lyrics as input) via two specific trackers. The combination helps to improve the tracking accuracy of the performance.

The proposed method requires a part segmentation of the reference performance. The beginnings and ends of each part are directly given by the manual bar annotations, useful to also handle structural mismatches in opera (Brazier and Widmer, 2021a). However, we plan to emancipate ourselves from the manual annotations with the development of a method that fully autonomously segments a piece.

Acknowledgments

The research is supported by the European Union under the EU's Horizon 2020 research and innovation programme, Marie Skłodowska-Curie grant agreement No. 765068 ("MIP-Frontiers"). The LIT AI Lab is supported by the Federal State of Upper Austria.

References

Andreas Arzt, Harald Frostel, Thassilo Gadermaier, Martin Gasser, Maarten Grachten, and Gerhard Widmer. 2015. Artificial Intelligence in the Concertgebouw. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2424–2430, Buenos Aires, Argentina.

Andreas Arzt and Gerhard Widmer. 2015. Real-Time Music Tracking Using Multiple Performances as a Reference. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 357–363, Málaga, Spain.

Andreas Arzt, Gerhard Widmer, and Simon Dixon. 2008. Automatic Page Turning for Musicians via Real-Time Machine Listening. In *Proc. of the European Conference on Artificial Intelligence (ECAI)*, pages 241–245, Patras, Greece.

Charles Brazier and Gerhard Widmer. 2020a. Addressing the Recitative Problem in Real-time Opera Tracking. In *Proc. of the Frontiers of Research in Speech and Music conference (FRSM)*, Silchar, India.

Charles Brazier and Gerhard Widmer. 2020b. Towards Reliable Real-Time Opera Tracking: Combining Alignment with Audio Event Detectors to Increase Robustness. In *Proc. of the Sound and Music Computing Conference (SMC)*, pages 371–377, Turin, Italy.

Charles Brazier and Gerhard Widmer. 2021a. Handling Structural Mismatches in Real-time Opera Tracking. In *Proc. of the European Signal Processing Conference (EUSIPCO)*, Dublin, Ireland.

Charles Brazier and Gerhard Widmer. 2021b. On-Line Audio-to-Lyrics Alignment Based on a Reference Performance. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Online.

Arshia Cont, Diemo Schwarz, Norbert Schnell, and Christopher Raphael. 2007. Evaluation of Real-Time Audio-to-Score Alignment. In *International Symp. on Music Information Retrieval (ISMIR)*, pages 315–316, Vienna, Austria.

Roger B Dannenberg. 1984. An On-Line Algorithm For Real-Time Accompaniment. In *Proc. of the International Computer Music Conference (ICMC)*, pages 193–198, Paris, France.

Simon Dixon. 2005. An On-Line Time Warping Algorithm for Tracking Musical Performances. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1727–1728, Edinburgh, Scotland, UK.

Thassilo Gadermaier and Gerhard Widmer. 2019. A Study of Annotation and Alignment Accuracy for Performance Comparison in Complex Orchestral Music. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 769–775, Delft, The Netherlands.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 369–376, Pittsburgh, Pennsylvania, USA.

Khaled Koutini, Hamid Eghbal-zadeh, Matthias Dorfer, and Gerhard Widmer. 2019. The Receptive Field as a Regularizer in Deep Convolutional Neural Networks for Acoustic Scene Classification. In *Proc. of the European Signal Processing Conference (EUSIPCO)*, A Coruña, Spain.

Olivier Lartillot, Carlos Cancino-Chacón, and Charles Brazier. 2020. Real-Time Visualisation of Fugue Played by a String Quartet. In *Proc. of the Sound and Music Computing Conference (SMC)*, pages 115–122, Turin, Italy.

Gabriel Meseguer-Brocal, Alice Cohen-Hadria, and Geoffroy Peeters. 2018. DALI: A Large Dataset of Synchronized Audio, Lyrics and Notes, Automatically Created using Teacher-Student Machine Learning Paradigm. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 431–437, Paris, France.

Matthew Prockup, David Grunberg, Alex Hrybyk, and Youngmoo E. Kim. 2013. Orchestral Performance Companion: Using Real-Time Audio to Score Alignment. *IEEE MultiMedia*, 20(2):52–60.

Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard, and Florence D'alché-Buc. 2020. Multilingual lyrics-to-audio alignment. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 512–519, Montréal, Canada.

What Musical Knowledge Does Self-Attention Learn ?

Mikaela Keller[1,2] **Gabriel Loiseau**[2] **Louis Bigo**[2]
[1] Inria
[2] Univ. Lille, CNRS, Centrale Lille
UMR 9189 CRIStAL, F-59000 Lille, France
{mikaela.keller,louis.bigo}@univ-lille.fr

Abstract

Since their conception for NLP tasks in 2017, Transformer neural networks have been increasingly used with compelling results for a variety of symbolic MIR tasks including music analysis, classification and generation. Although the concept of self-attention between words in text can intuitively be transposed as a relation between musical objects such as notes or chords in a score, it remains relatively unknown what kind of musical relations precisely tend to be captured by self attention mechanisms when applied to musical data. Moreover, the principle of self-attention has been elaborated in NLP to help model the "meaning" of a sentence while in the musical domain this concept appears to be more subjective. In this explorative work, we open the music transformer black box looking to identify which aspects of music are actually learnt by the self-attention mechanism. We apply this approach to two MIR probing tasks : composer classification and cadence identification.

1 Introduction

The Transformer (Vaswani et al., 2017) is a neural network architecture based on the self-attention mechanism that was designed for sequence prediction tasks (machine translation, syntactic parsing, etc.) in NLP. Subsequently, the self-attention principle has also been applied with success to improve MIR tasks including harmony analysis (Chen and Su, 2021) and generation with long-term coherence as demonstrated with Music Transformer (Huang et al., 2018b). The Music Transformer model has then inspired various researches including the generation of pop music (Huang and Yang, 2020) and guitar tablature (Chen et al., 2020).

Despite its increasing use in MIR tasks, the nature of the musical knowledge learned by Transformers is rarely studied. (Huang et al., 2018a) proposes a tool to visualise self-attention weights associated to a musical extract but without any systematic analysis. Inspired by NLP literature(Conneau et al., 2018; Coenen et al., 2019; Tenney et al., 2019; Manning et al., 2020) our work aims at opening the Music Transformer *black box* in order to extract its abstract representation of musical sequences and submit those representations to two selected MIR "probing" tasks : composer classification and cadence detection.

The self-attention mechanism is encoded within a transformer through matrices of coefficients, produced by *attention heads*, which are distributed in the subsequent layers of the network. Given a sequence of tokens $x_1, \ldots, x_T$ an attention head produces an attention matrix $A = (a_{ij})_{1 \leq i,j \leq T}$ where a_{ij} encodes "the attention that token x_i gives to token x_j" or the weight that x_j is going to play in in the next layer representation of x_i. The goal of our study[1] consists in identifying the musical knowledge that is encoded within these matrices in a trained Transformer. For this purpose we designed two "probing" datasets of musical sequences labeled with informations that were not explicitly available to the Transformer during training. The first dataset is labeled by the composer of the sequence. In the second dataset the sequences are characterized as containing a cadence (musical phrase ending) or not.

In the following we show, that a simple linear classifier fed with isolated attention matrices is able to discriminate between two composers when their styles are different enough. In contrast, an analogous experiment shows that marks of structural phenomena such as cadences appear more challenging to detect in attention matrices.

In the second part of our study, we examine attention values in order to gain insights into the

[1] Code avaliable at https://github.com/Music-NLP/MusicalSelfAttention

classification results. Our observations reveal various orientations (past or future) of attention spans among composers, as well as prominent attention values on theoretic cadence preparation points.

2 Attention Based Sequence Representation

In this work, the Music Transformer is used as a representation tool, to compute self-attention relations for any arbitrary musical sequence.

The MAESTRO dataset is used in this study to train the Music Transformer. This dataset gathers 1276 piano performances of pieces composed by 54 major composers of different styles, including Bach, Mozart, Beethoven or Debussy. In order to be compatible with the Transformer input format, the MAESTRO dataset is converted into sequences of tokens following the syntax proposed by (Huang et al., 2018b). This token representation includes `NOTE ON`, `NOTE OFF`, `TIME SHIFT`, and `VELOCITY` types. In this study, we trained[2] a Music Transformer neural network on this corpus as explained in (Huang et al., 2018b).

The Transformer architecture trained for this study includes an encoder with 6 layers, each composed of 4 attention heads. Given an input sequence of T elements an attention head produces a square real-valued attention matrix $X = (x_{ij})$ of dimension $T \times T$. The value x_{ij} is usually interpreted as the attention that the elements at position i has for the element at position j. Once the transformer is trained, it has the ability to systematically abstract any musical sequence of size T by a set of $6 \times 4 = 24$ attention matrices of size $T \times T$. Through probing tasks NLP literature (Tenney et al., 2019; Manning et al., 2020) has reported that lower attention heads seem to attend to lower level abstractions, such as syntactic parsing, while deeper layers attend to higher level abstract such as coreference resolution. Assuming that some of this knowledge is transferable to the musical domain we have chosen to focus on the deeper layer of the encoder for representing the sequences in our MIR inspired probing tasks. We have chosen to collapse the 4 attention matrices produced by the last layer into an average matrice, and to use these $T \times T$ coefficients as the input to the classification tasks that we define in the next section[3].

[2]Using the implementation in https://github.com/jason9693/MusicTransformer-tensorflow2.0

[3]Although probing tasks are often performed on other outputs of the transformer, limiting the transformation of attention values facilitates their musical interpretation in this work.

Figure 1 illustrates this pipeline.

3 Agnostic Probing Tasks

In this section, we describe two probing tasks that aim at highlighting the musical knowledge encoded in attention values computed by the Music Transformer. The first task is a composer classification and the second one is cadence detection. Both tasks are formulated as supervised binary classification performed on the attention matrices described in section 2.

3.1 Composer Identification

We evaluate the ability of learned attention representations to model musical style through a composer identification task.

We used a subset of the MAESTRO dataset that contains unique composer performances to create several binary classification tasks `composer1` vs `composer2`. To better highlight the ability of attention values to capture stylistic information, we deliberately selected composers that are known to be close in term of style, such as Haydn and Mozart, and far apart, such as Bach and Chopin.

For each couple, a set of training musical sequences of fixed size are abstractly represented as attention matrices (see Section 2). The training sets are balanced and contain 2648 sequences from each of the composers. The corresponding abstract representations are then given as input to a logistic regression classifier with $l2$-regularization that is trained to assign composer authorship to any input attention matrix. The experiment is repeated 5 times, sampling various training sets for every couple of composers and for various sizes of sequences. Figure 2 displays the average performance of the classifiers over a separate and fixed test set[4] of 426*2 sequences. A random classifier is here expected to have a 50% accuracy.

Low standard deviations, illustrated by vertical lines on each experiment, show that given a couple of composers the accuracy is quite stable with respect to the various training sets. Figure 2 also shows that the accuracy generally tends to increase with the size of the sequences (which was not obvious since when increasing the size of the sequence we increase quadratically the search space number of dimensions without increasing

[4]We used MAESTRO train/test split to insure that a same piece could not appear both in the train and the test set

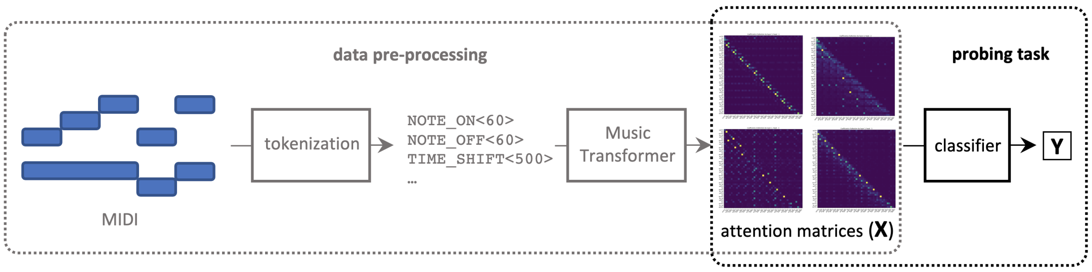


Figure 1: Pipeline used for the two probing tasks. The left part illustrates the systematic representation of a midi sequence into a set of self-attention values thanks to the Music Transformer. The right part illustrates how a probing task is formulated as a classification problem on attention values.

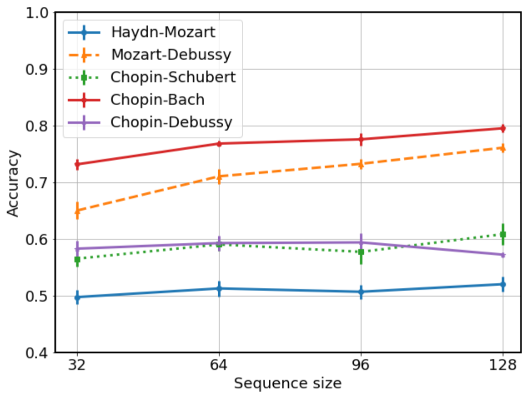


Figure 2: Mean accuracy of composer identification on attention matrices computed from sequences of various lengths.

the number of examples). The difficulty of the classification task of a pair of composers certainly relates to how they differ in style. Interestingly, by using birth date gaps as rough proxy for style differences, the accuracies appears to match the difficulty of the tasks[5].

3.2 Cadence Detection

Cadences are structural breaks widely used in the classical repertoire to emphasize the end of a musical phrase. Cadence are often associated with a closure feeling that resolves a tension region (Blombach, 1987). This concept therefore appears as a promising candidate to validate the principle of self-attention in music as the short past that precedes a cadence is supposed to be organized in close relation with the upcoming cadence. This short past is sometime referred to as the *preparation* of the cadence.

The present task consists in evaluating how much the attention values encode the presence of a cadence. Our hypothesis is that cadential points and preparation points should have important mutual attention one for each other if they appear concomitantly within the training set. Attention matrices are computed as explained in section 2 through a Transformer which is trained on the MAESTRO corpus. Given the pieces of music present in the MAESTRO dataset, it can reasonably hypothesized that cadences, that are typical of the classical era, are sufficiently represented in the training set to be modeled by the Transformer. Similarly to the composer identification task, a set of attention matrices, that represent musical sequences with and without cadences, are used to train a logistic regression classifier. For this purpose, we use a dataset of 24 fugues from J.-S. Bach with cadence annotation (Giraud et al., 2015). A set of 3864 sequences of 64 tokens is sampled from the fugue dataset, a third of which include a cadence[6] while the remaining do not include any cadence. We use a leave-one-piece-out strategy to evaluate the performance of the cadence classification and compare it to a random classification on each fold of the cross-validation. The micro-averaged F1 score of the cadence classifiers is 0.458 as compared to 0.315 for the random classifier. This results seems to suggest that attention values learned by the Transformer do encode some information about the notion of cadence.

3.3 Discussion

Cadences belong to high level elements of tonal musical language. Despite their unified closure meaning, they can be realized through a large vari-

[5]Birth date gaps (in years) : Chopin-Bach: 125, Debussy-Mozart: 106, Debussy-Chopin: 52, Mozart-Haydn: 24 and Chopin-Schubert: 13

[6]A same cadence can appear several time in our dataset but at different positions and necessarily in the 2nd half of the sequence in order to favor the inclusion of the *preparation* of the cadence within the sequence.

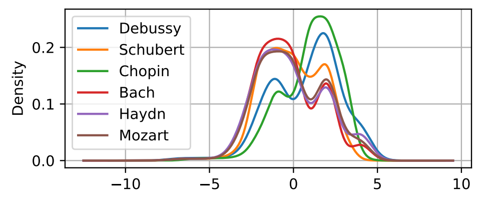


Figure 3: Distribution of the attention spans. The horizontal axis shows the length of attention spans (in tokens).

ety of musical surfaces, which makes their modeling particularly complex (Bigo et al., 2018). Musical style, on the other hand, can refer to lower level relationships between musical objects, like pitch intervals. It is therefore interesting to observe that our attention based classification approach give results better than chance both on style modelling and on cadence detection.

4 Musical Interpretation of Self-Attention Relations

In this section, we provide a few exploratory analysis to gain musical insights on the data that was given in input to the probing classifiers.

4.1 How Do Transformers Learn About Composers ?

As explained in section 2, the composer discrimination probing task was performed using an average attention matrix A_x computed from each sequence x. We averaged A_x for each composer over a subset of 1000 sequences used for training the linear classifiers. The sequence are of fixed size ($T = 64$). The result is a matrix $M = (m_{ij})_{1 \leq i,j \leq T}$ where m_{ij} is the average attention that the i^{th} token gives to the j^{th} token in the sequences of a given composer. We consider that a token at position i "looks at" a token in position j, *ie* it has an attention span of at least $i - j$, if the coefficient m_{ij} is greater than a certain threshold. In Figure 3 we report the distribution of attention spans for a threshold of 0.04 ($\approx 7\% - 10\%$ of coefficients) for several composers.

The figure shows that the learned attention span rarely exceeds five tokens in the past or in the future. Interestingly, the attention learned on early composers such as Bach, Haydn, Mozart, and Schubert seem to focuse *towards* tokens in the short past. In contrast, Chopin and Debussy attention is turned *towards* tokens in the short future, which might be partly related to a stylistic rupture of the composers with the classical era. Confirming this hypothesis would require a deeper study.

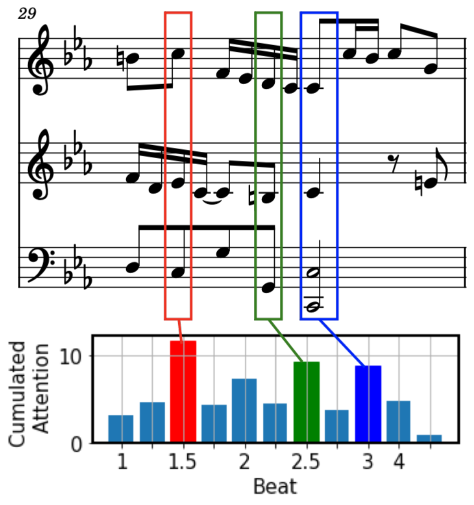


Figure 4: Cumulated attention on successive offsets of bar 29 of Fugue 2 of the *Well-Tempered Clavier* from Bach. A perfect authentic cadence is annotated on beat 3 (blue frame). Other points of prominent attention (red and green) correspond to important *preparation* points of the cadence.

4.2 How Do Transformers Learn About Cadences ?

In this experiment we observe the information within the attention matrix A_x of a sequence containing a cadence. The sequence can be divided into `TIME SHIFT` events that can be aligned with the beat pulse of the piece extract. Figure 4 shows the cumulated attention between `TIME SHIFT` events in regard with the sheet music.

5 Conclusions and Perspectives

We proposed in this work an original approach to improve our understanding of the musical knowledge that self-attention mechanism can learn. In spite of instructive results, these experiments highlight the difficulty to interpret neural values within a multi layer model but also confirm the necessity to pursue our efforts in that quest of comprehension of music deep learning models.

Futur works include experimenting with other probing tasks, such as harmony and tonality analysis, in order to better understand how Transformer architectures learn these high level concepts. It could also be interesting to test those tasks on different layers of the network to see if there is a gradation in the information levels of abstraction.

Acknowledgments

The authors are grateful to the Algomus and Magnet teams and to Kamil Akesbi for fruitful discussions. This work is supported by a special interdisciplinary funding (AIT) from the CRIStAL laboratory.

References

Louis Bigo, Laurent Feisthauer, Mathieu Giraud, and Florence Levé. 2018. Relevance of musical features for cadence detection. In *International Society for Music Information Retrieval Conference (ISMIR 2018)*.

Ann Blombach. 1987. Phrase and cadence: A study of terminology and definition. *Journal of Music Theory Pedagogy*, 1(2):225–251.

Tsung-Ping Chen and Li Su. 2021. Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models. *Transactions of the International Society for Music Information Retrieval*, 4(1).

Yu-Hua Chen, Yu-Hsiang Huang, Wen-Yi Hsiao, and Yi-Hsuan Yang. 2020. Automatic composition of guitar tabs by transformers and groove modeling. *arXiv preprint arXiv:2008.01431*.

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg. 2019. Visualizing and measuring the geometry of bert. *arXiv preprint arXiv:1906.02715*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. 2015. Computational fugue analysis. *Computer Music Journal*, 39(2):77–96.

Anna Huang, Monica Dinculescu, Ashish Vaswani, and Douglas Eck. 2018a. Visualizing music self-attention.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. 2018b. Music transformer. *arXiv preprint arXiv:1809.04281*.

Yu-Siang Huang and Yi-Hsuan Yang. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1180–1188.

Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Lyrics and Vocal Melody Generation conditioned on Accompaniment

Thomas Melistas
School of ECE
National Technical University of Athens
Greece
melistas.th@gmail.com

Theodoros Giannakopoulos
NCSR Demokritos
Greece
tyiannak@gmail.com

Georgios Paraskevopoulos
School of ECE
National Technical University of Athens
Greece
geopar@central.ntua.gr

Abstract

In this paper we present a previously unexplored task, the generation of lyrics and vocal melody for a given instrumental music piece in the symbolic domain. We model the above as a sequence-to-sequence task, using a memory efficient Transformer architecture, which we train on text event sequences that describe entire songs. Towards this end, we build a suitable dataset and apply musical analysis, compressing the instrumental part and making it key-independent. We further design a novel architecture to decouple lyrics and melody generation, making it possible to use pretrained language models and conditioning on lyrics. Finally, Mellotron is used to turn the generated sequences into singing audio.

1 Introduction

A significant part of research on singing has focused on information retrieval tasks, such as lyrics or melody transcription (Stoller et al., 2019; Nishikimi et al., 2019), as well as on singing voice synthesis (Nishimura et al., 2016). Generating the (symbolic) content of singing, namely lyrics and vocal melody, has only recently started gaining more attention. Relevant work has focused on generating lyrics for a specific music style or melody and lyrics-conditioned vocal melody generation.

Vocal music coexists with instrumental in most contemporary genres. However, despite the growing interest on studying the relation of lyrics and vocal melody, the connection of both to the accompaniment remains overlooked. In this work we aspire to fill this substantial gap.

We model the instrumental-conditioned generation of vocal melody and lyrics as a seq2seq task. First, we create pairs of text event sequences, which we use to train a baseline encoder-decoder Transformer architecture. We then propose a way to decouple lyrics from vocal melody generation, by inserting another decoder. Finally, we bring this symbolic output into the audio domain and perform a subjective evaluation study, using a singing voice synthesis model. In contrast to previous works that study vocal generation on the sentence level, we model full songs, which presents additional technical challenges.

Our main contributions to the field are the following: **(a)** We introduce the task of lyrics and vocal melody generation conditioned on the accompaniment. **(b)** We build a suitable dataset for this task by enforcing consistent tokenization. We apply musical analysis to compress the instrumental part up to 20% of the original, resulting to faster training. **(c)** We optimize the Transformer architecture in order to model full song sequences of up to 60k tokens in a single GPU. **(d)** We propose an architecture that decouples lyrics and vocal melody generation, providing the ability to use pretrained language models and predefined lyrics.

We release all code, datasets and some generated samples[1].

2 Related Work

Conditional Vocal Melody Generation In (Madhumani et al., 2020) a combination of word and syllable embeddings is used as input to an LSTM encoder (Hochreiter and Schmidhuber, 1997) that uses a vector to attend to three separate decoders, for note, duration and rest. Yu et al. (2021) use an LSTM that takes as input lyrics embeddings and noise vectors to sample MIDI sequences, which are provided to another LSTM alongside text embeddings and classified as real or fake. Another approach (Liu et al., 2020) studies singing voice generation without any melody or lyrics information, using GANs (Goodfellow et al., 2014) conditioned also on accompaniment.

[1]github.com/gulnazaki/lyrics-melody

Lyrics Generation In (Vechtomova et al., 2020) an LSTM-VAE creates latent representations of lyrics that condition lyrics generation on audio embeddings. Lu et al. (2019) use an encoder-decoder LSTM to generate lyrics, taking into account the only rhythmic quality of the melody. In (Watanabe et al., 2018) conditioning is done by concatenating each syllable to a local window of the corresponding melody note, before feeding it as input to an LSTM language model.

3 Dataset Description

Our dataset is built upon a subset of the Lakh MIDI Dataset (LMD) (Raffel, 2016) that consists of $45,129$ uniquely matched MIDI files.

3.1 Creating A More Consistent Dataset

LMD is not oriented towards the analysis of vocals. Therefore, many files do not include a vocal melody or synchronized lyrics. Moreover, the annotation of lyrics is not always consistent. Many tracks include different sentence and verse separators, mixing of MIDI lyrics and metadata, as well as inconsistent division of lyrics into sung syllables. The latter depends not only on the annotator but on the way the lyrics are sung, resulting, among others, to irregular tokenization of words.

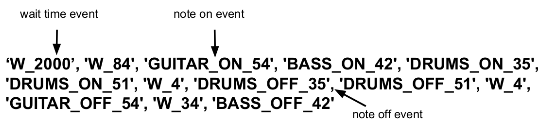
In order to formalize our dataset, we construct a pre-processing pipeline. We keep only English lyrics, remove any metadata and use standard sentence and verse separators. To derive the vocal part, we assign each lyric to the closest note and choose the track with the most matches. To make the division of lyrics consistent and reversible at inference time we enforce a strict syllabified format, using Phonetisaurus (Novak et al., 2015) for grapheme-to-phoneme conversion. We split words into syllables, each one ending at a vowel. If a note corresponds to $n > 1$ syllables we divide it to n equal duration notes and if a syllable spans $n > 1$ notes we match it to the first one and assign the next $n - 1$ notes to a special symbol.

After completing the above process we are left with 8505 valid MIDI tracks.

3.2 Text Event Format

We create separate text event sequences for the instrumental and the vocal parts.

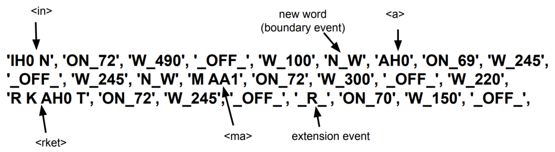
All sequences consist of the following types of tokens: (1) *Note on* (a note of this pitch starts), (2) *Note off* (a note ends), (3) *Wait time* (time passed in MIDI ticks[2]).



```
'W_2000', 'W_84', 'GUITAR_ON_54', 'BASS_ON_42', 'DRUMS_ON_35',
'DRUMS_ON_51', 'W_4', 'DRUMS_OFF_35', 'DRUMS_OFF_51', 'W_4',
'GUITAR_OFF_54', 'W_34', 'BASS_OFF_42'
```

Figure 1: Instrumental Text Event Representation

We restrict notes in the piano range, shifting octaves if needed, resulting in 88 MIDI pitches. All instruments are grouped into 8 classes (*Guitar, Bass, etc.*) and each name is appended to the corresponding note token. An instrumental sequence can be seen in Figure 1.



```
'IH0 N', 'ON_72', 'W_490', '_OFF_', 'W_100', 'N_W', 'AH0', 'ON_69', 'W_245',
'_OFF_', 'W_245', 'N_W', 'M AA1', 'ON_72', 'W_300', '_OFF_', 'W_220',
'R K AH0 T', 'ON_72', 'W_245', '_OFF_', '_R_', 'ON_70', 'W_150', '_OFF_',
```

Figure 2: Vocal Text Event Representation with phonemes corresponding to the lyrics: *in a marke(-e)t*

Vocal sequences contain additional tokens: (1) *Syllable/phoneme* (syllable of the following note), (2) *Extension* (following note extends previous syllable), (3) *Boundary* (comma, word, line or verse separators) events.

For lyrics we use the extracted phonemes to reduce the vocabulary and account for rhyming and homophones. Figure 2 shows a vocal melody sequence with phonemes preceding each note.

3.3 Chord Reduction

The above representation results in very long sequences when applied to full instrumental tracks, imposing infeasible memory requirements and slow training. A more compact representation can benefit both model performance and robustness.

Vocal scores in jazz or orchestral music commonly use a reduced representation of the accompaniment that informs the singer about the harmonic and rhythmical structure of a song. Inspired by this, we create a chord reduction of the instrumental, using the *music21* library (Cuthbert and Ariza, 2010). Individual instruments are merged and every new note results in the formation of a

[2]Takes values in $[1, 2000]$ (more tokens needed for larger durations). Each tick corresponds to $\frac{60}{TR}$ seconds, T being tempo in *Beats Per Minute* and *R* being the file resolution in *Pulses Per Quarter Note*.

new chord. This method achieves a substantial reduction factor of $\frac{1}{5}$, as shown in Table 1.

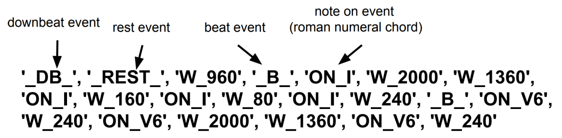
	Vocal	Instrumental	Reduced
median	1645	13041	3220
max	6115	59120	11730

Table 1: Number of tokens for Vocal, Instrumental and Reduced Instrumental Sequences

We decide not to include the percussion part and instead use notions such as *beats* and *downbeats* events, which give more abstract but concrete rhythmical information. Besides the significant reduction in the instrumental sequences, we avoid inserting noise to the process, since drum parts are interpreted as pitches by *music21*.

3.4 Roman Numeral Analysis

To further capitalize on this chord representation, we employ a type of musical analysis, called Roman Numeral Analysis. Its core idea is that chords and notes can be represented by a degree of the musical scale they belong in[3]. Usage of roman numerals reduces the token vocabulary and enables us to model the relative position of chords, essentially performing data augmentation, since all songs are transposed to a common but abstract key.



```
'_DB_', '_REST_', 'W_960', '_B_', 'ON_I', 'W_2000', 'W_1360',
'ON_I', 'W_160', 'ON_I', 'W_80', 'ON_I', 'W_240', '_B_', 'ON_V6',
'W_240', 'ON_V6', 'W_2000', 'W_1360', 'ON_V6', 'W_240'
```

Figure 3: Instrumental Text Event Representation with Roman Numeral Chords, Rests, Downbeats and Beats

We get a key estimation using the Krumhansl-Schmuckler algorithm (Temperley, 1999) and based on that we represent each chord as a roman numeral. An example of this representation for instrumental sequences can be seen in Figure 3[4].

We apply a similar procedure for the vocal melody, by converting each note to a scale degree. Since pitch information is important to get more expressive vocal performances, we complement each token with its octave number.

[3]Uppercase roman numerals are used to represent major chords, while lowercase represent minor ones. Numbers denote inversions and extra chord notes.

[4]Since the formed chords mostly succeed each other directly, we do not need *note off* events and instead use *rest* events when required.

4 Proposed Method

4.1 Encoder-Decoder Architecture

We optimize a Transformer architecture to model long sequences and use it as our baseline to generate vocal sequences, given the instrumental.

A drawback of the Transformer architecture is that the memory footprint of the dot-product attention mechanism scales quadratically with the sequence length. To avoid this, we use the Performer (Choromanski et al., 2020), an architecture that achieves linear space and time complexity by using a mechanism called *FAVOR+*, which makes an unbiased linear estimation of full-rank softmax attention.

We further use reversible layers (Gomez et al., 2017), storing the activations of only the last layer, and feed-forward chunking as showcased in (Kitaev et al., 2020). We perform layer normalization before each sublayer (pre-norm) (Chen et al., 2018), reporting more stable training, with no need to do warm-up. Finally, we use learnable positional embeddings and tie the token embeddings of the decoder (Press and Wolf, 2017).

4.2 Decoupled Architecture

We augment the above architecture by adding a separate decoder for lyrics. We use the encodings of its last layer to further condition the vocal melody decoder, adding a second cross-attention layer to it (Libovický et al., 2018). The architecture is presented in detail in Figure 4.

It should be noted that in this architecture lyrics are generated without any instrumental conditioning. While we experimented with models that also use cross-attention in the lyrics decoder, we found them to perform poorly and be much harder to train in comparison to this simplified version.

We train our model to minimize the sum of the cross-entropy losses of the lyrics text and the vocal sequence (without phonemes). The lyrics decoder can use its own tokenization and the two sequences are merged at inference time, following the syllable tokenization process of Subsection 3.1. This decoupling allows us to predefine lyrics and use prior knowledge, which we achieve by using a language model pretrained on English lyrics.

To this end, we use a distilled[5] version of GPT-2 (Radford et al., 2019) that fits our low-resource requirements. We load the model's weights into an

[5]https://huggingface.co/distilgpt2

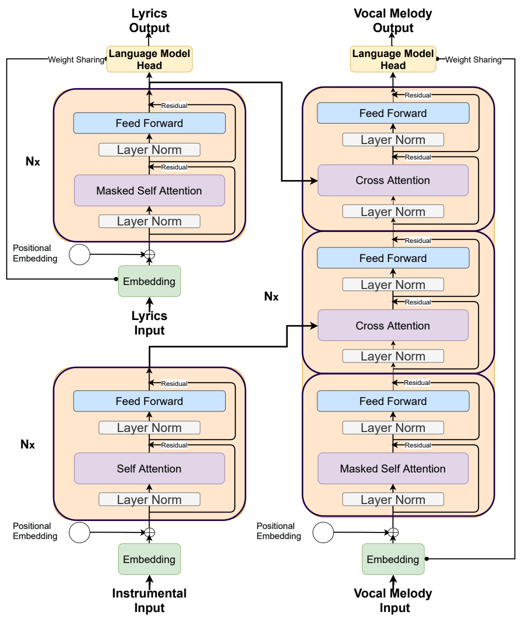


Figure 4: The Decoupled Architecture includes an instrumental encoder (bottom left), one decoder for lyrics (top left) and one decoder for vocal melody (right) with two cross-attention sublayers. The vocal melody decoder is conditioned by both instrumental and lyrics.

unconditional and causal decoder with *FAVOR+* attention, which we then fine-tune on a dataset of $263,666$ complete song lyrics. The purpose of this is twofold. As discussed in (Choromanski et al., 2020), fine-tuning is necessary for a Performer model to utilize the weights of a pretrained Transformer. Moreover, it helps our model adapt to the style of lyrics text (verse-chorus structure, repetition of words, onomatopoeia, etc.).

4.3 Experimental setup

We train **(a)** an encoder-decoder model on the full instrumental representation of Subsection 3.2 (*Vanilla Full - model A*), **(b)** an encoder-decoder on the reduced representation of Subsections 3.3, 3.4 (*Vanilla - model B*) and **(c)** a decoupled model on the latter representation (*Decoupled - model C*).

We use 6 layers with inner dimension of 512 and 8 attention heads for all models. The *AdamW* optimizer (Loshchilov and Hutter, 2019) is used with 0.001 learning rate and 0.1 weight decay.

During generation we use top-k sampling (Fan et al., 2018) keeping the top 0.1 tokens. We also take advantage of structural constrains in our representation (e.g. *note on* followed by *wait time* events only), by masking the valid tokens during inference.

We report that training for 6 epochs with batch size 8 takes a total of $\mathbf{74.6}$, $\mathbf{26.9}$ and $\mathbf{49.2}$ hours in an *NVIDIA Tesla T4 GPU* for models A, B and C respectively.

4.4 Subjective Evaluation

For 5 random instrumental tracks in the test set, we convert the outputs of our three models to audio, using a singing voice synthesis model called Mellotron (Valle et al., 2020). We then mix it with the synthesized instrumental. We ask 28 participant to choose between these samples on the basis of: **(a)** Rhythmic/Melodic Quality: how musical or interesting the vocal part is, **(b)** Relation to the Music: how well the vocal part fits with the instrumental, **(c)** Lyrical Content: quality of the generated lyrics. Table 2 shows the mean model preference values for all three objectives.

	Melody	Relation	Lyrics
Vanilla Full	0.369	0.246*	0.070
Vanilla	0.257*	0.374	0.052
Decoupled	**0.374**	**0.380**	**0.878***

Table 2: Mean value of each model preference for all 28 users. * denotes statistical significance ($p < 0.05$) using one-tail paired t-test (pairwise)

We observe that the reduced representation favors the instrumental-vocal relation, since it is more compact, but produces less interesting melodies. The decoupled model performs well on both metrics, which can be attributed to the separate modeling of the sequences, but does not reflect the independence between instrumental and lyrics. Finally, the lyrics generated by the decoupled model are significantly superior, which can be linked to the usage of a pretrained language model.

Using objective metrics to better understand the performance of these models remains to be done.

5 Conclusions

In this paper, we presented a pipeline to generate lyrics and vocal melody for any given instrumental MIDI file, using Transformer-based models. To this end, we have built and released a dataset oriented towards generation and study of vocals. We report that compressing the instrumental representation leads to substantially faster training and favors its connection to the vocal part. We further propose a decoupled architecture that allows us to use prior knowledge from language models and therefore generate more convincing lyrics.

References

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia. Association for Computational Linguistics.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2020. Rethinking attention with performers. *CoRR*, abs/2009.14794.

Michael Scott Cuthbert and Christopher Ariza. 2010. Music21: A toolkit for computer-aided musicology and symbolic music data. In *ISMIR*, pages 637–642. International Society for Music Information Retrieval.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Aidan N. Gomez, Mengye Ren, Raquel Urtasun, and Roger B. Grosse. 2017. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2214–2224.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer.

Jindřich Libovický, Jindřich Helcl, and David Mareček. 2018. Input combination strategies for multi-source transformer decoder. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 253–260, Brussels, Belgium. Association for Computational Linguistics.

Jen-Yu Liu, Yu-Hua Chen, Yin-Cheng Yeh, and Yi-Hsuan Yang. 2020. Score and lyrics-free singing voice generation. In *Proceedings of the Eleventh International Conference on Computational Creativity, ICCC 2020, Coimbra, Portugal, September 7-11, 2020*, pages 196–203. Association for Computational Creativity (ACC).

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Xu Lu, Jie Wang, Bojin Zhuang, Shaojun Wang, and Jing Xiao. 2019. A syllable-structured, contextually-based conditionally generation of chinese lyrics. In *PRICAI 2019: Trends in Artificial Intelligence - 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26-30, 2019, Proceedings, Part III*, volume 11672 of *Lecture Notes in Computer Science*, pages 257–265. Springer.

Gurunath Reddy Madhumani, Yi Yu, Florian Harscoët, Simon Canales, and Suhua Tang. 2020. Automatic neural lyrics and melody composition. *CoRR*, abs/2011.06380.

Ryo Nishikimi, Eita Nakamura, Satoru Fukayama, Masataka Goto, and Kazuyoshi Yoshii. 2019. Automatic singing transcription based on encoder-decoder recurrent neural networks with a weakly-supervised attention mechanism. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 161–165. IEEE.

Masanari Nishimura, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda. 2016. Singing voice synthesis based on deep neural networks. In *Interspeech*, pages 2478–2482.

Josef Novak, Nobuaki Minematsu, and Keikichi Hirose. 2015. Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the wfst framework. *Natural Language Engineering*, - 1:1–32.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Open AI Blog, 1(8)*.

Colin Raffel. 2016. Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching. PhD Thesis.

Daniel Stoller, Simon Durand, and Sebastian Ewert. 2019. End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model. In *IEEE International Conference on*

Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019, pages 181–185. IEEE.

David Temperley. 1999. What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered. *Music Perception: An Interdisciplinary Journal*, 17(1):65–100.

Rafael Valle, Jason Li, Ryan Prenger, and Bryan Catanzaro. 2020. Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 6189–6193. IEEE.

Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. 2020. Generation of lyrics lines conditioned on music audio clips. In *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, pages 33–37, Online. Association for Computational Linguistics.

Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. 2018. A melody-conditioned lyrics language model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 163–172, New Orleans, Louisiana. Association for Computational Linguistics.

Yi Yu, Abhishek Srivastava, and Simon Canales. 2021. Conditional lstm-gan for melody generation from lyrics. *ACM Trans. Multimedia Comput. Commun. Appl.*, 17(1).

Phoneme-Informed Note Segmentation of Monophonic Vocal Music

Yukun Li [*] **Emir Demirel** [†] **Polina Proutskova** **Simon Dixon**
Centre for Digital Music,
Queen Mary University of London, UK
yukun.li@qmul.ac.uk

Abstract

Note segmentation of vocal pitch tracks is an inherently difficult problem, on which human judgments often disagree. We propose a novel note segmentation method that leverages phonemic information. Phonemes and pitch tracks are automatically extracted and jointly utilised to estimate note transition regions. Note onsets are determined within these regions using an onset detection function. Finally, an HMM-based note tracker adds further note boundaries for the case where multiple notes are sung on the same vowel. Our note segmentation method outperforms the previous best method on a standard public test set, and is shown to be somewhat robust against different types of lyrical content. Because its performance is less convincing on another dataset, we analyse problem cases and suggest possible confounding issues.

1 Introduction

Automatic music transcription refers to converting an acoustic waveform into a symbolic representation. While monophonic instrument transcription is often considered to be a solved problem in music information retrieval (Benetos et al., 2013), this is not the case for singing, where pitch is rarely stable (Dai and Dixon, 2019).

A singing transcription system usually consists of two main steps: pitch tracking and note segmentation. Firstly, the pitch and voicing are estimated at each time point in the audio; secondly, the continuous pitch track is segmented into notes which have onset, offset and an indicative pitch. For the first step, we use the PYIN algorithm (Mauch and Dixon, 2014), which improves on the widely used YIN algorithm (de Cheveigné and Kawahara, 2002) for estimating the fundamental frequency and voicing (presence or absence of pitch) of a monophonic signal. As PYIN works well for monophonic pitch estimation, we only focus on note segmentation in this paper.

Despite the high level of research activity in this area, the average F-measures of note-level transcription metrics (Correct Onset, Pitch and Offset, COnPOff (Molina et al., 2014)) obtained by state-of-the-art systems are all lower than 60%. Detection of “soft” onsets and offsets is still an unsolved problem in note segmentation. Soft onsets and offsets occur when adjacent notes are smoothly connected without obvious loudness variations. In most cases, however, there is a phonetic change between notes. Various spectral features have been used to detect timbre changes, either by selecting as boundaries peaks above a threshold in the measure of timbre change (Gómez and Bonada, 2013; Yang et al., 2017), or by modelling vowels and their transitions using an HMM (Hsuan-Huei Shih et al., 2002; Heo and Lee, 2017). More recently, utilising the flexibility of deep neural networks, Fu and Su (2019) augmented their input data with onset- and offset-related features to improve note segmentation and transcription performance.

To solve the problem of soft onsets and offsets, this paper investigates whether phonemes extracted by a state-of-the-art automatic lyrics transcription system (Demirel et al., 2020) can make a positive contribution. We hypothesise that phoneme information can be used to narrow down the range of frames where onsets and offsets are likely to occur. In particular, consonants are possible indicators of note boundaries, whereas vowels, unless there is a significant change of pitch or of the vowel, indicate the body of a note.

2 Method

Based on the annotation approach of Molina et al. (2014), our method assumes that note boundaries

[*]YL is supported by a China Scholarship Council and Queen Mary University of London joint Ph.D. Scholarship.

[†]ED receives funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068.

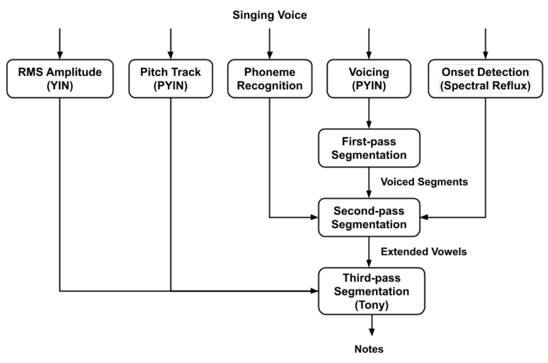


Figure 1: Proposed 3-step note segmentation method.

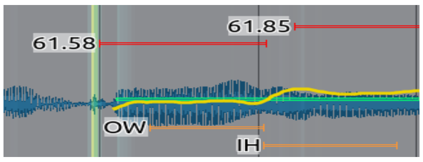
can be categorised into three types: (1) the beginnings and ends of voiced segments; (2) phonetic changes; (3) pitch[1] and amplitude changes. We detect these types of note boundaries and segment the vocal track in a three-step cascading approach which produces successively finer segmentations at each step (Figure 1).

In Step 1, voiced segments (segments of continuous pitch activity) are determined, based on the PYIN pitch track. In Step 2, the voiced segments are further segmented based on phonetic change, to create what we call *extended vowel* regions, as described in Section 2.1. In Step 3, extended vowel segments are further divided based on pitch and amplitude changes given by the PYIN algorithm. The main novelty of this approach is the incorporation of phonetic information into an existing framework for note segmentation, through the introduction of the second step, which we now describe in detail. [2]

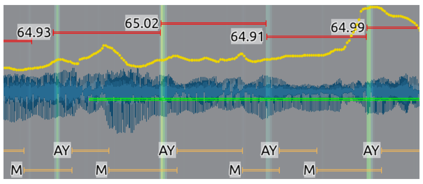
2.1 Step 2: Phoneme-Informed Segmentation

In order to detect phonetic change, the phonemes are automatically transcribed and temporally aligned using a state of the art lyrics transcription system (Demirel et al., 2020). The Spectral Reflux onset detection function (Sapp, 2006) is then used to estimate the note boundaries more precisely.

Demirel et al.'s system provides a transcribed phoneme sequence with aligned timings, but it claims a boundary accuracy tolerance of 50 ms. To detect note boundaries more precisely, we fine-tune the phonetic output with a simple additional signal processing step. First, we categorise the phonemes into vowels and consonants, determining the inter-vowel regions. We expand the inter-vowel regions by 50 ms each side to account for the system's tolerance. Finally, the maximum of spectral reflux in the expanded inter-vowel region determines the exact note boundary. The pitch of each segment is calculated as the median of the pitch track within the segment.



(a) Adjacent vowels with similar pitches erroneously merged into a single note.



(b) Successive notes sung on similar pitches with voiced phonemes between vowels, leading to multiple merge errors.

Figure 2: Examples of common errors made by the Tony software (Mauch et al., 2015). The ground truth segmentation (red) is labelled with median pitch in semitones (MIDI).[3] The PYIN pitch track is yellow, the note region extracted by Tony is bright green, detected phoneme boundaries are orange, and spectral flux is represented by the brightness of vertical lines.

Figure 2 illustrates the need for this step, showing examples where Tony, a benchmark method for monophonic singing voice transcription (Mauch et al., 2015), makes the systematic error of under-segmentation of successive notes having continuous steady pitch tracks during note transitions. These instances occur when consecutive notes are sung either with no consonants or silent gaps (breathing, articulation, etc.), or with short voiced consonants, between the successive vowels. When there are two adjacent vowels with no gap in between (Figure 2a), the note boundary is determined by the timing of the vowel transition. Where there is a gap between consecutive vowels (Figure 2b), we determine the note boundary as the location of the local maximum of the spectral reflux between the vowels in question.

2.2 Step 3: Pitch and Amplitude Changes

Steps 1 and 2 detect inter-vowel note boundaries, but there are also note boundaries within vowels that are communicated via pitch and amplitude changes. In such cases, phoneme-based segmen-

[1]We follow PYIN (Mauch et al., 2015) in setting the threshold of pitch change required for a note boundary to $\frac{2}{3}$ of a semitone.

[2]Step 1 is simple and does not require further description.

[3]We use non-integer values to represent continuous pitch.

tation is unable to determine the note boundaries. To estimate the timings of such boundaries, we apply the HMM-based segmentation method of Tony (Mauch et al., 2015) within the extended vowel regions resulting from steps 1 and 2 (Fig. 2).

3 Evaluation

For evaluation, we use the framework proposed by Molina et al. (2014), including their monophonic singing dataset (38 recordings with total duration 1154 seconds). We first show results from an ablation study using standard metrics on this dataset, and then follow this up with a comparison with recently published systems. We then examine the effect of linguistic properties of the data on segmentation results, and test on another publicly available dataset (Dai et al., 2015).

3.1 Ablation Experiments

To illustrate the contribution of each step in our approach to the overall performance, we report results for different combinations of the steps described in Section 2. Table 1 lists the methods, the features used, and their performance on three metrics. Since we consider voicing analysis (Step 1) as fundamental to any singing segmentation approach, we always include this feature, and test different combinations of Steps 2 and 3.

The first three evaluation metrics (columns) in Table 1 are the F-measures of COnPOff, COnP and COn, as used in MIREX, and the other three are the count proportions for various types of segmentation errors (Molina et al., 2014). COnPOff measures the rate of transcribed notes with correct onset ($\pm$50 ms), pitch ($\pm$0.5 semitones) and offset ($\pm$50 ms or $\pm$20% of the duration of the reference note). COnP represents correct onset and pitch, and COn evaluates correct onset only. A “Split” error means the ground truth note is split into multiple notes in the transcription, while a “Merged” error is the opposite. A “Spurious” note error occurs when a transcribed note does not overlap in time with any ground truth note. The results indicate that the various components of our approach each contribute positively to the overall performance on all three note-level metrics. In particular, disabling either Step 2 or 3 reduces performance by $\sim$9% on the strictest measure, with Step 2 making the greater contribution to the results.

In addition, for all versions of the system, relaxing the requirement of correct offset detection results in 15–20% better results, whereas relaxing the requirement to estimate the correct pitch only contributes a further 5% to the results. The remaining errors (relating to the onset) account for 20–25% of the results, so it is clear that a high proportion of errors relate to onsets and offsets, or in other words, the segmentation.

3.2 Comparison to the State-of-the-Art

In Table 2 we compare our results to published work on singing transcription. We tested our three-step method on Molina et al.'s dataset[4] (Molina et al., 2014), and compared its performance with six of the previous best sung note segmentation and transcription methods.

Overall, the results demonstrate that our proposed method achieves the best overall performance (F-measure), by a small margin over Fu and Su's recent work (Fu and Su, 2019). In addition, we have the lowest rates of merged and spurious note errors, and only on the split error metric are our results inferior to other systems. This means that the system has a tendency to over-segment the sung notes, compared to other published work. Looking more closely at the results, however, we see that most of the systems with lower split errors have very high rates of merged errors, so they are in fact under-segmenting the signal.

3.3 The Effect of Language

In this subsection, we investigate the robustness of the proposed system to various types of lyric content, including different languages and non-linguistic content. In addition, we discuss the sources of errors made by our system. We categorised the dataset by Molina et al. (2014) into the five groups represented by the columns of Table 3. Melodies in this dataset are sung either in English, Spanish and/or the following isolated syllables: /Na/, /Da/ and /La/. Using the F-measure of COnPOff, we compare performance of three versions of our system.

Several surprising results appear in Table 3. Starting with the complete system (the final row), the results for Spanish are about 19% higher than those for English. It is not entirely unexpected that Spanish is easier to segment, but this should be weighed against the fact that the phoneme predictions come from a lyrics transcriber that is trained on English language songs (Demirel et al., 2020).

[4]For methodological correctness, we exclude 3 samples during evaluation which had been used during analysis and development, even though we did not tune any hyperparameters on these samples. The results do not change substantially between the two versions of the dataset.

Methods	Features Used	COnPOff	COnP	COn	Split	Merged	Spurious
Steps 1+2	voicing(1), phoneme(2), onset(2)	0.525	0.712	0.761	**0.013**	0.235	0.128
Steps 1+3	voicing(1), pitch(3), amplitude(3)	0.520	0.683	0.741	0.079	0.233	0.114
Steps 1+2+3	voicing(1), phoneme(2), onset(2), pitch(3), amplitude(3)	**0.610**	**0.762**	**0.807**	0.093	**0.078**	**0.035**

Table 1: Transcription performance (F-measure) on the Molina dataset for three versions of our approach. Columns represent correct (C) onset (On), pitch (P) and/or offset (Off), respectively, and three types of error (see Sec. 3.1).

Method	Precision	Recall	F-measure	Split	Merged	Spurious
Ryynänen & Klapuri (Ryynänen and Klapuri, 2004)	0.304	0.315	0.308	0.105	0.248	0.116
Gómez & Bonada (Gómez and Bonada, 2013)	0.430	0.373	0.398	0.140	0.167	0.071
Molina et al. (SiPTH) (Molina et al., 2015)	0.397	0.440	0.415	0.074	0.309	0.157
Yang et al. (Yang et al., 2017)	0.409	0.436	0.421	0.064	0.230	0.120
Mauch et al. (Tony) (Mauch et al., 2015)	0.510	0.534	0.520	0.079	0.230	0.112
Fu and Su (Fu and Su, 2019)	0.625	0.569	0.594	**0.048**	0.080	0.044
Steps 1+2+3 (whole dataset)	**0.626**	**0.597**	**0.610**	0.093	**0.078**	**0.035**
Steps 1+2+3 (test set)	0.634	0.606	0.618	0.090	0.080	0.035

Table 2: Transcription and segmentation performance on the whole dataset of Molina et al. (2014), compared with published results (best results in bold). The first three rows are reported by Molina et al. (2015), and the following two are quoted from Yang et al. (2017). The final row compares performance evaluated on the smaller test set. The first three columns refer to COnPOff (correct pitch, onset and offset) results; the remaining columns are segmentation error types (see Sec. 3.1).

	English	Spanish	/Na/ and /La/	/Da/ and /La/	Syllable and Lyrics Mixed
Number of recordings	10	15	7	1	5
Steps 1+2	**0.612**	0.609	0.325	0.178	0.448
Steps 1+3	0.443	0.602	0.396	0.652	0.575
Steps 1+2+3	0.523	**0.709**	**0.520**	**0.677**	**0.596**

Table 3: Comparison of transcription performance (F-measure of COnPOff) for different categories of lyrics.

Methods	COnPOff (F-measure)	Merged	Split
Step 1	**0.645**	0.023	**0.004**
Steps 1+2	0.603	0.018	0.045
Steps 1+2+3	0.614	**0.005**	0.069

Table 4: Transcription and segmentation performance comparison for the dataset of Dai et al. (2015).

Since the English language songs match the training data quite well, there are very few merge errors in these songs after Step 2, and the subsequent segmentation causes over-segmentation and degrades performance. In other cases, Step 3 improves performance, especially for non-linguistic samples.

For non-linguistic content, we are wary of making strong claims as the amount of data is quite small. We observe that the Step 2 output is considerably worse with non-linguistic syllables than on songs with linguistic content, but this difference is diminished when Step 3 is included in the pipeline. Overall, Table 3 shows that the inclusion of phonetic information is consistently beneficial for segmentation in various linguistic scenarios.

We also test our methods on data from another dataset (Dai et al., 2015), in which singers perform three tunes using the syllable /Ta/. In Table 4, we show results for 12 recordings (singers 1,2,4,7). Step 1 performs relatively well because in this case each musical note comprises a voiced segment preceded by a voiceless consonant, so the voicing-based segmentation reflects the note boundaries. In this context Steps 2 and 3 cause split errors and reduce performance.

By analysing specific examples, we identified two sources of errors made by our system. Firstly, there are unrecognized phonemes that lead to merged errors, which could potentially be due to the constraints exerted by the pronunciation and language models of the lyrics transcriber. Secondly, the input pitch track is inactive during voiceless consonants, while the ground truth annotations of the dataset usually include the voiceless consonant at the end of a syllable, resulting in a longer duration note. This disagreement causes a number of offset errors. Despite these errors, we obtained state-of-the-art results on a public dataset for the task of sung note segmentation.

References

E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. 2013. Automatic music transcription: Challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434.

A. de Cheveigné and H. Kawahara. 2002. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America*, 111(4):1917–1930.

J. Dai and S. Dixon. 2019. Intonation trajectories within tones in unaccompanied soprano, alto, tenor, bass quartet singing. *Journal of the Acoustical Society of America*, 146(2):1005–1014.

J. Dai, M. Mauch, and S. Dixon. 2015. Analysis of intonation trajectories in solo singing. In *16th International Society for Music Information Retrieval Conference*, pages 420–426.

Emir Demirel, Sven Ahlbäck, and Simon Dixon. 2020. Automatic lyrics transcription using dilated convolutional neural networks with self-attention. In *International Joint Conference on Neural Networks*. ArXiv: 2007.06486.

Zih-Sing Fu and Li Su. 2019. Hierarchical classification networks for singing voice segmentation and transcription. In *International Society for Music Information Retrieval Conference*.

Emilia Gómez and Jordi Bonada. 2013. Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms as applied to A Cappella singing. *Computer Music Journal*, 37(2):73–90.

Hoon Heo and Kyogu Lee. 2017. Robust singing transcription system using local homogeneity in the harmonic structure. *IEICE Transactions on Information and Systems*, E100.D(5):1114–1123.

Hsuan-Huei Shih, S.S. Narayanan, and C.-C.J. Kuo. 2002. An HMM-based approach to humming transcription. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 337–340. IEEE.

Matthias Mauch, Chris Cannam, Rachel Bittner, George Fazekas, Justin Salamon, Jiajie Dai, Juan Bello, and Simon Dixon. 2015. Computer-aided Melody Note Transcription Using the Tony Software: Accuracy and Efficiency. In *First International Conference on Technologies for Music Notation and Representation (TENOR 2015)*, pages 23–30.

Matthias Mauch and Simon Dixon. 2014. PYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663. IEEE.

Emilio Molina, Ana M. Barbancho, Lorenzo J. Tardón, and Isabel Barbancho. 2014. Evaluation framework for automatic singing transcription. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 567–572.

Emilio Molina, Lorenzo J. Tardón, Ana M. Barbancho, and Isabel Barbancho. 2015. SiPTH: Singing transcription based on hysteresis defined on the pitch-time curve. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(2):252–263.

Matti P. Ryynänen and Anssi P. Klapuri. 2004. Modelling of note events for singing transcription. In *ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing*.

Craig Stuart Sapp. 2006. Mazurka Project plugins for Sonic Visualiser. *Poslední aktualizace*, 6(5).

Luwei Yang, Akira Maezawa, Jordan B. L. Smith, and Elaine Chew. 2017. Probabilistic transcription of sung melody using a pitch dynamic model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 301–305. IEEE.

Using Listeners' Interpretations in Topic Classification of Song Lyrics

Varvara Papazoglou **Robert Gaizauskas**
Department of Computer Science, University of Sheffield
{vpapazoglou1, r.gaizauskas}@sheffield.ac.uk

Abstract

Incorporating listeners' interpretations of song lyrics has been shown to significantly improve topic classification accuracy. Using a different type of interpretation, as compared to previous research, we propose four possible representations of songs as input for classification systems. The results show that (a) some representations are consistently better than others, and (b) the similarity of topic classes along with the ambiguity of song lyrics may affect the classification accuracy, which argues for using top-n classification (n>1) and associating multiple top ranking classes with each song. We also examine the case of training a system on both lyrics and interpretations and testing it on songs that lack interpretations.

1 Introduction

Song lyrics differ from prose text in various ways: they tend to be more ambiguous, to contain more figures of speech, to break syntactic rules, to be accompanied by music and to have a rhythm. Considering that the majority of popular music contains lyrics, it is assumed that a lot of information about songs can be extracted from lyrics. This information can be useful for many Music Information Retrieval (MIR) tasks, such as music recommendation, classification, and search.

We focus on the task of automatic topic classification of English-language songs based on song lyrics and interpretations of them. The interpretations have been retrieved from a website that hosts song lyrics and interpretations generated by the website's users. Our approach is novel in that (a) opposed to previous research, the interpretations we use refer to specific fragments of lyrics and not to the whole song (so it is less probable that they contain information unrelated to the topic of the song), (b) we propose a novel representation of songs using lyrics and interpretations, which consistently achieves high classification accuracy, (c) we examine a top-n topic classification approach, and (d) we combine lyrics and interpretations in an attempt to improve classification of unseen lyrics for which there are no available interpretations (e.g., recently released songs).

Our aim is to investigate what is the best song representation for the task and to create a system which predicts topics that meet listeners' needs and expectations. Our main hypotheses are that interpretations are more informative than lyrics in determining the topic of a song, and that a top-n topic classification approach is useful from the users' perspective. The intuition for the latter is that a song can actually belong to more than one topic class either because some topic classes are semantically related or because the song has indeed more than one topic. To illustrate our point, let us consider the case of a song A which talks about a breakup and heartache, and a song B which talks about a breakup but not heartache Although we could initially consider these two topics to be similar, there are cases of songs which do not belong to both topics. We assume that a user who searches for music based on the topic of the lyrics would be satisfied if multiple related topics were assigned to a song instead of a single one.

Our results suggest that listeners' interpretations of lyrics indeed improve the accuracy of the classification, and that top-n classification is an effective approach. However, using lyrics and their interpretations in the training stage and lyrics in the test stage does not significantly and consistently improve accuracy compared to using solely lyrics for both stages.

2 Related Work

Lyrics have been used in a range of MIR tasks, sometimes combined with acoustic properties of the respective songs. Watanabe and Goto (2020) introduce Lyrics Information Processing (LIP) as a research field specific to analysis and generation of lyrics, and present a range of applications. It is

Song Topics			
Sex	317	Political statement	191
Heartache	294	Death	190
Girl	277	War	185
Religion	272	Events in the news	179
Drugs	265	Cheating	158
Ex-partner	240	Dealing with fame	156
Parent	208	Autobiographical	154
Dead friend	205	Depression	154
Places	203	Criminals	147
Breakup	199	Loneliness/isolation	147

Table 1: Number of songs per topic in the dataset (prior to splitting into training and test set).

also worth mentioning that users of music search systems appear to use lyrics frequently (Lee and Downie, 2004).

Some of the first approaches to topic detection of song lyrics use clustering methods. Kleedorfer et al. (2008) used Non-Negative Matrix Factorisation; Sasaki et al. (2014) used Latent Dirichlet Allocation in order to detect and visualise five of the latent topics of the lyrics in an interactive system.

More recent research exploits listeners' interpretations of lyrics for topic classification (Choi, 2018; Choi and Downie, 2018; Choi et al., 2016, 2014). The highest accuracy was achieved when interpretations or the concatenation of lyrics and interpretations were used as features instead of the lyrics alone.

3 Data

Song topics and song titles are collected from Songfacts[1]. Songfacts provides information about songs and artists and assigns categories to the songs manually, based on sources like interviews, publicity releases, press, etc. We collect all the song titles and topics from the category "about", which contains 206 topics. There is no hierarchy in the topics, and some songs belong to more than one topic.

These song titles are then searched for in Genius[2], from where their lyrics and their interpretations are collected. In Genius, users annotate specific fragments of lyrics (e.g., one or more consecutive words or lines) with an interpretation. The users can upvote and downvote the suggested annotations, so the final interpretations usually reflect the single most widely acceptable view on the meaning of the song.

[1]https://www.songfacts.com

[2]https://genius.com

We selected the 20 most populated topics for our dataset (Table 1). The intuition is that in order to meet listeners' needs, the system should cover a relatively large number of topics, while at the same time guarantee that there are enough songs per topic for training the classifier. In this set the vast majority of songs belong to a single topic. The few songs left belonging to multiple topics are then assigned to the less populated of the 20 topics. Prior to this, we ensure that the language of each selected song's lyrics is English, using the Python module langdetect[3], a port of a library by Nakatani (2010). The final training dataset is balanced, consisting of 20 topics (130 songs each) and a total number of 2,600 songs, and we also have an unbalanced test set with 1,541 songs. We represent each song in four ways:

1. **Lyrics**: only the lyrics of the song (without the song title).

2. **Interpretations**: concatenation of all interpretations of the lyrics. If the annotated lyric fragments are repeated, the respective interpretations are repeated as well.

3. **Mixed**: starting with the lyrics, we detect fragments which have been annotated with an interpretation and replace these fragments with their respective interpretations. The rest of the lyrics remain unchanged (repetitions are preserved).

4. **Concatenation** of the first two representations.

All text is lowercased, contractions are expanded using the Python module contractions[4], song structure annotations (e.g.: "[Chorus]") are removed, and lemmatisation (WordNet lemmatiser) and stemming (Porter stemmer) are performed, using the Python module NLTK[5].

4 Experimental Setup

Using the scikit-learn Python library[6], we use TFIDF scores of unigrams as features for each of our four song representations. Unigrams with document frequency less than 5 are discarded. Using 5-fold stratified cross-validation we train each

[3]https://pypi.org/project/langdetect

[4]https://pypi.org/project/contractions

[5]https://www.nltk.org

[6]https://scikit-learn.org version 0.24.1

	Lyrics		Interpretations		Mixed		Concatenation		Concat. & Lyrics	
	top-1	top-3	top-1	top-3	top-1	top-3	top-1	top-3	top-1	top-3
kNN	0.1901	0.3790	0.2842	0.4822	0.2862	0.4646	0.2706	0.4763	0.1707	0.3310
LR	**0.3348**	**0.5892**	**0.4549**	**0.6788**	**0.4692**	**0.7333**	**0.4802**	**0.7352**	**0.3355**	**0.5964**
MNB	0.2732	0.5204	0.3180	0.5626	0.3731	0.6230	0.3679	0.6178	0.2726	0.5088
RF	0.2330	0.4276	0.3783	0.5912	0.4114	0.6424	0.3855	0.6275	0.2524	0.4640

Table 2: Accuracy scores for top-1 and top-3 classification with N=20 topic classes.

of four classification algorithms (described in the next paragraph) on each representation. In the top-1 classification approach, we consider the predicted topic to be the one that the classifier predicts. In the top-n classification approach for n>1, during the testing stage each of the classifiers returns the predicted probabilities per topic class for the current song. If the true topic class is one of the top-n predicted topic classes, then we consider the song to have been classified correctly; otherwise we consider the class with the highest probability to be the predicted class and the song to have been classified incorrectly. The intuition is that a song with a true label A but predicted label B should not be considered misclassified if A and B are in the top-n predicted classes. Besides, songs can be interpreted in different ways, so returning a small number of possible topics to a user who searches for songs on a specific topic is acceptable. Moreover, this approach covers cases of topics that are semantically similar to each other (if A and B are semantically similar, then predicting B should not be considered incorrect). Since we have a dataset with N=20 classes, we have selected n=3. For smaller N values we prefer decreasing n as well (e.g., for N=10, n=2 is intuitively more appropriate).

We have experimented with the following classification algorithms[7] (their parameters were selected using grid search): k-Nearest Neighbours (kNN, n_neighbors=5, weights='distance'), Logistic Regression (LR, random_state=17, max_iter=1000), Multinomial Naïve Bayes (MNB, default parameters), Random Forest (RF, random_state=17, class_weight='balanced', criterion='gini')

We also perform two classification experiments: in the first experiment, the training and test stages use features of the same song representation (i.e., either lyrics or interpretations or mixed representation or concatenation), while in the second experiment the training stage uses the mixed concatenation of lyrics and interpretations, and the test stage uses only lyrics. This is to test the hypothesis that training on lyrics and interpretations will lead to better classification of new songs without accompanying interpretations. We use the same training and test sets for both experiments.

[7]From scikit-learn.

5 Results

Table 2 contains the accuracy scores for each algorithm for both experiments. LR consistently returns the highest accuracy for all settings. For the first experiment, the two representations that return the highest accuracy are the mixed lyrics-interpretations representation and the concatenation of lyrics and interpretations. When only lyrics are used, the accuracy is consistently lower.

Training each classifier on the concatenation of lyrics and interpretations and testing on lyrics compared to training solely on lyrics (second experiment, last column in Table 2) improves results significantly with RF and marginally with LR, while it actually reduces accuracy scores with kNN and MNB.

6 Discussion

Results do not support the hypothesis that training on lyrics and interpretations will improve classification of unseen lyrics without interpretations. However, this does not necessarily imply that combining lyrics with interpretations is not helpful in improving classification of song lyrics. It is possible that better feature engineering and preprocessing might actually make this approach very effective.

Comparing the results between top-1 and top-3 classification approaches we noticed that there are indeed some frequently confused topic classes, such as: (a) events in the news, political statements, war, (b) heartache, breakup, ex-partner, cheating. In both examples, the classes seem to be

	Lyrics		Interpretations		Mixed		Concatenation		Concat. & Lyrics	
	top-1	top-2	top-1	top-2	top-1	top-2	top-1	top-2	top-1	top-2
kNN	0.3082	0.6123	0.4073	0.6974	0.4184	0.7322	0.4254	0.7280	0.3040	0.5858
LR	**0.4784**	**0.8020**	**0.5593**	**0.8466**	**0.6318**	**0.9038**	**0.6346**	**0.9052**	**0.4909**	**0.7894**
MNB	0.4198	0.7378	0.4658	0.7922	0.5467	0.8131	0.5467	0.8020	0.4561	0.6960
RF	0.4114	0.7308	0.5216	0.8089	0.5858	0.8452	0.5635	0.8466	0.4059	0.7075

Table 3: Accuracy scores for top-1 and top-2 classification with N=8 topic classes.

similar to each other. This suggests that topic similarity should be taken into account in our dataset.

A comparison of our results to previous research is very useful in order to evaluate our approach. In previous research (Choi, 2018; Choi and Downie, 2018; Choi et al., 2016, 2014), interpretations are in the form of general comments about the lyrics of the whole song and frequently contain information other than the meaning of the lyrics (e.g., how much the particular listener likes the song and why, what it reminds them of, comments about the album or a live concert in which the song was played or a music video, etc.). Both Choi's and our research retrieve song titles and topics from Songfacts. A direct comparison between the results of Choi's and our research is difficult, as we cannot use the same songs mostly due to the availability of interpretations and the fact that we cannot obtain the same dataset from Songfacts, which is updated regularly with new songs and information. However, we try to follow similar preprocessing and feature extraction steps, with the difference that we do not eliminate stopwords; the use of TFIDF weighting lowers the impact of terms with very high frequency in the dataset, so that using a list of standard and corpus-specific stopwords is not required, and in our experiments we did not notice any significant difference with stopword removal. Choi et al. (2016) use a dataset of 800 songs and 8 balanced topic classes that consists of lyrics, interpretations, and concatenation of them. Then, using TFIDF features, they compare four classifiers: kNN, SVM with a linear kernel, SVM with radial basis function kernel, and Naïve Bayes. The highest accuracy score (0.66) is achieved by Linear SVM, using the concatenation of lyrics and interpretations. Interpretations and concatenation consistently return higher accuracy than lyrics. Using fasttext[8] word embeddings and Naïve Bayes on the same dataset, concatenation returns again the highest accuracy (0.5788) (Choi, 2018). Table 3 shows the accuracy scores we achieve with the same four classifiers using our four representations, on the top 8 balanced topic classes (training set: 1,440 songs, test set: 717 songs). For top-1 classification, the highest accuracy score is 0.6346 using concatenation with LR, which appears to be similar with the results achieved in Choi et al. (2016). Using top-2 classification, the accuracy score is significantly improved, reaching 0.9052. For training on concatenation and testing on lyrics, accuracy scores follow a different trend than with N=20, but are still low. Finally, we preferred to use top-2 instead of top-3, due to the small number of topic classes.

7 Conclusion and Future Work

Our results suggest that the interpretations of the lyrics are indeed more informative than lyrics alone for identifying the topic of the lyrics, which is in line with previous research. The main differences compared to previous research are that: (a) we use interpretations targeted on specific fragments of lyrics instead of interpretations which are in the form of general comments about the lyrics, (b) we examine the impact that training a model with lyrics and their interpretations has on predicting the topic class of unseen song that lack interpretations, and (c) we allow for more flexibility in classification by accepting as correctly classified the songs which have the correct topic class as one of their top-n predicted topic classes. The latter is a reasonable approach for MIR applications which return to the user a list of songs of a selected topic or predict the topic of a specific song.

Using interpretations in the form of comments on specific fragments of lyrics allows us to analyse song lyrics in detail. We are planning to study the possible different impact of chorus and verse terms in topic classification, as well as to experiment with features other than TFIDF unigrams, e.g. word embeddings, and to examine human performance as an evaluation of our approach.

[8]https://fasttext.cc

References

Kahyun Choi. 2018. *Computational lyricology: quantitative approaches to understanding song lyrics and their interpretations*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Kahyun Choi and J. Stephen Downie. 2018. Exploratory investigation of word embedding in song lyric topic classification: Promising preliminary results. In *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*, JCDL '18, page 327–328, New York, NY, USA. Association for Computing Machinery.

Kahyun Choi, Jin Ha Lee, and J. Stephen Downie. 2014. What is this song about anyway?: Automatic classification of subject using user interpretations and lyrics. In *IEEE/ACM Joint Conference on Digital Libraries*, pages 453–454.

Kahyun Choi, Jin Ha Lee, Xiao Hu, and J. Stephen Downie. 2016. Music subject classification based on lyrics and user interpretations. *Proceedings of the Association for Information Science and Technology*, 53(1):1–10.

Florian Kleedorfer, Peter Knees, and Tim Pohle. 2008. Oh oh oh whoah! towards automatic topic detection in song lyrics. In *Proceedings of the 9th International Conference on Music Information Retrieval, ISMIR 2008*, pages 287–292.

Jin Ha Lee and J. Stephen Downie. 2004. Survey of music information needs, uses, and seeking behaviours: preliminary findings. In *Proceedings of the 5th International Conference on Music Information Retrieval, ISMIR 2004*.

Shuyo Nakatani. 2010. Language detection library for java.

Shoto Sasaki, Kazuyoshi Yoshii, Tomoyasu Nakano, Masataka Goto, and Shigeo Morishima. 2014. Lyricsradar: A lyrics retrieval system based on latent topics of lyrics. In *Proceedings of the 15th International Conference on Music Information Retrieval, ISMIR 2014*, pages 585–590.

Kento Watanabe and Masataka Goto. 2020. Lyrics information processing: Analysis, generation, and applications. In *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, pages 6–12, Online. Association for Computational Linguistics.

Music Playlist Title Generation: A Machine-Translation Approach

SeungHeon Doh[1] **Junwon Lee**[2] **Juhan Nam**[1]
Graduate School of Culture Technology, KAIST, South Korea[1]
Department of Electrical Engineering, KAIST, South Korea[2]
{seungheondoh, james39, juhan.nam}@kaist.ac.kr

Abstract

We propose a machine-translation approach to automatically generate a playlist title from a set of music tracks. We take a sequence of track IDs as input and a sequence of words in a playlist title as output, adapting the sequence-to-sequence framework based on Recurrent Neural Network (RNN) and Transformer to the music data. Considering the orderless nature of music tracks in a playlist, we propose two techniques that remove the order of the input sequence. One is data augmentation by shuffling and the other is deleting the positional encoding. We also reorganize the existing music playlist datasets to generate phrase-level playlist titles. The result shows that the Transformer models generally outperform the RNN model. Also, removing the order of input sequence improves the performance further.

1 Introduction

Music playlists have gained progressively more importance in music streaming services. A playlist represents a group of music tracks that shares similar genre, mood or musical context. When a new playlists is created by curators or users, or generated by recommender systems, they deliver messages about musical needs by providing playlist titles in a phrase (Pichl et al., 2015; Dias et al., 2017). However, it is not trivial to blend semantics of the music tracks and express them with a phrase. As a result, we often find noisy playlist titles which do not accord with the music tracks.

A fundamental issue in automatic playlist title generation is to extract the common semantic features from the music tracks in a playlist, independent of the number of tracks. This issue has been addressed by representing a playlist with track embedding averaging (Hao and Downie, 2020) or a sequential model (Choi et al., 2020). In (Hao and Downie, 2020), they treated playlists as the equivalent of phrases, and tracks as the equivalent of words. They then used the the word2vec model to learn the track embedding. In (Choi et al., 2020), they represented playlists and tracks as a matrix where the columns correspond to playlist IDs and the rows to track IDs. They then used a matrix factorization technique to learn the track embedding and, furthermore, applied an average or sequence model to predict high-level categorical labels.

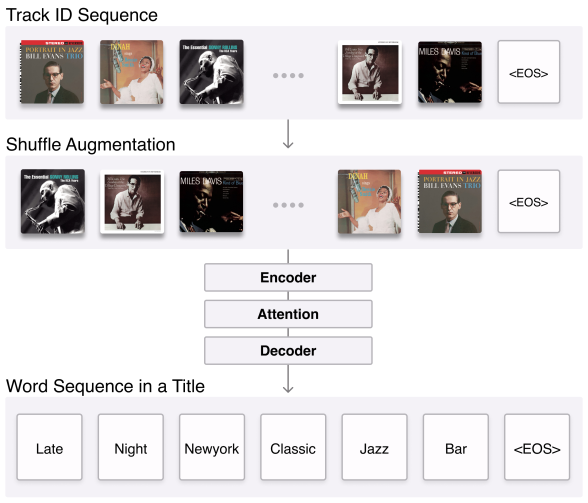


Figure 1: Diagram of track ID sequence to word sequence in a title.

Another issue is to generate a natural word sequence (e.g., a phrase or a sentence) as a playlist title from the common semantics of music tracks. This sequence-to-sequence setting is similar to the machine translation task. Therefore it is natural to attempt the methods in machine translation, in particular, the encoder-decoder models (Bahdanau et al., 2014; Vaswani et al., 2017). This approach was previously attempted for playlist title generation (Samaniego, 2018). However, the model output was mostly tag-level titles (e.g., a single word or short phrase) rather than phrase-level titles, presumably because they used an unfiltered noisy dataset and a simple RNN model. Also, they used the track name as an input sequence. This input setting can confine tracks with similar names to have similar semantics, and also can learn the order of input sequence, which may be discarded in music playlists (Hao and Downie, 2020).

In this paper, we present another machine translation approach based on the encoder-decoder framework for automatic playlist title generation as illustrated in Figure 1. Our contribution is as follows: (i) we compare two encoder-decoder models based on RNN and Transformer, (ii) we propose two simple techniques to make track ID sequence orderless and show that they improve the performance, and (iii) we propose a new data split by filtering existing playlist datasets and extracting phrase-level playlist title.

2 Dataset and Preprocessing

We apply our proposed approach to two different datasets respectively: Melon Playlist Dataset (Melon) (Ferraro et al., 2021) and Spotify Million Playlist Dataset (MPD) (Chen et al., 2018). As our task is generating a playlist title in phrase for a given track ID sequence, we need a dataset of playlists that contains a pair of track ID sequence and title. Both Melon and MPD satisfy this requirement and support different languages (Korean and English). In Melon, playlist titles are written in both Korean and English (some of titles are mixed with both languages). In case of English words, normalization was done by substituting all characters with lowercase. Both of the languages were simply tokenized by white spaces.

In our task, an ideal playlist title is a phrase that incorporates common features among the songs in a playlist. However, Melon and MPD were originally constructed for automatic playlist continuation (APC) task and so they have several problems to directly use them. First, there are many playlist titles that cannot be considered as a phrase. Melon includes 27,420 playlists with empty titles which is 18.4% of the total playlists. In the case of MPD, 646,868 playlists have titles with a single word which amount to 64.7% of the total. In addition, there are playlist titles that have multiple tokens but not a phrase, for example, "G e o r g e W i n s t o n e" and "beyonce - 4". Finally, some playlists have zero or few songs which are typically not considered as a playlist. The statistic of the two datasets is summarized in the *Original* column of Table 1.

Dataset	Statistic	Original	Filtered
Melon Playlist	Playlist Number	148,826	51,723
	Unique Track Number	649,092	430,746
	Unique Title Number	115,318	50,296
	Unique Word Number	88,524	56,296
	Average Char Length	2.8	3.6
	Average Title Length	3.6	4.7
	Average Track Length	39.7	46.2
Spotify Million Playlist	Playlist Number	1,000,000	50083
	Unique Track Number	2,262,292	402,523
	Unique Title Number	17,381	1,859
	Unique Word Number	11,146	1,886
	Average Char Length	5.2	4.2
	Average Title Length	1.4	3.4
	Average Track Length	66.3	66.3

Table 1: Compare statistic of datasets. After filtering, as the average title length increases, it can be seen that the noise of each phrase has been removed.

We reorganized the two datasets with the same criteria to improve the quality of data samples for playlist title generation. First, we gather all playlists provided by each dataset. In the case of Melon, we merged the provided train, validation, and test set into one, and then filtered out some playlists with three criteria. First, the number of title tokens should be more than 3. Second, the number of tracks should be more than 10. Third, the average character length of title tokens should be more than 3.

Finally, the filtered dataset is split by the number of title tokens. Playlists with the same number of title tokens are randomly split with a ratio of 8:1:1 and merged among different numbers of tokens subsequently to form train, validation, and test set. As a result, the statistics of data was changed as as shown in *Filtered* column of Table 1. The longer average character length and title length indicate the portion of playlist titles in phrases within the dataset has increased.

3 Playlist Title Generation

3.1 Encoder-Decoder Model

The model for playlist title generation is composed of an encoder and a decoder. The goal of the model is to find a title word sequence y that maximizes the conditional probability of y given a source track ID sequence x. The encoder reads a track ID sequence $x = (x_1, .., x_n)$, represents track ID as an embedding vector using random initialized embedding matrix $E \in \mathbb{R}^{|V| \times d}$, and transforms it to hidden states $z = (z_1.., z_n)$. The decoder takes these hidden states as a context input and outputs a

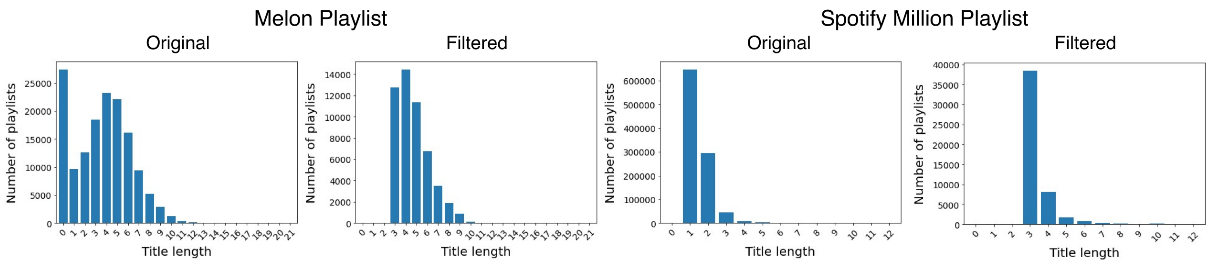


Figure 2: Compare distribution of datasets(original, filtered), title length 0 means missing data, and title length 1 means tag-level title.

summary $y = (y_1.., y_m)$. At each step the model is auto-regressive, consuming the previously generated symbols as an additional input when generating the next. During training, we used the softmax cross-entropy loss. The encoders and decoders can be RNN (Bahdanau et al., 2014), Convolutional Neural Network (CNN) (Gehring et al., 2017) or self-attention layer (Vaswani et al., 2017). In this paper, we compare the RNN model and the Transformer model composed of self-attention layers.

RNN Model: Our baseline model corresponds to the neural machine translation model used in (Bahdanau et al., 2014). The encoder consists of bidirectional Gated Recurrent Unit (GRU) (Chung et al., 2014), while the decoder consists of a unidirectional GRU with the same hidden-state size as that of the encoder, and an attention mechanism over the source-hidden states and a soft-max layer over the target vocabulary to generate words.

Transformer: The encoder and decoder are composed of multi-head self-attention layers and position-wise fully connected feed-forward network with a residual connection and a layer normalization.(Vaswani et al., 2017). The transformer views the encoded representation of the input as a set of key-value pairs and both the keys and values are the encoder hidden states. In the decoder, the previous output is compressed into a query and the next output is produced by mapping this query and the set of keys and values. The output of self-attention layer is a weighted sum of the values, where the weight is calculate by the dot-product the query with all the keys.

3.2 Ignoring the Order in Track Sequences

One of the characteristics of playlists is that the order of tracks in a playlist is generally not important. This feature can be exploited for data augmentation. In this paper, we propose two different techniques. The first is sequence shuffling which randomly changes the order of tracks in the same playlist. The second is to remove the positional encoding of the encoder. According to the loss of position information, the model can recognize the track sequence except for the sequence information of the data. On the other hand, the decoder applies the positional encoding to the word sequence for title generation. We applied the two techniques independently, because, when the positional encoding is removed, the transformer model does not recognize the input sequence differently regardless of shuffling.

Model	Melon		MPD	
	Val NLL	Test NLL	Val NLL	Test NLL
RNN Model	7.482	7.384	2.453	2.357
Transformer	7.150	7.124	1.821	1.805
+ shuffle aug	**6.952**	**7.019**	**1.543**	**1.502**
+ delete pos	7.036	7.099	1.552	1.538

Table 2: Validation and test NLL for melon and spotify million playlist dataset. The **shffle aug** means data augmentation through shuffling the input track sequence, and **delete pos** means that delete encoder's positional encoding in vanilla transformer.

3.3 Training Details

We fixed the number of layers of encoder and decoder to two and 128 embedding dimensions and 256 hidden dimensions for fair comparison in the two types of encoder-decoder models. We trained the model using a single GPU. We optimized the model using the Adam optimizer (Kingma and Ba, 2014) with a 0.005 learning rate, and 0.0001 learning rate decay for all models and datasets. We used a batch size of 64 and randomly shuffled the training data at every epoch. We used early stopping on the validation set, monitoring with the validation loss, and used the best model on the validation set to report all performance numbers.

Playlist ID	49169	22501
Input Tracks	Swear by Inc., Millionairess by Inc., Her Favorite Song (w/Crossfade) by Mayer Hawthorne, Dontcha by The Internet, All I Do by Majid Jordan, Her by Majid Jordan, Us by MOVEMENT, The Place by Inc., Coffee (Feat. Wale) by Miguel, Under Control by The Internet, Somthing‘s Missing by The Internet, Ocean Drive by Duke Dumont, Drive by Dornik, Make It Work by Majid Jordan, Jump Hi (Feat. Childish Gambino) by Lion Babe, Treat Me Like Fire by Lion Babe, sHe by ZAYN, Hallucinations by dvsn, Dapper (Feat. Anderson .Paak) by Domo Genesis, Bone + Tissue by Gallant, Miyazaki by Gallant, ...	Take Five by Michel Camilo, Angelina by Tommy Emmanuel, Monk‘s Dream (Live) by Martin Reiter, Stairway To Love by George Benson, Birdsong by Tommy Flanagan, Come Fly With Me by Frank Sinatra, Gemini by Chick Corea, Cheesecake by Dexter Gordon, Kathy by Horace Silver, Love Me by The Little Willies, Perdido by Earl Hines, ‘Round Midnight by Hank Jones, I Just Called To Say I Love You by Harry Allen, Killing Me Softly With His Song by Harry Allen, Let‘s Fall In Love by Diana Krall, Flight To Jordan by Duke Jordan, Quizas Quizas Quizas by Lisa Ono, ...
Ground Truth	late night drive	가을밤 로맨틱 재즈곡들 romantic jazz songs for an autumn night (translated)
RNN Model	몽환적인 r&b r&b dreamy r&b r&b (translated)	jazz jazz jazz jazz
Transfomer	생생한 고음질로 만나는 hi-fi 위클리 12월 16일 vol 1 lively and high-quality sound in hi-fi weekly December 16th vol 1 (translated)	카페에서 듣는 음악들 music in a cafe (translated)
Transfomer + shuffle aug	들을수록 좋은 세련되고 감각적인 pop stylish and sensual pop that you feel better as you listen more (translated)	카페에서 듣는 감각적인 재즈 sensational jazz in a cafe (translated)
Transfomer + delete pos	내가 좋아하는 노래 my favorite song (translated)	카페에서 듣는 잔잔한 음악 calm music in a cafe (translated)

Table 3: Inference example from the melon playlist test dataset. Reference refers to the ground turth of the dataset. Each first line is a generation result, and the second line is a phrase translated from Korean to English. Source means input track sequence, and track index over 15 are excluded.

4 Results and Discussion

4.1 Quantitative Results

We used negative log-likelihood (NLL) as an evaluation metric for the models. Table 2 lists the NLL values for the RNN and Transformer models on the two datasets. The result shows that the Transformer models generally outperform the baseline RNN model on both datasets. In addition, the Transformer models that ignore the order of track sequence improve the performance further. Between the two techniques, shuffling augmentation has a slightly lower NLL value than deleting the positional encoding on on both datasets. This indicates the data augmentation approach that involves ignoring the order is more effective than simply removing the order information.

4.2 Qualitative Results

Table 3 shows two examples of title generation given an input track sequence. We can first see that the RNN models generate a short title. They even have repetitions of the genre words (e.g., rb, jazz). On the other hands, the Transformer models generates a natural phrase composed of more than 3 different words. An interesting result in the example on the left side is that the basic Transformer model has a very specific title which seems to be copied from data with strong context (“hi-fi weekly December 16th, vol. 1”). This problem is alleviated in the Transformer models with shuffle augmentation or deleting the position encoding.

5 Conclusions

In this work, we propose music playlist title generation with a machine translation approach. There are several future directions to extend this work. First, we can try various track embedding vectors for the input sequence. For example, we can use tag prediction vectors or audio embedding vectors from music auto-tagging models or track embedding vector from matrix factorization of user listening data. Second, we should investigate more quantitative metrics to evaluate the models. The BLEU score used in machine translation may be a possibility in terms of accuracy but we should also consider the diversity of the generated playlist titles to provide rich expressions for music listeners. Finally, we need to have a user study designed systemically that compare different models of playlist title generation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. 2018. Recsys challenge 2018: Automatic music playlist continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys)*.

Jeong Choi, Anis Khlif, and Elena Epure. 2020. Prediction of user listening contexts for music playlists. In *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of the Advances in neural information processing systems, Workshop on Deep Learning (NeurIPS)*.

Ricardo Dias, Daniel Gonçalves, and Manuel J Fonseca. 2017. From manual to assisted playlist creation: a survey. *Multimedia Tools and Applications*.

Andres Ferraro, Yuntae Kim, Soohyeon Lee, Biho Kim, Namjun Jo, Semi Lim, Suyon Lim, Jungtaek Jang, Sehwan Kim, Xavier Serra, et al. 2021. Melon playlist dataset: A public dataset for audio-based playlist generation and music tagging. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Yun Hao and J Stephen Downie. 2020. Using latent semantics of playlist titles and descriptions to enhance music recommendations. In *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations, ICLR*.

Martin Pichl, Eva Zangerle, and Günther Specht. 2015. Towards a context-aware music recommendation approach: What is hidden in the playlist name? In *Proceedings of IEEE International Conference on Data Mining Workshop (ICDMW)*.

Sofía Samaniego. 2018. Playlist title generation using sequence to sequence. *Standford Unversity, cs224n project report*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in neural information processing systems (NeurIPS)*.

Are Metal Fans Angrier than Jazz Fans? A Genre-Wise Exploration of the Emotional Language of Music Listeners on Reddit

Vipul Mishra[1] vipul-mi@is.naist.jp

Kongmeng Liew[1] liew.kongmeng@is.naist.jp

Romain Hennequin[2] deezer@research.com

Elena V. Epure[2] deezer@research.com

Eiji Aramaki[1] aramaki@is.naist.jp

[1]Nara Institute of Science and Technology, Nara, Japan
[2]Deezer Research, Paris-IDF, France

Abstract

Music forms a big part of our identity and as such, people with a shared preference for certain kinds of music may also share similar traits. In this study, we explore differences in the emotional language of fan communities of different music genres. In focusing on Reddit, we analyze the utterances on online community forums of different music genres using lexicon-based sentiment (emotion) analysis. Upon clustering Subreddit forums, we obtained two clusters: forums discussing genres like Rock, RnB, Country, and Jazz were found to have a higher abundance of positively valenced emotions and a lower amount of negatively valenced emotions. Likewise, Subreddits discussing genres like Metal, Punk, and Rap had a lower amount of positively valenced emotions and a higher abundance of negatively valenced emotion. We observed a high correlation between counts in lyrics of a genre and counts in a fan community for the emotions of anger, disgust, fear, and joy. In sum, we found differences in the emotional language of fan utterances by genre, and these could be partially attributed to the emotions contained in the lyrics.

1 Introduction

How music and emotions relate to each other has long been a matter of interest to researchers. Music can induce certain feelings or emotions in the listeners, which can differ from the emotions expressed explicitly in the music. This mood-induction function of music is why music is commonly used for mood regulation or mood management (Saarikallio 2011; Thoma et al. 2012; Papinczak et al. 2015). It has also been found that different styles, or genres of music, are associated with different strategies of mood regulation. For example, Cook et al. (2019) found that genres of electronica and dance music were associated with use for increasing emotional arousal, while classical music was associated with use for negative mood management. These results suggest that preference for a certain style of music may be associated with a particular set of emotions. Although the link of emotions to various styles of music has been explored in the music psychology literature, these studies have mostly focused on the immediate reactions induced by the music in study participants (Zentner et al. 2008; Sharman and Dingle 2015; Merz et al. 2020). While there have been a few studies that explored the emotions related to specific genre preferences, they have mainly relied on self-reported surveys and questionnaires which can be subjective. Emotionality, or emotional language, associated with preference for music genres has not yet been explored using real-world, naturalistic data. Motivated by this research gap, we explore the emotions expressed in online fan communities of 10 different music genres, to identify the emotion they are strongly associated with. Specifically, we investigate the emotional language associated with 10 Subreddits corresponding to the music genres listed in Table 1.

Reddit[1] is a website having content rating and discussion forums dedicated to different topics and each discussion forum on Reddit is called a Subreddit. For example, r/electronicmusic is a Subreddit for discussing all things related to electronic music.

Additionally, we also run correlation analyses to determine if the prevalence of a type of emotion in the Subreddit is reflective of its prevalence in the music itself. We analyze the track lyrics of each genre in a fashion similar to the comments on the Subreddits to get the prevalence of each type of emotion.

[1]https://www.reddit.com/

Genre Subreddit	Number of comments	Avg. number of tokens	Number of unique users	Lyrics genre	Number of tracks	Avg. number of tokens	Avg. release year
r/Metal	2,069k	30.4	109k	Metal	8.0k	185	2002
r/electronicmusic	858k	26.3	128k	Electronic	5.6k	180	2003
r/punk	728k	26.3	64k	Punk	2.7k	193	1995
r/Jazz	434k	33.0	58k	Jazz	3.3k	173	1991
r/rap	218k	21.4	54k	Rap	7.3k	572	2002
r/country	58k	27.9	12k	Country	6.8k	219	1996
r/blues	48k	26.3	11k	Blues	2.4k	193	1986
r/Rock	44k	25.8	12k	Rock	54.6k	210	1997
r/folk	24k	24.8	8k	Folk	2.8k	214	1996
r/rnb	13k	20.1	4k	RnB	6.6k	332	1993

Table 1: General statistics about Subreddit comments and track lyrics

2 Related Work

Literature on Music Psychology is rich with studies measuring the reactions to different genres of music. A study by Cook et al. (2019) found that reactions to music genres like Pop and Rock were associated with anger and revolt while genres like Classical and Jazz were associated with peacefulness and spirituality. Another study also claimed that 'extreme' music genres like Metal, punk, etc leads to anger and even aggressive behavior (Zalk et al., 2008). On the other hand, several recent studies claimed no such association exists between intense genres of music and aggression (Merz et al. 2020; Sharman and Dingle 2015; Susino and Schubert 2019). Some of these studies suggested that negative associations with a particular style of music might simply be due to low familiarity with the style of music or even due to holding negative stereotypes about the culture related to that music. Given this ambiguity, we approached this research with the intent to explore and uncover possible relationships in naturalistic data. Furthermore, as we focused on online fan communities, we also assumed that users were familiar with that specific genre when posting on a genre's Subreddit.

3 Methods

3.1 Dataset

For all 10 Subreddits, we collected all the comments posted before March 2nd, 2021, using the Pushshift Reddit API (Baumgartner et al., 2020). Table 1 shows the statistical summary about the genre Subreddits. For every Subreddit, we removed the comments that were deleted and were marked '[deleted]', or '[removed]'. We also filtered the comments posted by bots using string matching of phrases like 'I'm a bot'. Finally, the comments for each genre Subreddit were rid of mentions of the particular genre name and its subgenre. This was done to mitigate bias caused by the genre and subgenre names when computing counts for each emotion. The subgenre names were collected and cleaned manually from Wikipedia.

For analysis with lyrics, we first obtained a list of tracks for each genre using the genre annotations provided with the Million Song Dataset[2] called tagtraum genre annotations. Using the Genius API[3], we then retrieved the lyrics of tracks using these track names and artist names. The Genius API returned responses for about $160,000$ tracks. We filtered the noisy responses from the API using string matching for track title and artist name. We finally got lyrics for a total of about $115,000$ tracks. A general statistics of the lyrics collected in this way is given in Table 1.

3.2 Sentiment Analysis

We rely primarily on lexicon-based sentiment analysis methods for analyzing the emotionality of each Subreddit, following similar approaches used by Yinger and Springer (2019). Using the emotional categories from the National Research Council of Canada (NRC) Lexicon (Mohammad and Turney, 2013), we group the Subreddits based on how close the emotional language of people is between the Subreddits using the k-means algorithm. Then, we use decision trees to interpret the clustering produced by k-means. Then, we examine if the prevalence of a type of emotion in the comments of a genre Subreddit is reflective of the prevalence of that type of emotion in the music itself.

We used the NRCLex python library[4] to obtain the counts for emotional words for all the Subreddits. The NRC Lexicon consists of words labeled with one or more of the 8 emotion cate-

[2]http://millionsongdataset.com/
[3]https://docs.genius.com/
[4]https://pypi.org/project/NRCLex/

gories: anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. It has been used for a diverse range of tasks, including in the detection of hate speech (Gao and Huang, 2017), and for studying the emotional development of COVID-19 Twitter communities (Marinov et al., 2020), etc. The python library fuses about additional 17, 000 wordnet-based synonyms to the NRCLex and holds an emotion dictionary of approximately 27, 000 words in total. As counts were larger for Subreddits with more comments, in order to compare the genre Subreddits with subsequent analyses, we normalized the counts of each genre.

3.3 Experiments

The Subreddits were clustered using the k-means algorithm according to the normalized emotion counts. The optimal number of clusters was found to be 2 using the elbow method (Thorndike, 1953). We visualize the clustering results using Principal Component Analysis (PCA) in Figure 1. To interpret our results from the k-means clustering, we used the labels predicted by k-means and trained decision trees. Decision trees was chosen due to their ease of interpretation. An example of the decision tree is shown in Figure 3.

To verify whether the results we obtained were not simply due to the a large number of common users between the Subreddits of a cluster, we also clustered Subreddits based on the number of common users. First, about 4000 unique users were randomly sampled from each of the Subreddits. Then, the number of common users within the 4,000 users was calculated for all the Subreddit pairs. We constructed a fully connected graph with the Subreddits as the nodes and the number of common users as the edge weights. We used Gephi's modularity class function (Bastian et al., 2009) to detect the communities in this graph. The result of the community detection on the common users graph is shown in Figure 2. We also calculated the adjusted rand score to check the similarity of the clustering obtained from the k-means clustering of the normalized emotion counts and the common users graph community detection.

In the second stage of experiments, we examine the correlations between lyrics and Subreddit posts for each emotion category. We obtained normalized counts for the lyrics of each genre in the same way as with the comments. We examined Spearman's rank-order correlation between the prevalence of a particular emotion in the comment with that emotion's prevalence in the lyrics.

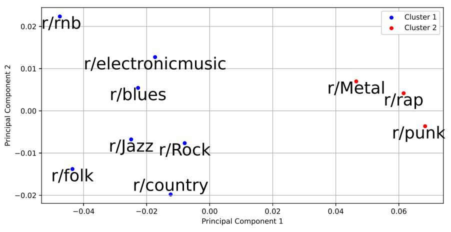


Figure 1: Results of the Subreddits clustering based on the normalized emotion counts. The clustering is visualized using Principal Component Analysis (PCA).

4 Results

With the k-means clustering of the Subreddits, we got two clear clusters with the first cluster having Subreddits of r/blues, r/country, r/electronicmusic, r/folk, r/Jazz, r/Rock, and r/rnb and the second cluster had Subreddits of r/Metal, r/punk, and r/rap. This clustering is also reflected in the PCA visualization as shown in Figure 1. Similarly, as shown in Figure 2, the common users analysis within the Subreddits revealed two communities. One of the communities contains Subreddits of r/rnb, r/rap, r/Metal, and r/electronicmusic and the other community contains Subreddits of r/blues, r/country, r/folk, and r/Jazz, r/Rock, and r/punk. Subreddits in a community have a higher number of common users amongst themselves compared to Subreddits in other communities. Number of communities detected can differ according to the chosen modularity threshold. However, the modularity threshold was left at a default value of 1.0 which resulted in two detected communities. The adjusted rand score of the clustering obtained from the emotional features and the common users analysis was calculated to be 0.07.

Next, we discuss the analysis with decision trees trained using the cluster labels from k-means. We trained 100 different decision trees, all of which yielded trees of depth 1. The decision trees classified the clusters perfectly using only one of 5 emotions each time i.e a single feature had importance of 1.0 each time. These emotions were anger, disgust, fear, joy, and surprise.

Finally, the correlation analysis with the Subreddit comments and the lyrics revealed a strong positive correlation for anger (r=0.818, p=0.004), disgust (r=0.794, p=0.006), and fear(r=0.697,

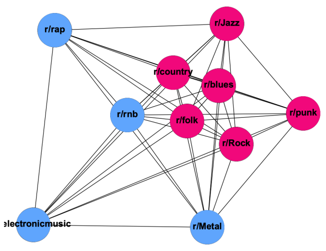


Figure 2: Detection of communities of Subreddits with high number of common users. Force atlas layout is used for arrangement of the nodes. Similar colored nodes belong to the same community.

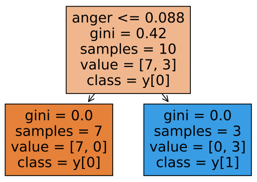


Figure 3: An example of decision tree fit to the clustered Subreddits. This illustrates that the two clusters are clearly distinguished by anger. All Subreddits in cluster 1 have a lower amount of anger words compared to Subreddits in cluster 2.

p=0.025). A positive correlation was also observed for joy emotion (r=0.661, p=0.038). No significant correlations were found for the other emotions. The scripts used for the analysis and full results can be found on our GitHub repository[5].

5 Discussion

Our results provide insight on the difference in how fans of different music genres express themselves. The k-means clustering of the emotion counts yielded two clusters of genre Subreddits. The first cluster is characterized by having a relative abundance of positive-valence emotions i.e. joy, anticipation, and surprise. The second cluster is characterized by the higher prevalence of words expressing negative-valence emotions of anger, disgust, and fear. The adjusted rand score between the clustering obtained from the common users analysis and the clustering found through the k-means clustering normalized emotion counts was close to zero. This signifies that emotional difference in the Subreddit clusters is not simply due to a large number of common users within the Subreddits of the cluster. Rather, the Subreddit clustering hints at a difference in the expressive style of people associated with their music listening habits. Furthermore, we found evidence that a higher degree of high-arousal negative-valence emotions of anger, disgust, and fear expressed in the genre through lyrics is reflected in the fans' emotional expression. These results are in line with previous work by Rubin et al. (2001) that linked fans of metal and rap/hip-hop with trait anger, suggesting that fans of these genres do indeed use more extreme and negative emotional words. However, this does not necessarily disprove research by Merz et al. (2020) that found no correlational or causal link between music genre and psychopathology. One explanation could be due to the specificity of the platform: Subreddit users, in the company of other like-minded users, may express themselves in a manner congruent with the music genre, which may have set the prevailing social norms and standards modeled by the lyrics. Thus, these behaviors may not extend to their daily life beyond the Subreddit. Alternatively, these extreme expressions could function as cathartic release, towards the maintenance of emotional well-being and positive self-regulation Olsen et al. 2020; Sharman and Dingle 2015. This might explain our result, that the prevalence of 'joy' in lyrics of a genre is also reflected with higher amounts of joyful words in the utterances of a genres' Subreddit. However, more research is needed before a definite conclusion can be made, and future research can consider the emotional mechanisms involving genres and fan communities. Our study however has several limitations. We use a single emotion lexicon to quantify the emotions in the comments which can potentially produce biased results. There is a need to validate the results with other emotion lexicons or emotion detection methods. The study also processes comment data only from music listeners on Reddit. Therefore, the findings of this study might also not generalize to the average music listener of a genre. Additionally, we examine the relationship between the emotion expressed by the music and its reflection in the comments using only music with lyrics, without considering the audio component. Therefore, the use of emotion measures computed from the audio could be a direction for future research.

[5]https://github.com/nlp4musa-emotional-language/fan-community-emotion.git

6 References

References

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: An open source software for exploring and manipulating networks. *International AAAI Conference on Weblogs and Social Media*.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. *CoRR*, abs/2001.08435.

Terence Cook, Ashlin R. K. Roy, and Keith M. Welker. 2019. Music as an emotion regulation strategy: An examination of genres of music and their roles in emotion regulation. *Psychology of Music*, 47(1):144–154.

Lei Gao and Ruihong Huang. 2017. Detecting online hate speech using context aware models. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 260–266, Varna, Bulgaria. INCOMA Ltd.

Boris Marinov, Jennifer Spenader, and Tommaso Caselli. 2020. Topic and emotion development among Dutch COVID-19 Twitter communities in the early pandemic. In *Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media*, pages 87–98, Barcelona, Spain (Online). Association for Computational Linguistics.

Zachary C Merz, John W Lace, Thatcher R Coleman, and Robert M Roth. 2020. Challenging the presumptive link between musical preference and aggression. *Psychology of Music*, 0(0):0305735620963756.

Saif Mohammad and Peter Turney. 2013. Nrc emotion lexicon. *National Research Council, Canada, 2*.

Kirk N. Olsen, Merrick Powell, Aydin Anic, Robert J. Vallerand, and William Forde Thompson. 2020. Fans of violent music: The role of passion in positive and negative emotional experience. *Musicae Scientiae*, 0(0):1029864920951611.

Zoe E. Papinczak, Genevieve A. Dingle, Stoyan R. Stoyanov, Leanne Hides, and Oksana Zelenko. 2015. Young people's uses of music for well-being. *Journal of Youth Studies*, 18(9):1119–1134.

Alan M. Rubin, Daniel V. West, and Wendy S. Mitchell. 2001. Differences in aggression, attitudes toward women, and distrust as reflected in popular music preferences. *Media Psychology*, 3(1):25–42.

Suvi Saarikallio. 2011. Music as emotional self-regulation throughout adulthood. *Psychology of Music*, 39(3):307–327.

Leah Sharman and Genevieve A. Dingle. 2015. Extreme metal music and anger processing. *Frontiers in Human Neuroscience*, 9:272.

Marco Susino and Emery Schubert. 2019. Cultural stereotyping of emotional responses to music genre. *Psychology of Music*, 47(3):342–357.

Myriam V. Thoma, Stefan Ryf, Changiz Mohiyeddini, Ulrike Ehlert, and Urs M. Nater. 2012. Emotion regulation through listening to music in everyday situations. *Cognition and Emotion*, 26(3):550–560. PMID: 21902567.

Robert L. Thorndike. 1953. Who belongs in the family. *Psychometrika*, pages 267–276.

Olivia Swedberg Yinger and D Gregory Springer. 2019. Using psycholinguistic inquiry to measure emotional response to music: A feasibility study. *Psychology of Music*, 47(4):606–614.

Maarten Zalk, Marc Delsing, Tom Bogt, and Wim Meeus. 2008. Heavy metal and hip-hop style preferences and externalizing problem behavior: A two-wave longitudinal study. *Youth & Society*, 39.

Marcel Zentner, Didier Grandjean, and Klaus Scherer. 2008. Emotions evoked by the sound of music: Characterization, classification, and measurement. *Emotion (Washington, D.C.)*, 8:494–521.

Atypical Lyrics Completion Considering Musical Audio Signals

Kento Watanabe and **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST)
{kento.watanabe, m.goto}@aist.go.jp

Abstract

This paper addresses the novel task of lyrics completion for creative support. Our proposed task aims to suggest words that are (1) atypical but (2) suitable for musical audio signals. Previous approaches focused on fully automatic lyrics generation tasks using language models that tend to generate frequent phrases, despite the importance of atypicality for creative support. In this study, we propose a novel vector space model and hypothesize that embedding multimodal aspects (words, draft sentences, and music audio) in a unified vector space contributes to capturing (1) the atypicality of words and (2) the relationships between words and the moods of music audio. To test our hypothesis, we used a large-scale dataset to investigate whether the proposed model suggests atypical words.

1 Introduction

Lyrics are important in conveying emotions and messages in popular music, and the recently increasing popularity of user-generated content on video sharing services makes writing lyrics popular even for novice writers. Lyrics writers, however, unlike the writers of prose text, need to create attractive phrases suitable for the given music. Writing lyrics is thus not an easy job.

Its difficulty has motivated a range of studies for automatic lyrics generation (Oliveira et al., 2007; Potash et al., 2015; Watanabe et al., 2018). For example, Watanabe et al. train a Recurrent Neural Network Language Model (RNN-LM) that generates fluent lyrics while maintaining compatibility between the boundaries of lyrics and melody structures. However, even if LMs generate perfect lyrics, a fully automatic generation system cannot support writers because it ignores their intentions.

In this study, for creative support instead of lyrics generation, we design a lyrics completion task that

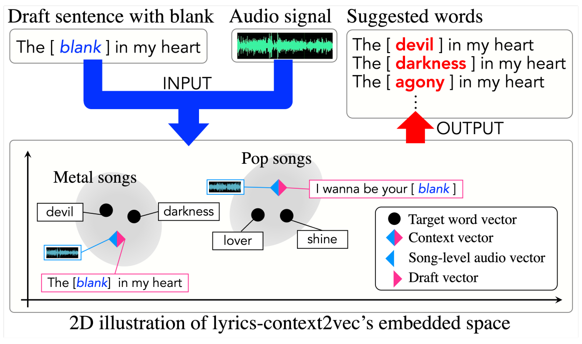

Draft sentence with blank
The [blank] in my heart
Audio signal
Suggested words
The [devil] in my heart
The [darkness] in my heart
The [agony] in my heart
INPUT
OUTPUT
Metal songs
Pop songs
devil
darkness
lover
shine
I wanna be your [blank]
The [blank] in my heart
● Target word vector
Context vector
Song-level audio vector
Draft vector
2D illustration of lyrics-context2vec's embedded space

Figure 1: Overview of lyrics completion task.

recommends candidate words for the blank in a given sentence (Fig. 1). Specifically, we focus on the following two properties of lyrics. (1) Lyrics sometimes depend on the *moods* of music audio (e.g., "death" is often used in metal songs) (Watanabe and Goto, 2019). We propose a task in which a system recommends words suitable for the mood of a given song excerpt represented as an audio signal. (2) *Atypicality* is important in writing lyrics; to make lyrics attractive, writers consider both typical and atypical phrases. However, previous study on lyrics generation has used LMs that predict highly frequent (i.e., *typical*) words (Barbieri et al., 2012; Potash et al., 2015; Watanabe et al., 2017, 2018). Creative support systems need to recommend unusual and rare (i.e., *atypical*) words while maintaining the fluency of the sentence.

We therefore propose a multimodal vector space model (VSM), *lyrics-context2vec*, that, given a draft sentence with a blank, suggests atypical words while maintaining the relationship with the mood of the music audio. With lyrics-context2vec, input vectors (i.e., combinations of music audios and draft sentences) and output vectors (i.e., atypical words) are located near each other in a unified high-dimensional vector space (Fig. 1). This model suggests atypical words because we use typical words as negative examples in its training.

The contributions of this study are summarized as follows: (1) We propose, for creative support, a novel multimodal vector space model that captures the relationship between atypical words and the mood of music audio. (2) We demonstrate that our model suggests words suitable for the mood of the input musical audio signal. (3) We demonstrate that our model suggests words more atypical than those suggested by RNN-LMs.

This paper is a short version of our MMM 2021 paper (Watanabe and Goto, 2021).

2 Lyrics-audio data

To model the relationship between lyrics and moods of music audio, we obtained 458,572 songs, each consisting of a pair comprising a text file of English lyrics and an audio file of a music excerpt[1]. Here each text file contains all sentences of the lyrics of a song, and each audio file is a music excerpt (30 sec) that was collected from the Internet and provided for trial listening. We embedded the moods of audio signals as well as the words of lyrics into a unified vector space without using coarse metadata (e.g., genre tags).

2.1 Bag-of-Audio-Words

To represent the mood feature of a short music excerpt, we use a discrete symbol called an *audio-word* (Liu et al., 2010). The bag-of-audio-words (BoAW) creation procedure is as follows. (1) Each music excerpt is downsampled to 22,050 Hz. (2) *LibROSA*, a python package for music and audio analysis, is used to extract 20-dimensional mel-frequency cepstral coefficients (MFCCs) with the FFT size of 2048 samples and the hop size of 512 samples. This result is represented as an MFCC matrix (20×1280). (3) The MFCC matrix is divided into 128 submatrices (20×10) without overlap. (4) To create a vocabulary of k audio-words, we apply the k-*means*++ algorithm to all the divided MFCCs of all the songs. In other words, each k-th cluster corresponds to an audio-word (aw). In this study we made 3000 audio-words.

3 Atypical word completion model

We propose a multimodal vector space model *lyrics-context2vec* that, given a music audio signal and a draft sentence with a blank, suggests atypical words while maintaining the relationship with the mood of the music audio. Specifically, lyrics-context2vec suggests the best N atypical words $w^1, ..., w^N$ that could fit with the context. Here we assume two types of contexts: (1) the words on the left and right sides of the blank and (2) the BoAW converted from the audio signal.

There are two technical problems in recommending atypical words suitable for the music audio. First, since most LMs learn to predict highly frequent words, it is hard to suggest atypical words that are important for creative support. Second, how to model the relationship between words and musical audio signals is not obvious.

To address the first problem, we focus on the negative sampling strategy in word2vec (Mikolov et al., 2013). This strategy was proposed for the purpose of approximation because computation of loss function is time-consuming. We, however, use negative sampling for the purpose of suppression of typical word recommendation because we want to suggest *atypical* words for creative support. Since negative examples are drawn from the distribution of highly frequent words, it is expected that input vectors of contexts are located far from vectors of typical words. It is not obvious that the negative sampling contributes to suggesting atypical words.

To address the second problem, we utilize the mechanism of *lyrics2vec* proposed by Watanabe and Goto. In lyrics2vec, co-occurring audio-words and lyric words are located near each other under the assumption that *some words of lyrics are written depending on the musical audio signal.*

3.1 Model construction

Lyrics-context2vec is based on lyrics2vec and context2vec (Melamud et al., 2016). Formally, context2vec is a vector space model that encodes left draft words $w_1, ..., w_{t-1}$ and right draft words $w_{t+1}, ..., w_T$ into latent vectors $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, respectively, using two Recurrent Neural Networks (RNNs). Then the target word vector $\boldsymbol{v}(w_t)$ and a vector that is nonlinearly transformed from the latent vectors are mapped closely into a unified vector space. The loss function of context2vec E_{c2v} is defined so that the inner product of the target word vector $\boldsymbol{v}(w_t)$ and the nonlinearly transformed vector is maximized:

$$\begin{aligned} E_{c2v} &= -\log\sigma\Big(\boldsymbol{v}(w_t)^\mathsf{T} \cdot \mathrm{MLP}([\boldsymbol{z}_1, \boldsymbol{z}_2])\Big) \\ &\quad - \sum_{s=1}^{S} \log\sigma\Big(-\boldsymbol{v}(w'_s)^\mathsf{T} \cdot \mathrm{MLP}([\boldsymbol{z}_1, \boldsymbol{z}_2])\Big), \qquad (1) \end{aligned}$$

[1]Lyrics were provided by a lyrics distribution company.

where $\sigma(\cdot)$ is a sigmoid function. To obtain an x-dimensional word vector representation, we define an embedding function $\boldsymbol{v}(\cdot)$ that maps the target word to an x-dimensional vector. S is the number of negative examples w'_s. $[\boldsymbol{z}_1, \boldsymbol{z}_2]$ denotes a concatenation of latent vectors $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. $\mathrm{MLP}(\cdot)$ stands for multilayer perceptron. In this loss function, negative examples w'_s are sampled from the distribution $P(w'_s) = D(w'_s)^{0.75} / \sum_{w' \in V}(D(w')^{0.75})$ where V is the vocabulary and $D(w')$ is the document frequency of a word w'. In other words, since frequent words tend to be sampled as negative examples, we expect that a draft sentence vector and the vector of highly frequent typical words are located far away from each other. When computing word completion, our system displays target words with high cosine similarity to the input context vector $\mathrm{MLP}([\boldsymbol{z}_1, \boldsymbol{z}_2])$.

Then we extend context2vec to suggest atypical words suitable for both the music audio and the draft sentence by embedding three aspects (i.e., target words, draft sentences, and song-level audio). We concatenate song-level audio and draft vectors and define the loss function E so that the concatenated vector $[\boldsymbol{z}_1, \boldsymbol{z}_2, \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{u}(aw_m)]$ is located close to the target word vector $\boldsymbol{v}(w)$:

$$E = -\log\sigma\Big(\boldsymbol{v}(w_t)^{\mathsf{T}} \cdot [\boldsymbol{z}_1, \boldsymbol{z}_2, \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{u}(aw_m)]\Big)$$
$$-\sum_{s=1}^{S}\log\sigma\Big(-\boldsymbol{v}(w'_s)^{\mathsf{T}} \cdot [\boldsymbol{z}_1, \boldsymbol{z}_2, \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{u}(aw_m)]\Big), \quad (2)$$

where we define the dimension of draft vectors $\boldsymbol{z}_1, \boldsymbol{z}_2$ as d and define an embedding function $\boldsymbol{u}(\cdot)$ that maps the context word/audio-word to a d-dimensional vector. M is the number of audio-words in the song. We define the average of audio-word vectors as a song-level audio vector.

4 Experiments

To evaluate whether lyrics-context2vec can suggest (1) atypical words and (2) words suitable for music audio, we designed word completion tasks. The input of these tasks is $T-1$ draft words $w_1, ..., w_{t-1}, w_{t+1}, ..., w_T$ of each sentence in a test song. Therefore the model needs to fill in the t-th blank with a word. We used the following $Score$ to evaluate the performance of models in the lyrics completion task: $Score@N = \sum_{r \in R}\mathbb{1}(r \in \{h^1, ..., h^N\})/|R|$, where r denotes the correct word and $|R|$ is the number of blanks in the test data. $h^1, ..., h^N$ are the top N suggested words. $\mathbb{1}(\cdot)$ is the indicator function. In this study we calculated $Score@N$, with N ranging from 1 to 20 under the assumption that our support system suggests 20 words to users.

Here it is important to define which word in each sentence is the correct word r. We defined four types of correct answers:

Typicality We defined a randomly chosen word in each sentence of the test song as the correct word r. In this metric, high-frequency words tend to be chosen as the correct answer. In other words, this metric is a measure of typical word completion.

Atypicality We first calculated the document frequency of words of the test song and then defined the minimum-document-frequency word in each sentence as the correct word. This metric is a measure of atypical word completion.

Music+Typicality In each sentence of the test song, we extracted the word most similar to the music audio of the song by using the pre-trained lyrics2vec that was proposed by Watanabe and Goto. If the document frequency of the extracted word was more than 1,000, we defined this word as the correct word for the sentence and did not use the other words. This metric is a measure of prediction of *typical* words suitable for the music audio.

Music+Atypicality We extracted the word most similar to the music audio of the song as with Music+Typicality. If the document frequency of the extracted word was less than or equal to 1,000, we defined this word as the correct word for the sentence and did not use the other words. This metric is a measure of prediction of *atypical* words suitable for the music audio of the song.

4.1 Comparison methods

To investigate the effect of our lyrics-context2vec, we compared the following four models. (1) *Bi-RNN-LM*, a bidirectional RNN-LM trained with lyrics without audio. (2) *Encoder-Decoder*, a Bi-RNN-LM in which the song-level audio vector $\frac{1}{M}\sum_{m=1}^{M}\boldsymbol{u}(aw_m)$ is input to the initial RNN state. (3) *Context2vec* (Melamud et al., 2016). (4) *Lyrics-context2vec*, the proposed model. The RNN-LMs (Bi-RNN-LM and Encoder-Decoder) predict words with high predicted probability in the blank, and the VSMs (context2vec and lyrics-context2vec) predict the most similar words in the blank. Examples of words suggested by the models are available at a web page (`https://kentow.github.io/`

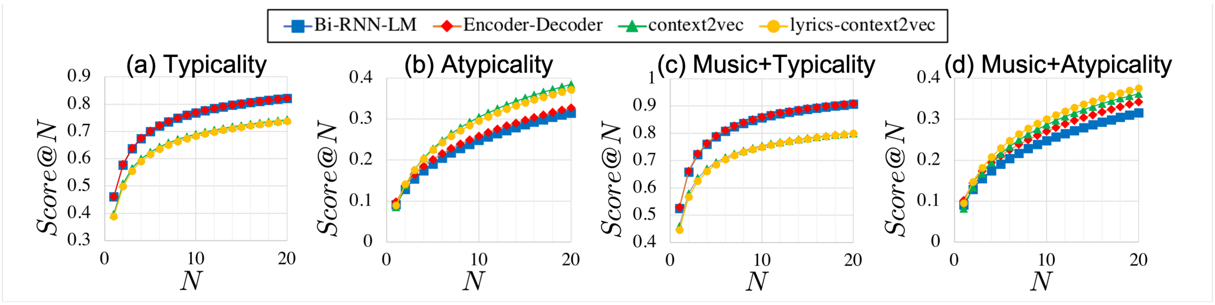


Figure 2: Results of the lyrics completion tasks.

nlp4musa2021/).

4.2 Settings

We randomly split our dataset into 80-10-10% divisions to construct the training, validation, and test data. In all models, we utilized Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) as the RNN layer. We chose $d = 300$ for the dimension of the audio-word vector $\boldsymbol{u}(\cdot)$ and the dimension of the LSTM hidden state $\boldsymbol{z}$. We chose $x = 900$ for the dimension of the target word vector $\boldsymbol{v}(\cdot)$. We used negative sampling with $S = 20$ negative examples. We used Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001 for parameter optimization. The model used for testing was the one that achieved the best Music+Atypicality score on the validation set.

4.3 Results

Figure 2(a) shows the result of the typical word completion task (Typicality). As shown in this figure, RNN-LMs achieved higher scores than VSMs. This is because the RNN-LMs are trained to maximize the probability of generating highly frequent phrases. Interestingly, we can see that there is no difference between the scores of Bi-RNN-LM and Encoder-Decoder. This indicates that audio information does not contribute to predicting typical words. Typical words were thus expected to be correlated with draft sentences rather than audio.

Regarding the task of predicting the typical words suitable for music audio (Fig. 2 (c)), we can observe results similar to those for the task Typicality. This reinforces the fact that typical words can be predicted from only the draft sentence, without using audio information.

Regarding the atypical word completion (Fig. 2 (b)), VSMs achieved higher scores than RNN-LMs. This indicates that negative sampling contributes to suppression of typical word completion. Overall, for atypical word completion tasks it is desirable to use a VSM with negative sampling rather than a LM aimed at generating typical phrases.

Regarding the main task Music+Atypicality (Fig. 2 (d)), lyrics-context2vec predicted atypical words suitable for music audio better than any of the other models. This means that our model captures both the atypicality and the relationship between a music audio and words simultaneously. Moreover, we can see that lyrics-context2vec performs better than context2vec and that Encoder-Decoder performs better than Bi-RNN-LM. This indicates that using audio information contributes to suggesting atypical words suitable for the music audio.

5 Conclusion

We proposed lyrics-context2vec, a multimodal vector space model that suggests atypical but appropriate words for the given music audio and draft sentence. In the vector space of lyrics-context2vec, a vector corresponding to an atypical word in a song and a song-level audio vector corresponding to an audio excerpt of the song are located near each other. We trained the models to suggest atypical words by embedding the highly frequent word vector away from the song-level audio vector.

In the experiment, we used a large-scale dataset to investigate whether the proposed model suggests atypical but appropriate lyrics. Several findings were obtained from experiment results. One is that the negative sampling contributes to suggesting atypical words. Another is that embedding audio signals contributes to suggesting words suitable for the mood of the music audio. We conclude that embedding multiple aspects into a vector space contributes to capturing atypicality and relationship with audio.

Acknowledgements. The authors appreciate SyncPower Corporation for providing lyrics data. This work was supported in part by JST ACCEL Grant Number JPMJAC1602, JST CREST Grant Number JPMJCR20D4, and JSPS KAKENHI Grant Number 20K19878, Japan.

References

Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. 2012. Markov constraints for generating lyrics with style. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 115–120.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).*

Yang Liu, Wan-Lei Zhao, Chong-Wah Ngo, Chang-Sheng Xu, and Han-Qing Lu. 2010. Coherent bag-of audio words model for efficient large-scale video copy detection. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 89–96.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, pages 3111–3119.

Hugo R. Gonçalo Oliveira, F. Amialcar Cardoso, and Francisco C. Pereira. 2007. Tra-la-lyrics: an approach to generate text based on rhythm. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*, pages 47–55.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. GhostWriter: Using an LSTM for automatic Rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1924.

Kento Watanabe and Masataka Goto. 2019. Query-by-Blending: A music exploration system blending latent vector representations of lyric word, song audio, and artist. In *Proceedings of the 20th annual conference of the International Society for Music Information Retrieval*, pages 144–151.

Kento Watanabe and Masataka Goto. 2021. Atypical lyrics completion considering musical audio signals. In *Proceedings of the 27th International Conference on Multimedia Modeling*, volume 12572, pages 174–186.

Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. 2018. A melody-conditioned lyrics language model. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 163–172.

Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, Tomoyasu Nakano, Satoru Fukayama, and Masataka Goto. 2017. LyriSys: An interactive support system for writing lyrics based on topic transition. In *Proceedings of the 22nd Annual Meeting of the Intelligent User Interfaces Community*, page 559—563.