Lyrics Segmentation: Textual Macrostructure Detection using Convolutions

Michael Fell[♣], Yaroslav Nechaev[♦], Elena Cabrio[♥], Fabien Gandon[♠]
Université Côte d'Azur, CNRS, Inria, I3S, France[♣♥♠]
Fondazione Bruno Kessler, University of Trento[♦]
michael.fell@unice.fr[♣], nechaev@fbk.eu[♦]
elena.cabrio@unice.fr[♥], fabien.gandon@inria.fr[♠]

Abstract

Lyrics contain repeated patterns that are correlated with the repetitions found in the music they accompany. Repetitions in song texts have been shown to enable lyrics segmentation – a fundamental prerequisite of automatically detecting the building blocks (e.g. chorus, verse) of a song text. In this article we improve on the state-of-the-art in lyrics segmentation by applying a convolutional neural network to the task, and experiment with novel features as a step towards deeper macrostructure detection of lyrics.

Title and Abstract in French

Segmenter les paroles de chansons:
détection par réseau de neurones convolutif d'une macrostructure textuelle

Les paroles de chansons contiennent des passages qui se répètent et sont corrélés aux répétitions trouvé dans la musique qui les accompagne. Ces répétitions dans les textes de chansons ont montré leur utilité pour la segmentation des paroles qui est une étape préalable fondamentale dans la détection automatique des blocs de construction d'une chanson (ex. le refrain, les couplets). Dans cet article, nous améliorons l'état de l'art de la segmentation des paroles en concevant un réseau de neurones convolutif pour cette tâche et expérimentons de nouvelles caractéristiques pour aller vers une détection plus profonde de la macrostructure des paroles.

1 Introduction

Among the seas of textual resources available online are the lyrics of songs that are very popular resources for professional, cultural and social applications. To support intelligent interactions with these resources (e.g. browsing, visualizing, synchronizing) one of the first needs is to access the structure of the lyrics (e.g. intro, verse, chorus). However, lyrics are essentially provided as flat text files without structural markup. In this article, we address the problem of textual macrostructure detection in lyrics.

Lyrics encode an important part of the semantics of a song and a motivating scenario for this work is to improve structural clues that can be used by search engines handling large collections of lyrics. Ideally we would like to be able to detect these structures reliably in different music genres to support new search criteria such as "find all the songs where the chorus talks about freedom".

As a first step, structure segmentation could serve as a front end processor for music content analysis, since it enables a local description of each section rather than a coarse, global representation of the whole song (Cheng et al., 2009; Casey et al., 2008). Music structure discovery is a research field in Music Information Retrieval where the goal is to automatically estimate the temporal structure of a music track by analyzing the characteristics of its audio signal over time. However only a few works have addressed such task lyrics-wise (P. G. Mahedero et al., 2005; Watanabe et al., 2016; Baratè et al., 2013). Given that lyrics contain rich information about the semantic structure of a song, relying on textual features could

Proceedings of the 27th International Conference on Computational Linguistics, pages 2044–2054
Santa Fe, New Mexico, USA, August 20-26, 2018.

help in overcoming the existing difficulties associated with large acoustic variation in music. Moreover, a complete music search engine should support search criteria exploiting both the audio and the textual dimensions of a song.

In this direction, this paper focuses on the following research question: *given the text of the lyrics, can we learn to detect the lines delimiting segments in the song?* This question is broken down into two sub questions: *which features are the most relevant to detect that a line delimits a segment?* and *which classification method is the most efficient to detect that a line delimits a segment?*

In order to infer such lyrics structure, we introduce a neural network-based model that *i)* efficiently exploits the Self-Similarity Matrix representations (SSM) used in the state-of-the-art (Watanabe et al., 2016), *ii)* can utilize traditional features alongside the SSMs, and *iii)* jointly infers the structure of the entire text. An evaluation on two standard datasets of English lyrics (Music Lyrics Database V.1.2.7[1], and the WASABI corpus (Meseguer-Brocal et al., 2017)), shows that our proposed method can effectively detect the boundaries of music segments outperforming the state of the art, and is portable across collections of song lyrics of heterogeneous musical genre.

Generally speaking, structure segmentation consists of two stages: a text segmentation stage that divides lyrics into segments, and a semantic labeling stage that labels each segment with a structure type (e.g. intro, verse, chorus). Although a few works have addressed the task of finding chorus or repeated parts in music (P. G. Mahedero et al., 2005; Baratè et al., 2013), full song text segmentation remains challenging unless some complexity reduction strategies are applied (as selecting a subset of songs belonging to musical genres characterized by repeating patterns, e.g. Country or Pop songs). As the accuracy of lyrics segmentation in state of the art is not fully satisfying yet (Watanabe et al., 2016), and given the variability in the set of structure types provided in the literature according to different genres (Tagg, 1982; Brackett, 1995), rare attempts have been made to achieve semantic labeling. Therefore, in our work, we focus on improving the state of the art in lyrics segmentation, leaving the task of semantic labeling for future work.

Earlier in this section, we presented our research questions and motivation. In the remainder of the paper, in Section 2 we define the task of classifying lines as segment borders, and in Section 3 we detail the features used to represent the lines and the classification methods we selected for the task. Section 4 presents the experiment we conducted and compares the results obtained by the different models and configurations to perform the segment border classification. In Section 5 we position our work in the current state of the art, and in Section 6 we conclude with future research directions to provide ever more metadata to music information retrieval systems.

2 Segmentation task definition

Figure 1 shows a song text and its segmentation into the segments A, B, C, D, E, as given by the annotation of the text. As a Pop song, this example lyrics has a fairly common structure which can be described as: *Verse 1-Chorus-Verse 2-Chorus-Outro*. The reasoning behind this structure analysis is that perfectly repeating parts usually correspond to the chorus. Hence, B and D should both be a chorus. A and C are verses, as they lead to the chorus and there is no visible bridge. The last segment, E, repeats the end of the chorus and is very short, so it can be classified as an outro. While the previous analysis appears plausible, it relies on world knowledge, such that the chorus is the most repeated part, a verse usually leads into a chorus (optionally via a bridge), and an outro ends a song text, but is optional.[2]

Labelling the segments of a song text is a non-trivial task that requires diverse knowledge. As explained in the introduction, in this work we focus on the first task of segmenting the text. This first step is fundamental to segment labelling when segment borders are not known. Even when segment borders are "indicated" by line breaks in lyrics available online, those line breaks have usually been annotated by users and neither are they necessarily identical to those intended by the songwriter, nor do users in general agree on where to put them. Thus, a method to automatically segment unsegmented song texts is needed to automate that first step. Many heuristics can be imagined to find the segment borders. For

[1] `http://www.odditysoftware.com/page-datasales1.htm` (retrieved 11/24/17)

[2] For more details on the set of structure types, we refer the reader to (Tagg, 1982; Brackett, 1995).

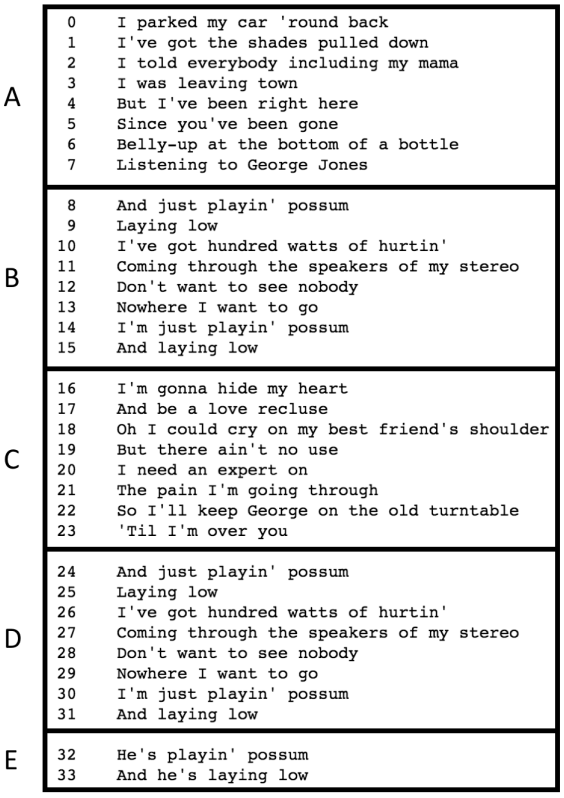

Figure 1: Segment structure of a Pop song ("Don't Rock The Jukebox" by A. Jackson, MLDB-ID: 2954)

example, separating the lines into segments of consistent lengths, which in our example gives 8-8-8-8-2 lines. Another heuristic could be to never start a segment with a conjunction. But as we can see, this rule does not hold for our example, as the chorus starts with the conjunction "And". This is to say that enumerating heuristic rules can be an open-ended task. Among previous works in the literature on lyrics structure analysis, (Watanabe et al., 2016) heavily exploited repeated patterns present in the lyrics to address this task, and it shows that this general class of pattern is very helpful with segment border detection.

For this reason in this paper we follow (Watanabe et al., 2016) by casting the segmentation task as binary classification task. Each line of a song text is defined as either ending a segment or not. Let L_s be the lyrics of a song s composed of n lines of text: $L_s = \{l_1, l_2, ..., l_n\}$. Let $seg \subseteq (L_s, \mathbb{B})$ be a function that returns for each line $l_i \in L_s$ if it is the end of a segment. We define the segmentation task as learning a classifier that approximates seg. At the learning stage, the segment borders are observed from segmented text as double line breaks. At the testing stage the classifier has to predict the now hidden segment borders.

3 Modelling segments in song texts

In order to infer the lyrics structure, we propose a neural network-based model that *i)* efficiently relies on repeated patterns in a song text that are conveyed by the self-similarity matrix representations (SSM) used in the state-of-the-art (Watanabe et al., 2016), *ii)* can utilize traditional features alongside the SSMs (e.g., n-grams and characters count) and *iii)* jointly infers the structure of the entire text.

In the following, Section 3.1 describes the similarity measures that we have tested to produce the SSM representations, and Section 3.2 explains how such SSMs are produced. Section 3.3 lists the set of features used by the neural network-based model, and Sections 3.4 and 3.5 describe the model itself.

3.1 Similarity measures

The choice of the similarity measure is particularly important. Different measures and their combinations have been explored, for example, in musicology (Cohen-Hadria and Peeters, 2017). For our model, we produce SSMs based on three line-based similarity measures:

1. *String similarity* ($\text{sim}_{\text{string}}$): a normalized Levenshtein string edit distance between the characters of two lines (Levenshtein, 1966).

2. *Phonetic similarity* (sim_{phon}): a simplified phonetic representation of the lines computed using the "Double Metaphone Search Algorithm" (Philips, 2000). When applied to "i love you very much" and "i'l off you vary match" it returns the same result: "ALFFRMX". This algorithm was developed to capture the similarity of similar sounding words even with possibly very dissimilar orthography. After translating the words into this "phonetic language", we compute the character-wise edit distance like with string similarity.

3. *Lexico-structural similarity* ($\text{sim}_{\text{lex-struct}}$): this measure, initially proposed in (Fell, 2014), combines lexical with syntactical similarity. $\text{sim}_{\text{lex-struct}}$ captures the similarity between text lines such as "Look into my eyes" and "I look into your eyes": these are partially similar on a lexical level and partially similar on a syntactical level. Given two lines x, y lexico-structural similarity is defined as: $sim_{lex-struct}(x, y) = sim^2_{lex}(x, y) + (1 - sim_{lex}) \cdot sim_{struct}(\hat{x}, \hat{y})$, where sim_{lex} is the overlap of the bigrams of words in x and y, and sim_{struct} is the overlap of the bigrams of pos tags in $\hat{x}, \hat{y}$, the remaining tokens that did not overlap on a word level.

3.2 Self-similarity Matrices

We start by constructing features for each target lyrics. First, the line-based self-similarity matrices are produced to capture repeated patterns in a text. Such representations have been previously used in the literature to estimate the structure of music (Foote, 2000; Cohen-Hadria and Peeters, 2017) and lyrics (Watanabe et al., 2016). Given a text with n lines, a self-similarity matrix $SSM_M \in \mathbb{R}^{n \times n}$ is constructed, where each element is set by computing a similarity measure between the two corresponding lines $(SSM_M)_{ij} = sim_M(l_i, l_j)$, where sim_M is one of the similarity methods we defined. As a result, SSMs highlight distinct blocks of the target text revealing the underlying structure.

Figure 2 shows an example of lyrics consisting of five segments (left) along with its $\text{SSM}_{\text{string}}$ encoding (right). The segment borders are indicated by green lines. The repeated patterns in the song text are revealed by its SSM. This is illustrated as the song text (left) is nicely aligned to the SSM (right).

There are two common patterns that were investigated in the literature: *diagonals* and *rectangles*. Diagonals parallel to the main diagonal indicate text sequences that repeat and are typically found in a chorus. Rectangles, on the other hand, indicate text sequences in which all the lines are highly similar to one another. Both of these patterns were found to be indicators of segment borders.

3.3 Final features

After the SSMs were produced, each possible segment border was represented by the surrounding lines in a context window. Formally, given a fixed window size w_{size}, for each line l_i the submatrix (or patch) of the SSM was selected: $P_i = SSM_M[i - w_{\text{size}} + 1, ..., i + w_{\text{size}}; *] \in \mathbb{R}^{2w_{\text{size}} \times d}$. The resulting submatrices are the SSM features used for classification ($\text{SSM}_{\text{string}}$, SSM_{phon}, and $\text{SSM}_{\text{lex-struct}}$).

In addition to similarity matrices, we extracted the character count from each line, a simple proxy of the orthographic shape of the song text. Intuitively, segments that belong together tend to have similar shapes. Finally, we extracted n-grams (similar to (Watanabe et al., 2016)'s term features) from each line that were most indicative for segment borders, using the tf-idf weighting. We extracted n-grams that are typically found left or right from the border, varied n-gram lengths and also included indicative PoS tag n-grams. This resulted in 240 term features in total. The most indicative words at the start of a segment were: {ok, lately, okay, yo, excuse, dear, well, hey}. As segment-initial phrases we found: {Been a long, I've been, There's a, Won't you, Na na na, Hey, hey}. Typical words ending a segment were: {..., ..,

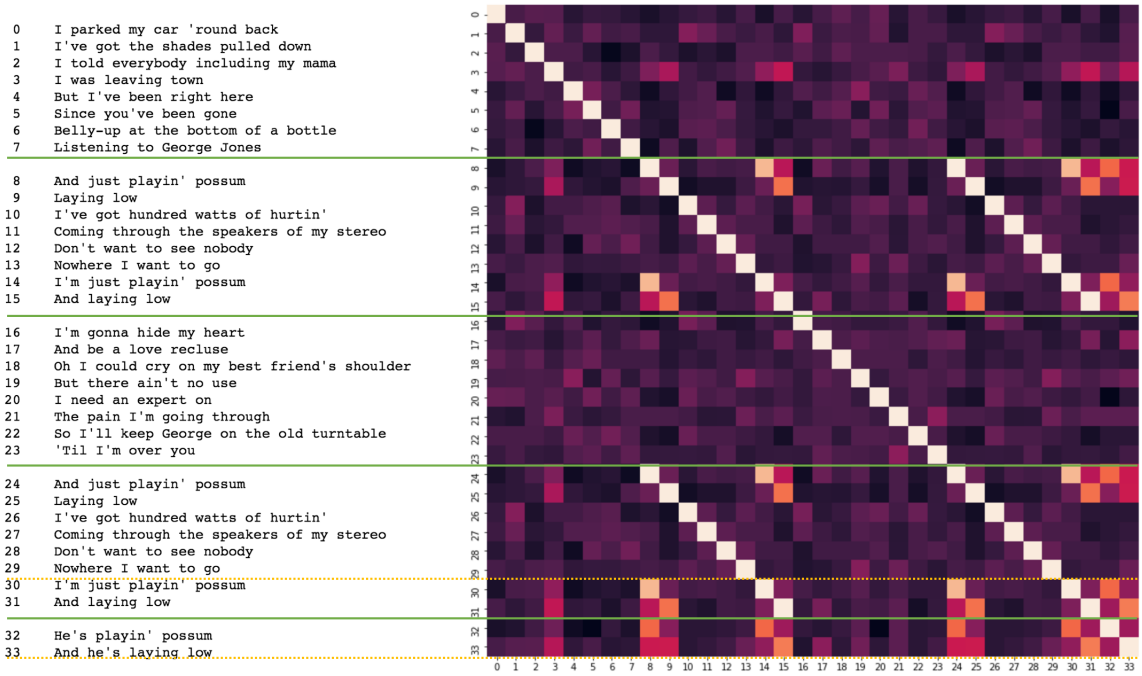

Figure 2: Green lines indicate the true segment borders. Diagonal stripe patterns from (8,24) to (15,31) co-occur well with the borders of the true segment from line 8 to line 15. Note, there is also a large square pattern from (30,30) to (33,33) indicated by the yellow lines - this is a highly repetitive musical state, but not a true segment. This is a typical false positive segment. ("Don't Rock The Jukebox" by Alan Jackson, MLDB-ID: 2954)

!, ., yeah, ohh, woah. c'mon, wonderland}. And as segment-final phrases we found as most indicative: {yeah!, come on!, love you., !!!, to you., with you., check it out, at all., let's go, ...}

3.4 Convolutional Neural Network-based Model

Figure 3 outlines our approach. The goal of the model is to predict if the segment border should be placed after the given line in a text. So, for each line, the model receives patches extracted from the precomputed SSMs: $\text{input}_i = \{P_i^1, P_i^2, ..., P_i^c\} \in \mathbb{R}^{2w_{\text{size}} \times d \times c}$, where c is the number of SSMs or number of channels.

Then, the input goes through a series of convolutions. Convolutional layers (Goodfellow et al., 2016) have been extensively used in image processing to allow the neural network to extract translation, scaling, and rotation invariant features anywhere on the input image. We employ the same motivation here: segment borders manifest themselves in the form of distinct patterns in the SSM. Thus convolutions allow the network to capture those patterns efficiently regardless of their location and relative size.

The first convolution uses a filter of size $(w_{\text{size}}+1) \times (w_{\text{size}}+1)$ so that each feature extracted captures a prospective segment border. The following max pooling layer downsamples the resulting tensor by w_{size} on both dimensions. The second convolution has a filter of size $1 \times w_{\text{size}}$ and the second max pooling layer further downsamples each feature to a scalar. Convolutions employ the ReLU function for activation. After the convolutions, the resulting feature vector is concatenated with the line-based features described above and goes through a series of densely connected layers with $tanh$ as the activation function. Then the $softmax$ is applied to produce probabilities for each class (border/not border). The model is trained using Adam (Kingma and Ba, 2014) with the cross-entropy as an objective function.

Finally, after repeating the procedure for each line in the text and picking the most probable class, we acquire the inferred structure for the entire text. Note that the usage of the self-similarity matrix as input allows us to make predictions based on the entire text: each input patch contains similarity scores between the lines surrounding the provisional segment border and all others. As shown in Section 4, CNN-based models significantly outperform the state-of-the-art.

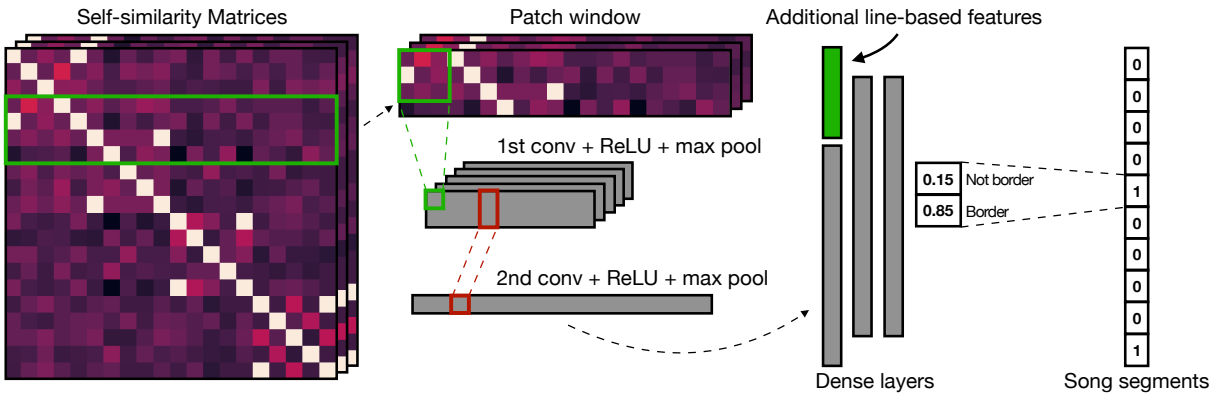

Figure 3: Convolutional Neural Network-based model inferring text structure.

3.5 Joint Predictions via a Recurrent Layer

Following the CNN-based model, we investigate the possibility of making segment border predictions jointly for the entire input text instead of producing labels for a single line at the time. The idea is to refine predictions for the target line using previous predictions. With this in mind, instead of producing the label directly, dense layers are wired through an additional recurrent layer.

Recurrent layers (Goodfellow et al., 2016) consist of cells containing a state that is able to carry information from the previous predictions for the same song and combine it with the current input to produce a refined output. We use LSTM cells and $tanh$ as an activation function. Each line is fed consecutively through the convolutional layers into the RNN cell, which outputs class probabilities.

4 Experimental setting

This section describes the experiments carried out to perform the segment border classification. First, we describe the different models and configurations that we have investigated (Section 4.1), the datasets on which we have run our method evaluation (Section 4.2), and then we discuss the obtained results (Section 4.3).

4.1 Models and configurations

We compare to the state-of-the-art (Watanabe et al., 2016) and successfully reproduce their best features to validate their approach. Two groups of features are used in the replication: repeated pattern features (RPF) extracted from SSMs and n-grams extracted from text lines.

Then, our own models are neural networks (CNNs and RNNs) that use as features SSMs, n-grams (as described in Section 3.3), and character count. We index the SSM features according to the similarity they embed (see Section 3.1): $\text{SSM}_{\text{string}}$ embeds the string similarity, SSM_{phon} the phonetic similarity and $\text{SSM}_{\text{lex-struct}}$ the lexico-structural similarity. The feature set SSM_{all} means that all SSMs are used in parallel. For convolutional layers we empirically set $w_{\text{size}} = 2$ and the amount of features extracted after each convolution to 128. Dense layers have 512 hidden units, the size of the LSTM hidden state is 25. We have also tuned the learning rate (negative degrees of 10), the dropout probability with increments of 0.1. The batch size was selected from the beginning to be 256 to better saturate our GPU. Both the CNN and RNN models were implemented using Tensorflow.

For comparison, we implement two baselines, i.e. the random baseline (as defined in (Watanabe et al., 2016)), and a logistic regression that uses the character count of each line as feature. For the simple character count baseline we used sklearn.LogisticRegression with the option class_weight='balanced' to account for highly imbalanced data.

4.2 Datasets

Song texts are available widely across the Web in the form of user-generated content. Unfortunately for research purposes, there is no comprehensive publicly available online resource that would allow a more standardized evaluation of research results. This is mostly attributable to copyright limitations and has been criticized before in (Mayer and Rauber, 2011). Research therefore is usually undertaken on corpora that were created using standard web-crawling techniques by the respective researchers. Due to the user-generated nature of song texts on the Web, such crawled data is potentially noisy and heterogeneous, e.g. the way in which line repetitions are annotated can range from verbatim duplication to something like *Chorus (4x)* to indicate a triple repetition of a chorus.

To compare our approach with the state of the art, as a first corpus we selected the English part of the Music Lyrics Database V.1.2.7, a proprietary lyrics corpus previously used in (Watanabe et al., 2016). We call this corpus henceforth MLDB. Like (Watanabe et al., 2016) we only consider those song texts that have five or more segments. We use the same training, development and test indices, which is a 60%-20%-20% split. In total we have 102802 song texts with at least 5 segments. 92% of the remaining song texts count between 6 and 12 segments. Evaluation metrics are precision, recall, and f-score.

To test the system portability to bigger and more heterogeneous data sources, we selected the WASABI corpus[3] (Meseguer-Brocal et al., 2017), which is a larger corpus of song texts (henceforth called WASABI). This dataset contains 743939 English song texts with at least 5 segments, and for each song it provides the following information: its lyrics[4], the synchronized lyrics when available[5], DBpedia abstracts and categories the song belongs to, genre, label, writer, release date, awards, producers, artist and/or band members, the stereo audio track from Deezer, when available, the unmixed audio tracks of the song, its ISRC, bpm, and duration. Moreover, we aligned MLDB to WASABI as the latter provides genre information. Song texts that had the exact same title and artist names (ignoring case) in both data sets were aligned. This rather strict filter resulted in an amount of 58567 (57%) song texts with genre information in MLDB. Table 2 shows the distribution of the genres in MLDB song texts. Thanks to the alignment with WASABI, we were able to group MLDB lyrics according to their genre. We then tested our method on each subset separately, to verify our intuition that classification is harder for some genres in which almost no repeated patterns can be detected (as Rap songs). To the best of our knowledge, previous work did not report on genre-specific results.

In this work we did not normalize the lyrics in order to rigorously compare our results to (Watanabe et al., 2016). We estimate the proportion of lyrics containing tags such as *Chorus* to be marginal (0.1-0.5%) in the MLDB corpus. When applying our methods for lyrics segmentation to lyrics found online, an appropriate normalization method should be applied as a pre-processing step. For details on such a normalization procedure we refer the reader to (Fell, 2014), Section 2.1.

4.3 Results and discussion

Table 1 shows the results of our experiments on the MLDB dataset. We start by measuring the performance of our replication of (Watanabe et al., 2016)'s approach. This reimplementation exhibits 56.3% F_1, similar to the results reported in the original paper. The divergence could be attributed to a different choice of hyperparameters and feature extraction code. Much weaker baselines were explored as well. The random baseline resulted in 13.3% F_1, while the usage of simple line-based features, such as character count, improves this to 25.4%.

The best CNN-based model, SSM_{all}+n-grams, outperforms all our baselines reaching 67.4% F_1, 8.2% better than the results reported in (Watanabe et al., 2016). We perform an approximate randomization test of this model against all other models reported below. In every case, the performance difference is statistically significant ($p < .05$). Subsequent feature analysis revealed that the $\text{SSM}_{\text{string}}$ is by far the best individual SSM. The $\text{SSM}_{\text{lex-struct}}$ feature exhibits much lower performance, despite being a

[3] https://wasabi.i3s.unice.fr/

[4] Extracted from http://lyrics.wikia.com/

[5] From http://usdb.animux.de

Model	*Configuration*	*Precision[%]*	*Recall[%]*	*F_1[%]*
Baselines	Random	14.3	12.5	13.3
	Char count (CC)	16.7	52.8	25.4
(Watanabe et al., 2016)	RPF (replication)	48.2	67.8	56.3
	RPF	56.1	59.4	57.7
	RPF + n-grams	57.4	61.2	59.2
CNN	SSM_{string} + CC	70.4	63.0	66.5
	SSM_{phon} + CC	75.9	55.6	64.2
	$SSM_{lex\text{-}struct}$ + CC	74.8	50.0	59.9
	SSM_{all} + CC	74.1	60.5	66.6
	SSM_{all} + n-grams	72.1	63.3	**67.4**
RNN	SSM_{string} + CC	69.8	59.6	64.3
	SSM_{phon} + CC	65.3	63.1	64.2
	$SSM_{lex\text{-}struct}$ + CC	72.8	51.4	60.3
	SSM_{all} + CC	68.3	63.2	65.6
	SSM_{all} + n-grams	71.7	61.8	66.4

Table 1: Performances obtained by the different models and configurations on MLDB test set.

much more complex feature. We believe the lexico-structural similarity is much noisier as it relies on n-grams and PoS tags, and thus propagates error from the tokenizers and PoS taggers. The SSM_{phon} exhibits a small but measurable performance decrease from SSM_{string}, possibly due to phonetic features capturing similar regularities, while also depending on the quality of preprocessing tools and the rule-based phonetic algorithm being relevant for our song-based dataset. However, despite lower individual performance, SSMs are still able to complement each other with the SSM_{all} model yielding the best performance.

In addition, we test the performance of several line-based features on our dataset. Most notably, the n-grams feature provides a significant performance improvement producing the best model. The CC feature shows marginal improvement.

Finally, the best RNN-based model exhibits 66.4% F_1. As mentioned above, the motivation to use this model was to improve predictions on the subsequent lines in the text by leveraging previous predictions. We did not observe an improvement over the CNN-based model most likely due to SSMs already accommodating enough global information so that convolutional layers are able to utilize it for prediction efficiently. As a result, the introduction of an additional recurrent layer increased the complexity of the model without providing much of a benefit. However, we believe that the addition of a more substantial amount of local features could make this additional complexity relevant.

Results differ significantly based on genre. In Table 2 we report the performances of the SSM_{string} on subsets of songs divided by their musical genre. The model is trained on the whole MLDB dataset and separately tested on each subset. Songs belonging to genres such as Country, Rock or Pop, contain recurrent structures with repeating patterns, which are more easily detectable by the CNN-based algorithm. Therefore, they show significantly better performance. On the other hand, the performance on genres such as Hip Hop or Rap, is much worse. An SSM for a Rap song is depicted in Figure 4. As texts in this genre are less repetitive, the SSM-based features are becoming much less reliable to determine a song's structure.

To show the portability of our method to bigger and more heterogeneous datasets, we ran the CNN model on the WASABI dataset, obtaining the following results (very close to the ones obtained on MLDB dataset): precision: 67.4%, recall: 67.3%, f-score: 67.4% using the SSM_{string} features.

5 Related Work

Besides the work of (Watanabe et al., 2016) that we have discussed in detail in Section 3, only a few papers in the literature have focused on the automated detection of the structure of lyrics. (Mahedero et al., 2005) report experiments on the use of standard NLP tools for the analysis of music lyrics. Among the tasks they address, for structure extraction they focus on lyrics having a clearly recognizable structure (which is not always the case) divided into segments. Such segments are weighted following the results given by descriptors used (as full length text, relative position of a segment in the song, segment simi-

Genre	*Lyrics[#]*	*Precision[%]*	*Recall[%]*	F_1*[%]*
Rock	6011	73.8	57.7	64.8
Hip Hop	5493	71.7	43.6	54.2
Pop	4764	73.1	61.5	66.6
RnB	4565	71.8	60.3	65.6
Alternative Rock	4325	76.8	60.9	67.9
Country	3780	74.5	66.4	**70.2**
Hard Rock	2286	76.2	61.4	67.7
Pop Rock	2150	73.3	59.6	65.8
Indie Rock	1568	80.6	55.5	65.6
Heavy Metal	1364	79.1	52.1	63.0
Southern Hip Hop	940	73.6	34.8	47.0
Punk Rock	939	80.7	63.2	**70.9**
Alternative Metal	872	77.3	61.3	68.5
Pop Punk	739	77.3	68.7	**72.7**
Soul	603	70.9	57.0	63.0
Gangsta Rap	435	73.6	35.2	47.7

Table 2: SSM_{string} model performances across musical genres in the MLDB dataset. Underlined are the performances on genres with less repetitive text. Genres with highly repetitive structure are in bold.

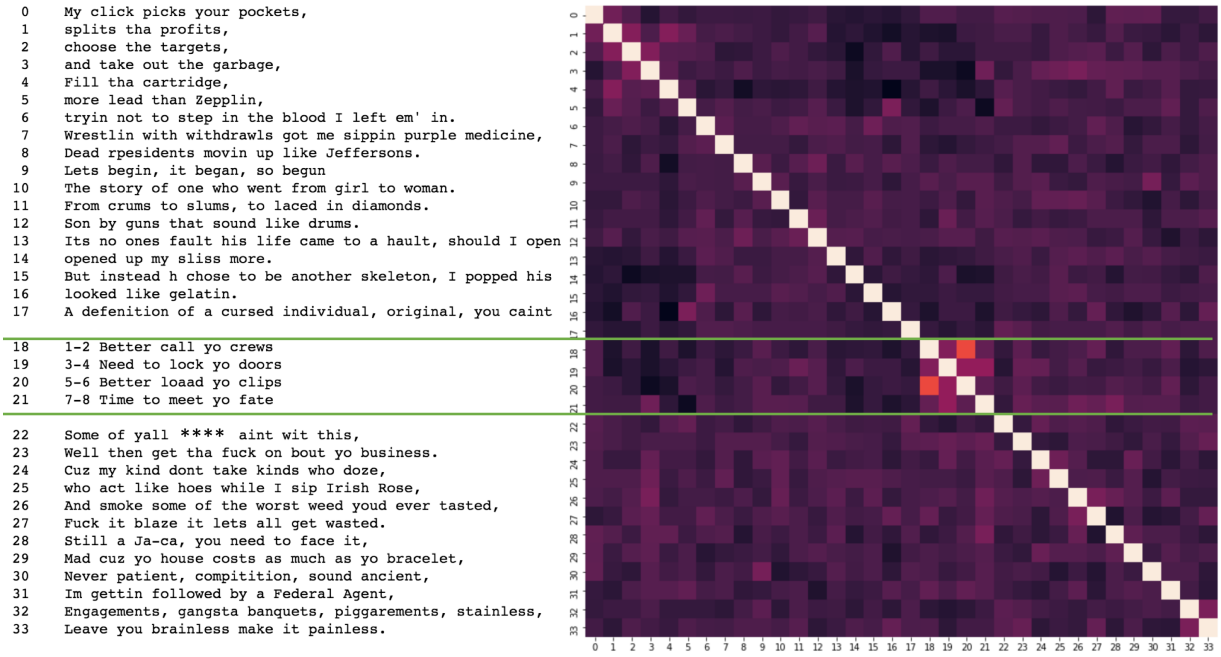

Figure 4: As common for Rap song texts, this example does not have a chorus (diagonal stripes). However, there is a highly repetitive musical state from line 18 to 21 which can be recognized by inspecting the SSM the from line 1 to line 4 is indicated by the corresponding rectangle in the SSM spanning from (18,18) to (21,21). ("Meet Your Fate" by Southpark Mexican, MLDB-ID: 125521)

larity), and then tagged with a label describing them (e.g. chorus, verses). They test the segmentation algorithm on a small dataset of 30 lyrics, 6 for each language (English, French, German, Spanish and Italian), which had previously been manually segmented.

More recently, (Baratè et al., 2013) describe a semantics-driven approach to the automatic segmentation of song lyrics, and mainly focus on pop/rock music. Their goal is not to label a set of lines in a given way (e.g. verse, chorus), but rather identifying recurrent as well as non-recurrent groups of lines. They propose a rule-based method to estimate such structure labels of segmented lyrics, while in our approach we apply ML methods to unsegmented lyrics.

To enhance the accuracy of audio segmentation, (Cheng et al., 2009) propose a new framework utiliz-

ing both audio and lyrics information for structure segmentation. For audio segmentation, the proposed constrained clustering algorithm improves the accuracy of boundary detection by introducing constraints on neighboring and global information. For semantic labeling, they derive the semantic structure of songs by lyrics processing to improve structure labeling.

With the goal of identifying repeated musical parts in music audio signals to estimate music structure boundaries (lyrics are not considered), (Cohen-Hadria and Peeters, 2017) propose to feed Convolutional Neural Network with the square-sub-matrices centered on the main diagonals of several self-similarity-matrices (SSM), each one representing a different audio descriptors, building their work on (Foote, 2000).

Lastly, the discourse segmentation literature heavily relies on segments to reflect topics and on segment borders to indicate topic changes. However, the segments in lyrics tend to be short and lyrics tend not to change topic a lot compared to longer prose texts. We experimented with word embeddings to measure topical similarity, but did not find it useful in our experiments. Still, integrating a domain-independent segmentation algorithm into our approach - such as the divisive clustering found in (Choi, 2000) - might be beneficial.

6 Conclusion and Future Work

Lyrics encode an important part of the semantics of a song and relying on their textual features could help us in overcoming the difficulties in audio processing for music management systems. Moreover the support for search criteria exploiting both the audio and the textual dimensions of a song would improve the usefulness of music search engine and retrieval systems by providing more and combinable selection and filtering dimensions.

In this paper, we focused on improving the state of the art in lyrics segmentation which is a prerequisite before considering the semantic labeling of these segments. We defined the task of classifying lyrics lines as segment borders or not, and we detailed the features we considered to feed to the classification methods. Having identified several feature configurations and state of the art classification methods we conducted an experiment and compared performance on the task of segment border classification. The best result (67.4 % f-score) was obtained using the Convolutional Neural Network with a self-similarity-matrix with layers combining all the similarity metrics and, in addition, the n-gram features. Moreover, at this stage, the CNN models outperformed the state of the art models using hand-crafted features[6].

For future work, we plan to complement this analysis with acoustic and cultural metadata (coupling our work for instance with (Cohen-Hadria and Peeters, 2017)), in the direction of developing a comprehensive music information retrieval systems over a real-world song collection.

Acknowledgements

The work described in this paper is carried out in the framework of the WASABi project, funded by the French National Research Agency (ANR-16-CE23-0017-01). Moreover, we want to thank the anonymous reviewers for their insightful comments.

References

A. Baratè, L. A. Ludovico, and E. Santucci. 2013. A semantics-driven approach to lyrics segmentation. In *2013 8th International Workshop on Semantic and Social Media Adaptation and Personalization*, pages 73–79, Dec.

D. Brackett. 1995. *Interpreting Popular Music*. Cambridge University Press.

M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. 2008. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, April.

H. T. Cheng, Y. H. Yang, Y. C. Lin, and H. H. Chen. 2009. Multimodal structure segmentation and analysis of music using audio and textual information. In *2009 IEEE International Symposium on Circuits and Systems*, pages 1677–1680, May.

[6]To allow for experimental reproducibility, the code is available at `https://github.com/TuringTrain/lyrics_segmentation`

Freddy YY Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33. Association for Computational Linguistics.

Alice Cohen-Hadria and Geoffroy Peeters. 2017. Music Structure Boundaries Estimation Using Multiple Self-Similarity Matrices as Input Depth of Convolutional Neural Networks. In *AES International Conference Semantic Audio 2017*, Erlangen, Germany, June.

Michael Fell. 2014. Lyrics classification. Master's thesis, Saarland University, Germany.

Jonathan Foote. 2000. Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 1, pages 452–455. IEEE.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. 12.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Jose P. G. Mahedero, Álvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. 2005. Natural language processing of lyrics. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, pages 475–478, New York, NY, USA. ACM.

Rudolf Mayer and Andreas Rauber. 2011. Musical genre classification by ensembles of audio and lyrics features. In *Proceedings of the 12th International Conference on Music Information Retrieval*, pages 675–680.

Gabriel Meseguer-Brocal, Geoffroy Peeters, Guillaume Pellerin, Michel Buffa, Elena Cabrio, Catherine Faron Zucker, Alain Giboin, Isabelle Mirbel, Romain Hennequin, Manuel Moussallam, Francesco Piccoli, and Thomas Fillon. 2017. WASABI: a Two Million Song Database Project with Audio and Cultural Metadata plus WebAudio enhanced Client Applications. In *Web Audio Conference 2017 – Collaborative Audio #WAC2017*, London, United Kingdom, August. Queen Mary University of London.

Jose P. G. Mahedero, Alvaro Martinez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. 2005. Natural language processing of lyrics. pages 475–478, 01.

Lawrence Philips. 2000. The double metaphone search algorithm. 18:38–43, 06.

Philip Tagg. 1982. Analysing popular music: theory, method and practice. *Popular Music*, 2:37–67.

Kento Watanabe, Yuichiroh Matsubayashi, Naho Orita, Naoaki Okazaki, Kentaro Inui, Satoru Fukayama, Tomoyasu Nakano, Jordan Smith, and Masataka Goto. 2016. Modeling discourse segments in lyrics using repeated patterns. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1959–1969.