Expert Systems with Applications 163 (2021) 113749

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Explicit song lyrics detection with subword-enriched word embeddings

Marco Rospocher

Universitá degli Studi di Verona, Lungadige Porta Vittoria 41, 37129 Verona, Italy

ARTICLE INFO

Article history:
Received 30 April 2020
Revised 30 June 2020
Accepted 11 July 2020
Available online 28 July 2020

Keywords:
Word embeddings
Text classification
Explicit content detection

ABSTRACT

In this paper, we investigate the problem of automatically detecting explicit song lyrics, i.e., determining if the lyrics of a given song could be offensive or unsuitable for children. The problem can be framed as a binary classification task, and in this work we propose to tackle it with the fastText classifier, an efficient linear classification model leveraging a peculiar distributional text representation that, by exploiting subword information in building the embeddings of the words, enables to cope with words not seen at training time. We assess the performance of the fastText classifier and word representations with a lyrics dataset of over 800K songs, annotated with explicit information, that we assembled from publicly available resources. The evaluation shows that the fastText classifier is effective for explicit lyrics detection, substantially outperforming a reference approach for the task, and that the subword information effectively contributes to this result.

1. Introduction

Over the last 40 years, several initiatives and measures have addressed the problem of preventing the exposure of children to musical content that could be offensive or unsuitable for them. Music industry organizations, such as the Recording Industry Association of America (RIAA) or the British Phonographic Industry (BPI), have introduced marks (e.g., the RIAA Parental Advisory Label – PAL) to alert adults that the content of a given CD or LP, mainly lyrics but also related merchandising, is not appropriate for young people because of strong language, references to violence, physical, or mental abuse, references to sexual behaviour, discriminatory language, and so on.[1]

With the advent of digital music and streaming content providers (e.g., iTunes, Amazon Music, Spotify, Deezer), and the possibility to listen to/buy single songs (independently of the CD, if any, they are packaged in), it became important for online platforms to mark – typically with a tag such as "explict", "explict lyrics", or "E" – songs whose lyrics are offensive or unsuitable for children, so that their reproduction is prohibited if appropriate filters are in place. This explicit content information is typically provided to these platforms directly by right-holders, as in the case of Spotify,[2] or by "record labels, industry partners and our users", as in the case of Deezer.[3] However, as it is mainly a manual, time-consuming activity to be performed on a huge amount of content,[4] platforms themselves acknowledge that they "cannot guarantee all explicit content is marked as such".[5]

Therefore, the music industry would greatly benefit from systems automatically tagging explicit songs, and in the last couple of years few approaches were proposed, exploiting methods such as dictionary lookup (Chin, Kim, Kim, Shin, & Yi, 2018; Kim & Yi, 2019; Fell, Cabrio, Corazza, & Gandon, 2019) or machine/deep learning techniques (Kim & Yi, 2019; Fell et al., 2019). Interestingly enough, as experimentally shown in (Fell et al., 2019), advanced deep learning techniques do not perform much better on this task than dictionary lookup methods or basic machine learning classifiers, such as logistic regression, the latter being considered a strong baseline in terms of performance and computational costs (i.e., efficiency of training, no need for GPU architecture, etc.).

In line with these considerations, in this work we assess the performance of the fastText classifier (Joulin, Grave, Bojanowski, & Mikolov, 2017) for the task of explicit lyrics detection. The fastText classifier is an efficient linear model that achieves state-of-the-art performances (on par with deep learning classifiers, but orders of magnitude faster – c.f., Joulin et al., 2017) on various classification tasks (e.g., sentiment analysis, tag prediction). fastText exploits a peculiar distributional text representation (bag of character *n*-grams), effective in dealing with words not seen at training

E-mail address: marco.rospocher@univr.it

[1] C.f., https://support.imusiciandigital.com/article/154-what-is-considered-as-explicit-content.

[2] https://support.spotify.com/us/using_spotify/system_settings/explicit-content/.

[3] https://support.deezer.com/hc/en-gb/articles/360000590898-Explicit-content.

[4] Spotify shares over 50 million tracks - c.f. https://newsroom.spotify.com/company-info/.

[5] https://support.spotify.com/us/using_spotify/system_settings/explicit-content/.

https://doi.org/10.1016/j.eswa.2020.113749
0957-4174/

time, a situation that may likely occur with lyrics, where new words, slang, variations (e.g., word compounding) or shortenings tend to be frequently used.

The contributions of this paper is threefold. First, we show that the fastText classifier achieves state-of-the-art performance for the task of explicit lyrics detection, substantially outperforming a strong baseline as logistic regression trained with a BOW text representation (Fell et al., 2019). Second, we empirically show that the bag of character *n*-grams strategy effectively contributes to improve classification performance for the considered task. Third, both assessments were conducted on a large dataset of over 800 K English song lyrics – that we make available[6] to favor reproducibility, further experiments and comparisons – providing solid findings for a task that was previously evaluated on smaller datasets from 20K to 180K resources.

The paper is organized as follows. Section 2 overviews the relevant related work on explicit lyrics detection. Section 3 recalls the main characteristics of the fastText classifier and distributional representation model. Section 4 presents the conducted performance assessment, describing the evaluation dataset, methodology, and results. Section 5 discusses some of the findings of the evaluation and other related assessments, such as the application of dataset balancing techniques and the adoption of a more advanced classifier on the fastText distributional text representation, while Section 6 concludes.

2. Related work

The detection of explicit content in text and music has recently received growing attention by the scientific community. While in this work we focus on the detection of explicit song lyrics, some very recent work (Vaglio, Hennequin, Moussallam, Richard, & d'Alché-Buc, 2020) has investigated the audio-based detection of explicit content in music. The approach exploits an audio-to-character recognition model, performing (dictionary-based) keyword spotting on singing voice, and inferring content explicitness by a binary classifier (Random Forest) over the detected keywords.

Focusing on text, the detection of offensive content with Natural Language Processing (NLP) techniques is a quite active research field, especially in social media. Davidson, Warmsley, Macy, and Weber (2017) focused on hate speech detection, i.e., recognizing "'language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group". A Logistic Regression classifier is used to detect tweets into three categories: hate speech, offensive language (i.e., using offensive terms but without hatred), or neither. The system achieves overall high scores, though it performs substantially worse on the hate speech class (40% of hate speech is misclassified). Corazza, Menini, Cabrio, Tonelli, and Villata (2020) recently proposed a modular neural architecture for hate speech detection that works for different languages (i.e., English, Italian, and German), exploiting various features of different type (e.g., textual features, social network features, emotion lexica), including fastText word embeddings. In Carta. et al. (2019), the authors proposed a supervised multi-class multi-label word embeddings to classificy toxic comments (e.g., personal attacks, verbal bullying). The approach is evaluated on a dataset of comments taken from the Wikipedia's talk page, showing that the usage of sets of word embeddings allows to outperform the canonical BOW model.

NLP techniques have been previously applied to lyrics, in order to complement the traditional analyses of acoustic and cultural song metadata. Mahedero, Martínez, Cano, Koppenberger, and Gouyon (2005) evaluated the application of NLP to song lyrics for tasks such as language identification, structure (e.g., verse, chorus, bridge) extraction, topic (e.g., love, drugs) classification, and similarity search (i.e., finding similar songs/ artists).

A few recent approaches have addressed more specifically the task considered in our work, i.e., automatically detecting explicit lyrics via NLP. Among the first works, Chin et al. (2018) tackled the problem of detecting explicit lyrics for Korean songs. For this study, a corpus of 27,695 Korean song lyrics, 1024 marked as explicit (3.7% of the total), was used. First, the authors proposed a baseline for detecting explicit content when the song lyrics include words in a profanity language dictionary (provided by Namu-wiki, a Korean wiki),[7] achieving a (macro) F_1 score of 0.61 (0.88 for the weighted variant, wF_1).[8] Then, they experimented with different machine learning classification algorithms (AdaBoost, Bagging (Bloehdorn & Hotho, 2006)), achieving the best scores with Bagging ($F_1 = 0.78$, $wF_1 = 0.96$).

Bergelid (2018) tested various machine learning algorithms – linear Support Vector Machine (LSVM) (Cortes & Vapnik, 1995), Multinomial Naive Bayes (MNB) (Kibriya, Frank, Pfahringer, & Holmes, 2004), k-Nearest Neighbors (KNN) (Fix et al., 1951), Random Forest (RF) (Rokach, 2010) – on a corpus of 25,441 English song lyrics, 3310 marked as explicit (13.0% of the total). The algorithms were fed with features extracted using TF-IDF (Sparck Jones, 1988) vectorization (a BOW representation, based on word frequencies) and Doc2Vec (Le & Mikolov, 2014) vectorization (a distributional representation exploiting also paragraph information). The best score ($F_1 = 0.677$) is achieved using the TF-IDF vectorization with LSVM and MNB. A substantial performance boost ($F_1 = 0.826$) was obtained with RF by under sampling the dataset, i.e., randomly removing non-explicit lyrics from the dataset in order to have the same amount of explicit and non-explicit lyrics, a configuration which however is rather unrealistic in practice (non-explicit lyrics are many more than explicit ones). A slight improvement is also obtained by extending the feature set with information beyond the lyrics textual content, such as artist name or music energy level.

Kim and Yi (2019) contributed to the explicit lyrics detection task experimenting with a corpus of 70,077 Korean song lyrics, 7,468 of which marked as explicit (10.7% of the total). First, they proposed to automatically build a dictionary of explicit words, basically comparing (and thus weighting differently) the specific usage of words in explicit lyrics against non-explicit ones, and to classify a lyrics as explicit if it contains any word in the generated lexicon. This lexicon-based filtering strategy enables to achieve 0.756 as F_1 score. Then, they tested the usage of Hierarchical Attention Networks (HAN) (Yang et al., 2016), an RNN-based model for sequential and hierarchical processing of words, obtaining 0.766 as F_1 score. However, they also showed that combining the vector representation of the lyrics obtained with HAN with a vector representation according to the occurrence of words from the automatically generated lexicon, an F_1 score of 0.805 can be achieved.

Finally, Fell et al. (2019) compared various machine and deep learning algorithms for explicit lyrics classification. The work exploited the largest dataset consider so far for the task, consisting of 179,391 English song lyrics, 17,808 marked as explicit (9.9% of the total). Different methods were compared: dictionary-based ones (best F_1 score of 0.785), a logistic regression classifier fed with TF-IDF BOW vector representations ($F_1 = 0.780$), BERT Language Model (Devlin, Chang, Lee, & Toutanova, 2019) ($F_1 = 0.777$), and Textual Deconvolution Saliency (Vanni, Ducoffe, Aguilar, Precioso, & Mayaffre, 2018) ($F_1 = 0.796$), a Convolutional Neural Network

[6] Due to licensing issue, only the URLs of the songs composing the dataset are provided, which can be accessed to obtain the actual lyrics and annotations.

[7] https://namu.wiki/.

[8] The definition of the metrics here reported is presented in details in Section 4.2.

for text classification. As observed in the paper, deep models do not outperform the other shallow approaches. Later on, as part of a follow-up work (Fell, Cabrio, Korfed, Buffa, & Gandon, 2020), a logistic regression classifier fed with TF-IDF BOW vector representations was released, pre-trained on 438K song lyrics, reporting an F_1 score of $= 0.773$.

3. fastText classifier

fastText (Bojanowski, Grave, Joulin, & Mikolov, 2017; Joulin et al., 2017) is a library developed by the Facebook AI Research group for efficient learning of word representations and text classification. Differently from other popular approaches learning word representations, that typically assign a single, distinct vector to each word, in fastText each word is represented as a bag of character *n*-grams (a.k.a., *subwords*) (Bojanowski et al., 2017). More precisely, each word (e.g., "lyrics") is represented as a set of character *n*-grams (e.g., with $n = 3$: "<ly", "lyr", "yri", "ric", "ics", "yr>") and the whole word itself (i.e., "<lyrics>").[9] Then, a vector representation is associated to each *n*-gram, and each word is actually codified as the sum of the vector representations of its *n*-gram. The intuition behind this approach is that subwords allow taking into account the internal structure of the words when building the representation, thus capturing also morphological aspects of the words, and hence enabling to better deal with languages with large vocabularies and rare words. Indeed, fastText allows computing word representations also for words never seen during the learning phase of the representation. The size *n* of the *n*-grams to consider in building the word representations is controlled by two parameters of the approach (*minn* and *maxn*), which constrain the range of size for the subwords, and can be tuned according to the foreseen application.

The fastText method for learning these word representations enables to efficiently train models on large corpora with traditional CPUs, without requiring a GPU or other advanced architectures, and to achieve state-of-the-art performances on tasks such as word similarity and analogy, in different languages (Bojanowski et al., 2017).

Building on this approach for learning word representations, an efficient document classifier was later proposed (Joulin et al., 2017). The classifier works as follow. First, a document vector is obtained by averaging the fastText vector representations (i.e., with subwords) of all the words in the document. The document vectors are then fed as features to a linear classifier (multinomial logistic regression) to perform the classification task. To improve efficiency, a hierarchical softmax can be selected to compute the probability distribution over the predefined classes considered in the task, and thus estimating the most probable class for a given text. The fastText classifier was evaluated on various classification tasks (e.g., sentiment analysis, tag prediction) showing results that are on par with advanced deep learning models in terms of accuracy, but much faster (orders of magnitude) for training and evaluation.

Our intuition is that the subword information exploited by the fastText vector representations and classifier may be particularly effective in the context of song lyrics, where new words, slang, variations (e.g., word compounding, shortenings) starting from existing word, including strong language terms, tend to be frequently used. To the best of our knowledge, the performances of the fastText vector representations and classifier have not been previous assessed for the explicit lyrics detection task. Furthermore, we experiment and compare the approach against relevant baselines, on the largest dataset (807,707 English song lyrics) considered so far for the task. Next section presents such assessment.

[9] Special boundary symbols <, > are used to distinguish prefixes and suffixes from other character sequences.

Table 1
Main characteristics of the song lyrics dataset used in the evaluation.

Sources	LyricWiki, Spotify
Year Coverage	1950–2019
Language	English
# Lyrics	807,707
# Explicit Lyrics	62,549
% Explicit Lyrics	7.74

4. Evaluation

The evaluation we conduct aims at understanding the potential of fastText vector representations and classifier for the task of detecting song lyrics with explicit content. First we describe the dataset and evaluation method that we use. All evaluation material (including dataset, evaluation splits, code, and classifiers predictions) is made available at https://bit.ly/ft-explyr.[10]

4.1. Dataset

To evaluate the proposed solution for explicit lyrics detection, we need a dataset consisting of song lyrics annotated with explicitness information, i.e., whether the lyrics contain explicit content or not. To build such dataset (to the best of our knowledge, no dataset from previous works is ready available and distributed) we rely on content provided through public platforms, namely LyricWiki and Spotify. More precisely, we start from a recent dump of LyricWiki made available through Internet Archive.[11] We process the XML dump to select only the pages containing song lyrics (i.e., having the tag <lyrics>), thus excluding pages about artists or albums. From these pages, we extract the content of the <lyrics> element, together with the metadata information about the song (from the header and footer template of the page), such as artist, title, year, language, and the links to other resources, such as iTunes, Spotify, Youtube, etc. We automatically pre-process the text of the lyrics removing possible tags (e.g., HTML code) or annotations (e.g., wiki markup denoting song sections), and dropping songs with empty lyrics (e.g., among them, instrumental tracks). Furthermore, of all the resulting song lyrics, we keep only the English ones, and having a corresponding Spotify ID. Exploiting the Spotify API, we retrieve the corresponding track information from Spotify, and in particular the "explicit" boolean value (set to 1 in case the song contains explicit content, 0 otherwise). We thus obtain a dataset with the characteristics reported in Table 1. To the best of our knowledge, this is the largest dataset on which experiments on explicit song lyrics detection have ever been conducted. As expected, the dataset is unbalanced, as non explicit lyrics are many more than explicit ones: note that the percentage of explicit lyrics in the dataset is 7.74, slightly lower than the one for the dataset used in Bergelid (2018), Kim and Yi (2019) and Fell et al. (2019) and slightly higher than the one for the dataset used in Chin et al. (2018).

The dataset is used both for training and testing purposes, as explained in the next sections.

4.2. Research questions and evaluation measures

In our evaluation, we address the following research questions:

RQ1 Does the fastText classifier achieve state-of-the-art performance for explicit lyrics detection?

[10] For lyrics, we can only made available Spotify song IDs and corresponding LyricWiki page URLs, due to licensing issues.

[11] https://archive.org/download/wiki-lyricsfandomcom/lyricsfandomcom-20200216-history.xml.7z.

RQ2 Does the subword information exploited in the fastText word representations positively contribute to the task of explicit lyrics detection?

To address these research questions, given a set of lyrics gold-annotated with the explicit/non-explicit information, we compare the prediction of the fastText classifier on those lyrics against gold annotations computing standard metrics for classification tasks, such as precision, recall, and F_1 score. More in details, for each classification class i (1 = explicit/0 = non-explicit), we compute the following class metrics:

- class i true positives (TP^i): the number of lyrics predicted in class i that are in class i according to the gold standard;
- class i false positives (FP^i): the number of lyrics predicted in class i that are in the other class according to the gold standard;
- class i false negatives (FN^i): the number of lyrics that are in class i according to the gold standard, but are predicted in the other class;
- $P^i = \frac{TP^i}{TP^i+FP^i}$, $R^i = \frac{TP^i}{TP^i+FN^i}$ and $F_1^i = \frac{2 \cdot P^i \cdot R^i}{P^i+R^i}$.

To combine the performance on the two classes, we then compute the following aggregated metrics:

- accuracy (A): the ratio between the correctly predicted classes, i.e., $TP^0 + TP^1$ divided over the test set size;[12]
- macro-averaged metrics (P, R, F_1): average of the corresponding metrics for the two classes;
- weight-averaged metrics (wP, wR, wF_1): same as the macro averaged, but with each metric scaled according to the relative number of examples in that class.

All the metrics are computed using the Python scikit-learn classification_report method.[13]

4.3. Evaluation procedure

To address the research questions, we compare the performance of the following systems:

- Majority: a classifier that always predicts the majority class (i.e., non-explicit) for any given lyrics;
- WASABI: the pre-trained explicit lyrics classifier discussed in Fell et al. (2020): i.e., a logistic regression classifier, where the word features are weighted with the well-known TF-IDF scheme[14];
- LR-BOW: the same classifier as WASABI, but trained on the dataset considered in this paper;
- fastText: the fastText classifier, trained on the dataset considered in this paper, where word representations exploits subword information;
- $\text{fastText}_{\text{nos}}$: the fastText classifier, trained on the dataset considered in this paper, where word representations are built without using subword information.

We decided to compare the fastText classifier with these baselines for several reasons. First, all these systems run on CPUs, without requiring any advanced GPU architecture, so making the performances reproducible on any commodity hardware available. Second, while other approaches were previously investigated in the literature (see Section 2), to the best of our knowledge, WASABI is the only implemented system for explicit lyrics detection that is available and downloadable. Third, we explicitly compare with LR-BOW as in previous works (Fell et al., 2019; Bergelid, 2018) it emerged as a strong baseline for explicit lyrics detection in terms of performance and computational costs (i.e., efficiency of training, no need for GPU architecture, etc.), showing performance substantially on par with more advanced approaches (e.g., BERT, CNN).

Given the large size of the evaluation dataset, we adopt the standard Pareto 80-20 split between training and testing, using a further 20% portion of the training part for optimizing the parameters of the classifiers.

As a result of the optimization phase at training time, we set the following values for the parameters:

- $\text{LR} - \text{BOW}: \ C = 1.0, \ \ max\_iter = 2000, \ penalty = l2, \ \ solver = lbfgs, \ tol = 0.0001, \ max\_features = 100,000$;
- $\text{fastText}_{\text{nos}}$: $lr = 0.042, epoch = 40, wordNgrams = 4, dim = 65, loss = hs, minn = 0, maxn = 0$;
- fastText: $lr = 0.042, epoch = 40, wordNgrams = 4, dim = 65, loss = hs, minn = 4, maxn = 6$.

Note that for the fastText classifiers, the optimal training performance is obtained with vector representations of relatively small dimension (65), and, for the variant that exploit subword information, considering n-grams of length between 4 and 6.

4.4. Evaluation results

Table 2 reports the performance of the various systems considered for assessing research question RQ1. We recall that the reported scores were obtained on a test set of 161,542 annotated song lyrics.

The results shows that fastText substantially outperforms all other systems on all F_1 scores reported (both on singles class and averaged). Concerning the other systems, LR-BOW performs slightly better than the pre-trained WASABI. The scores for WASABI are slightly higher than the ones reported in Fell et al. (2019) ($F_1 = 0.797$ here, $F_1 = 0.780$ in Fell et al. (2019)); the same applies also for the majority class baseline ($F_1 = 0.480$ here, $F_1 = 0.474$ in Fell et al. (2019)), and this is due to the explicit lyrics ratio being slightly lower in our dataset.

The improvement of the fastText classifier over LR-BOW is substantial $\left(F_1^0 = +0.015, F_1^1 = +0.073, A = +0.026, F_1 = +0.044, wF_1 = +0.019\right)$, and the difference between the two systems is statistically significant as confirmed by the McNemar's test[15] (typically used for assessing the statistical significance of system comparisons in classification tasks) with a p-value smaller than 0.01. Similar scores and trends were also observed running 10-fold cross validation of the systems on the whole 807,707 lyrics dataset.

A big part of the improvement of the fastText classifier over LR-BOW is due to the better performance on the explicit class, which being substantially smaller than the other, it is also the hardest to predict, as confirmed by the lower absolute scores with respect to the non-explicit class. Interestingly, we can observe a substantial difference between fastText and the models using the TF-IDF weighting scheme: fastText achieves a substantially higher P^1 than R^1, while the contrary holds for the other models.

[12] In a binary classification task like the one here considered, this value corresponds also to the micro-averaged precision, recall, and F_1 score.

[13] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html.

[14] Code downloaded from:https://github.com/micbuffa/WasabiDataset.

[15] https://en.wikipedia.org/wiki/McNemar%27s_test.

Table 2
Performance scores of the various systems considered for assessing research question RQ1. Highest F_1/A scores are marked in bold. For the Majority system, being P^1 and F_1^1 not computable, they are set to.000 by default by the scorer.

System	Non-explicit			Explicit			A	Macro			Weighted		
	P^0	R^0	F_1^0	P^1	R^1	F_1^1		P	R	F_1	wP	wR	wF_1
Majority	.923	1.000	.960	.000	.000	.000	.923	.461	.500	.480	.851	.923	.885
WASABI	.984	.935	.959	.517	.822	.635	.927	.751	.879	.797	.948	.927	.934
LR-BOW	.981	.953	.967	.582	.782	.668	.940	.782	.868	.817	.950	.940	.944
fastText	.970	.994	**.982**	.897	.631	**.741**	**.966**	.933	.812	**.861**	.964	.966	**.963**

The substantial improvements over LR-BOW, a strong baseline for explicit lyrics detection, confirm the capability of the fastText approach to achieve state-of-the-art performance also on detecting explicit lyrics, thus allowing us to positively answer RQ1.

To understand the impact of the subword information on the explicit lyrics classification task (RQ2), we also run the variant of fastText not exploiting subword information (fastText$_{nos}$) on the same evaluation dataset.

Table 3 reports side-by-side the performance of both variants of fastText, together with the score differences between the one exploiting subword information and the other variant. fastText outperforms fastText$_{nos}$ on all metrics considered, both single classes and averaged. The improvements are substantially higher on the explicit class, suggesting that subword information has a relevant impact on predicting lyrics in this class. Again, the difference between the two systems is statistically significant as confirmed by the McNemar's test with a p-value smaller than 0.01. These experimental results show that the subword information exploited in the fastText word representations enables to improve the performance on the task of explicit lyrics detection, thus allowing us to positively answer RQ2.

5. Discussion and additional findings

Qualitative analysis of the prediction To better analyze how the fastText classifier works for the explicit lyrics detection task, we had a closer look at the predictions produced on the evaluation test set.

First, we had a look at a sample of the lyrics where the fastText classifier predicted the wrong class. Roughly 20% of these cases are lyrics predicted as explicit which are instead non-explicit, while the remaining 80% of the cases are the opposite situation. In some cases the classifier was not able to properly handle the textual content of the lyrics: for instance, we observed a few cases of very long lyrics, containing a single strong language word, predicted as non-explicit while gold-labeled as explicit in the dataset. We also noted a handful of cases where the explicit annotations in the dataset do not seem to be properly assigned to the songs: for instance, songs labeled as non-explicit while containing strong language (and predicted as explicit by fastText) as well lyrics marked as explicit but where no explicitness could be actually found (and predicted as non-explicit by fastText). While the latter may be songs that have incorrectly inherited the explicit label from the album they are in (e.g., *Be Thankful* by Nate Dogg),[16] in the former cases it is likely that the explicit label is wrongly missing (e.g., *Under Attack* by Impaled Nazarene).[17]

While clearly not ideal for properly assessing the performance of the approach, these cases are even more problematic if found in the training data. We recall Spotify publicly acknowledges that not all explicit lyrics are guaranteed to be marked as explicit. To assess the impact of these cases on the whole evaluation dataset (i.e., both training and testing parts), we searched for non-explicit lyrics containing occurrences of strong language words,[18] finding 1075 (potentially) problematic lyrics in the whole dataset (i.e., out of 807,707 lyrics). By removing them and retraining the fastText classifier, a not-negligible improvement of performance can be achieved (a macro $F_1 = 0.877, +0.016$ wrt. the same measure in Table 2); a similar improvement can be obtained if labeling these lyrics as explicit (a macro $F_1 = 0.878, +0.017$ wrt. the same measure in Table 2). This further confirms that, if trained with good quality annotated data, the fastText classifier may effectively detect explicit song lyrics.

Next, we had a look at cases where the fastText classifier correctly predicted explicit lyrics, but the other systems (e.g., LR-BOW) failed to do so. Here we noticed cases where the subword information of the peculiar fastText word vectorization helps detecting strong language when a known offensive word appears as part of a closed compound word, unlikely to be seen "as is" in the training data (c.f., *Intro 2 Tha Hood* by Blaze Ya Dead Homie).[19]

In other interesting cases, the fastText classifier was able to detect explicit lyrics that, even if not using typical strong language words, they reference violence or physical abuse, like "[..] it don't mean nothing to kill a beautiful girl [..] I wanna watch as you die [..]" (*Nothing* by Cancerslug),[20] or "Eaten alive, Munched into pulp, [..]" (*Excreted Alive* by Carcass).[21] These examples suggest that the fastText classifier and word representations can also cope with lyrics where the offensiveness of the content is due to the way the words are used, rather than the strong language itself.

Time performance Although the main goal of our evaluation was to assess the performance in term of *effectiveness* of the considered systems, we briefly report also some data on their *efficiency* in terms of running time. More precisely, we measured[22] the running time of LR-BOW, fastText$_{nos}$ and fastText both for training the models (more than 500 K songs) and testing them (more than 160K songs):

- training: fastText$_{nos}$ (552s), LR-BOW (560s), fastText (1308s)
- testing: fastText$_{nos}$ (38s), LR-BOW (67s), fastText (69s)

Given the size of the dataset, all systems proved to be quite efficient both in training (a little bit less than 22 min as worst time) and testing (a little bit more than one minute as worst time). The

[16] Metadata: https://open.spotify.com/track/64c6atpCZcM7xMrv1qyrRh, lyrics: https://lyrics.fandom.com/wiki/Nate_Dogg:Be_Thankful.

[17] Metadata: https://open.spotify.com/track/5STln6fDYClZlOobLWfAyO, lyrics: https://lyrics.fandom.com/wiki/Impaled_Nazarene:Under_Attack.

[18] As dictionary, we used the strong and strongest words listed in a 2016 Ofcom report on the usage of offensive language – https://www.ofcom.org.uk/__data/assets/pdf_file/0022/91624/OfcomOffensiveLanguage.pdf.

[19] Metadata: https://open.spotify.com/track/0Y9GlM4XyI7Av4FqN6g2QF, lyrics: https://lyrics.fandom.com/wiki/Blaze_Ya_Dead_Homie:Intro_2_Tha_Hood.

[20] Metadata: https://open.spotify.com/track/4h6zWkjeAY7FSC5Gn1X9P8, lyrics: https://lyrics.fandom.com/wiki/Cancerslug:Nothing.

[21] Metadata: https://open.spotify.com/track/0ZesFcK1Gv4xAnmiZYncL6, lyrics: https://lyrics.fandom.com/wiki/Carcass:Excreted_Alive.

[22] Average times on 5 different runs of the systems, on a server with 2 CPUs Intel Xeon E5-2430 2.50 GHz, 8 GB RAM, 480 GB SSD HD, without applying any particular performance tuning.

Table 3
Comparison of the fastText variants, with and without subword information. Highest F_1/A scores are marked in bold.

System	Non-explicit			Explicit			A	Macro			Weighted		
	P^0	R^0	F_1^0	P^1	R^1	F_1^1		P	R	F_1	wP	wR	wF_1
fastText$_{nos}$	.967	.993	.980	.882	.591	.708	.962	.924	.792	.844	.960	.962	.959
fastText	.970	.994	**.982**	.897	.631	**.741**	**.966**	.933	.812	**.861**	.964	.966	**.963**
Δ	.003	.001	.002	.015	.040	.033	.004	.009	.020	.017	.004	.004	.004

reported times show that fastText$_{nos}$ is the fastest approach both for training and testing. LR-BOW takes more or less the same time than fastText$_{nos}$ for training, but substantially more for testing. fastText takes substantially more than the other two systems for training, but more or less the same than LR-BOW for testing.

Balancing the dataset Given the substantial unbalance between the number of explicit (.7% of the total) and non-explicit (92.3% of the total) training examples, we investigated the adoption of some balancing techniques to see if they may be beneficial especially for correctly predicting the gold explicit lyrics, i.e., the less represented class in the dataset. As mentioned in Section 2, balancing proved beneficial for this task in some previous work (Bergelid, 2018). We separately applied both random under-sampling and over-sampling techniques (Batista, Prati, & Monard, 2004), testing different level of balancing. We define the level of balancing $l = S/(|\text{not} - \text{explicit}| - |\text{explicit}|)$, where $|\text{not} - \text{explicit}|$ and $|\text{explicit}|$ represented the cardinality of the subsets of non-explicit and explicit lyrics in the training dataset, and S is the number of samples added to the explicit class for over-sampling, or removed from the non-explicit class for under-sampling. l goes from 0.0 (i.e., no sampling at all) to 1.0 (i.e, both classes have the same number of samples). Table 4 reports the performance of the fastText classifier when different level of balancing is applied.

In general, both under-sampling and over-sampling positively affect R^1 (resp. P^0), having some detrimental effect on P^1 (resp. R^0). Overall, no improvements is observed on A and wF_1, while a marginal improvement on macro-F_1 (+0.003) can be achieved by under-sampling the dataset with a level of balancing from 0.2 to 0.5. No improvement is noticed by applying over-sampling. Similar considerations hold also LR-BOW: no improvement is observed, both with over-sampling and under-sampling (full results are available in the evaluation material).

Using more advanced classifiers on the fastText word representations Given the possibility to train the fastText word embeddings separately from the fastText classifier, we concluded our assessment checking whether a substantial improvement of classification performance can be achieved by adopting, on the same word embeddings with sub-word information, a more effective classification algorithm than the linear classifier implemented in fastText. Thus, we fed the fastText word embeddings learned on the dataset to a 1D-CNN, i.e. a 1 dimension Convolutional Neural Network, a neural approach that proved effective for text classification (Kim, 2014). Indeed, as discussed in Section 2, Fell et al., 2019 reported some minor improvement (+0.016 on macro F_1) using a CNN architecture (TDS) over logistic regression with a BOW word (i.e., LR-BOW), and the authors concluded that "The final takeaway is that deep models do not necessarily outperform shallow models.".

Table 5 reports the comparison of the performance of the 1D-CNN[23] and the fastText classifier. While we observe some minor overall improvements $(A = +0.001, F_1 = +0.005, wF_1 = +0.001)$, the scores show that the linear fastText model performs substantially on par with the more advanced 1D-CNN model (with the latter taking almost 8 times the training time of the former, in the considered configuration). While further improvements of the 1D-CNN performance may be potentially achieved with a bigger model (we used a network configuration that enabled training and testing on commodity hardware, the same used for the fastText classifier, without the adoption of more advanced hardware architectures), this further confirms the effectiveness of a simple, ready-to-use system like the fastText classifier for explicit lyrics detection.

[23] Used training parameter configuration: Words = 20,000; Sequence Length = 250; Filter size = 128; Layers = 3 layers with 5 Kernels + 1 vanilla layer + global max pooling; Activation function = ReLU; epochs = 10.

6. Conclusions and future work

In this work we proposed to use the fastText classifier for detecting explicit song lyrics, i.e., determining if the lyrics of a given song could be offensive or unsuitable for children. The fastText classifier implements an effective linear classification model, and works with a peculiar distributional text representation that exploits subword information in building the embedding of the words. With respect to other approaches previously proposed for this task, the fastText classifier has several advantages: it is a ready-to-use solution, which can be directly applied "out-of-the-box" on raw lyrics text without any particular pre-processing; it adopts a peculiar bag of *n*-grams word representation, particularly suitable to tackle unknown words, i.e., words not seen during the training of the algorithm; and it works on standard CPU architectures, as available in commodity hardware, without requiring more demanding hardware resources, such as GPUs.

To investigate the performance of the fastText classifier for explicit lyrics detection, we first assembled from publicly available resources (LyricWiki, Spotify) a lyrics dataset of over 800 K songs, annotated with explicit information. Second, we trained and evaluated the fastText classifier and a strong baseline for the task (logistic regression with BOW) on the assembled lyrics dataset, comparing them also with an available pre-trained approach (WASABI) and a standard majority-class baseline. The evaluation showed that the fastText classifier performs effectively on explicit lyrics detection, substantially outscoring all the considered baselines. Third, we empirically confirmed that the subword information effectively contributes to achieve this result, showing that the fastText classifier trained with sub-word information outperforms the same classifier trained without it.

We complemented the quantitative analysis of the classification performance with further assessments, including: (i) a qualitative inspections of the predictions returned by the approach, providing further insight on the task and the way the fastText classifier, and its word embeddings, performs; (ii) an investigation of the impact of applying some class balancing techniques on the dataset, showing that some marginal improvement can be achieved; and (iii), a preliminary experiment on applying more powerful, neural classifiers (a 1D-CNN) on the fastText word embeddings, showing that, despite some marginally higher scores obtained by the considered 1D-CNN, the fastText classifiers performs substantially on par with it.

Future work will address the problem of applying the approach on languages other than English, in order to study the task and the

Table 4
Assessment of the impact of different level of balancing (under-sampling, over-sampling) on classification performance of fastText. Configurations scoring a higher macro-F_1 than the one without balancing are marked in bold.

Sampling-l	Non-explicit			Explicit			A	Macro			Weighted		
	P^0	R^0	F_1^0	P^1	R^1	F_1^1		P	R	F_1	wP	wR	wF_1
under-0.0	.970	.993	.982	.897	.631	.741	.966	.933	.812	.861	.964	.966	.963
under-0.1	.970	.994	.982	.887	.639	.743	.966	.929	.816	**.862**	.964	.966	.963
under-0.2	.971	.993	.982	.881	.646	.745	.966	.926	.819	**.864**	.964	.966	.963
under-0.3	.972	.991	.981	.863	.659	.747	.966	.917	.825	**.864**	.964	.966	.963
under-0.4	.973	.989	.981	.841	.673	.747	.965	.901	.831	**.864**	.963	.965	.963
under-0.5	.974	.988	.981	.822	.685	.747	.964	.898	.836	**.864**	.962	.964	.963
under-0.6	.974	.986	.980	.811	.689	.745	.963	.892	.838	**.863**	.962	.963	.962
under-0.7	.976	.980	.978	.750	.716	.732	.960	.863	.848	.855	.959	.960	.959
under-0.8	.978	.972	.975	.689	.735	.711	.954	.833	.854	.843	.955	.954	.954
under-0.9	.980	.951	.966	.571	.771	.656	.937	.776	.861	.811	.949	.937	.942
under-1.0	.984	.902	.941	.413	.821	.549	.896	.698	.861	.745	.939	.896	.911
over-0.0	.970	.994	.982	.897	.631	.741	.966	.933	.812	.861	.964	.966	.963
over-0.1	.972	.990	.981	.842	.659	.739	.964	.907	.824	.860	.962	.964	.962
over-0.2	.972	.988	.980	.822	.667	.736	.963	.897	.827	.858	.961	.963	.961
over-0.3	.973	.987	.980	.809	.672	.734	.962	.891	.830	.857	.960	.962	.961
over-0.4	.973	.986	.979	.802	.675	.733	.962	.888	.830	.856	.960	.962	.960
over-0.5	.974	.985	.979	.793	.681	.733	.962	.883	.833	.856	.960	.962	.960
over-0.6	.974	.985	.979	.788	.682	.731	.961	.881	.833	.855	.959	.961	.960
over-0.7	.974	.984	.979	.783	.684	.730	.961	.878	.834	.854	.959	.961	.960
over-0.8	.974	.983	.979	.776	.687	.729	.960	.875	.835	.854	.959	.960	.959
over-0.9	.974	.983	.979	.775	.687	.728	.960	.875	.835	.853	.959	.960	.959
over-1.0	.974	.983	.978	.770	.690	.728	.960	.872	.837	.853	.958	.960	.959

Table 5
Comparison of the classification performance of the fastText classifiers and a 1D-CNN, fed with the same fastText word embeddings.

System	Non-explicit			Explicit			A	Macro			Weighted		
	P^0	R^0	F_1^0	P^1	R^1	F_1^1		P	R	F_1	wP	wR	wF_1
fastText	.970	.994	.982	.897	.631	.741	.966	.933	.812	.861	.964	.966	.963
1D-CNN	.971	.994	.982	.899	.643	.750	.967	.935	.818	.866	.965	.967	.964

performance of the fastText classifier also in cases where fewer training/testing material is available. We also plan to further investigate the exploitation of the fastText word embeddings with more advanced binary classifiers than the fastText (linear) classifier, checking if the formers, despite requiring more resources or advanced processing architectures, may substantially outperform the latter.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations Newsletter, 6*, 20–29. https://doi.org/10.1145/1007730.1007735. URL: https://doi.org/10.1145/1007730.1007735.

Bergelid, L. (2018). *Classification of explicit music content using lyrics and music metadata*. Master's thesis KTH, School of Electrical Engineering and Computer Science (EECS). URL:http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1228997&dswid=-4612..

Bloehdorn, S., & Hotho, A. (2006). Boosting for text classification with semantic features. In B. Mobasher, O. Nasraoui, B. Liu, & B. Masand (Eds.), *Advances in web mining and web usage analysis* (pp. 149–166). Berlin, Heidelberg: Springer, Berlin Heidelberg.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5*, 135–146. URL:https://www.aclweb.org/anthology/Q17-1010.10.1162/tacl_a_00051.

Carta., S., Corriga., A., Mulas., R., Recupero., D. R., & Saia., R. (2019). A supervised multi-class multi-label word embeddings approach for toxic comment classification. In *Proceedings of the 11th international joint conference on knowledge discovery, knowledge engineering and knowledge management – Vol. 1: KDIR* (pp. 105–112). INSTICC SciTePress. doi: 10.5220/0008110901050112..

Chin, H., Kim, J., Kim, Y., Shin, J., & Yi, M. Y. (2018). Explicit content detection in music lyrics using machine learning. In *2018 IEEE International Conference on Big Data and Smart Computing, BigComp 2018, Shanghai, China, January 15-17, 2018* (pp. 517–521). IEEE Computer Society. https://doi.org/10.1109/BigComp.2018.00085. URL: https://doi.org/10.1109/BigComp.2018.00085.

Corazza, M., Menini, S., Cabrio, E., Tonelli, S., & Villata, S. (2020). A multilingual evaluation for online hate speech detection. *ACM Transactions on Internet Technology, 20*. URL: https://doi.org/10.1145/3377323. doi: 10.1145/3377323..

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*, 273–297. https://doi.org/10.1023/A:1022627411411. URL: https://doi.org/10.1023/A:1022627411411.

Davidson, T., Warmsley, D., Macy, M. W., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Proceedings of the eleventh international conference on web and social media, ICWSM 2017* (pp. 512–515). AAAI Press. URL:https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15665.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, Volume 1 (Long and short papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. URL:https://www.aclweb.org/anthology/N19-1423. doi: 10.18653/v1/N19-1423..

Fell, M., Cabrio, E., Corazza, M., & Gandon, F. (2019). Comparing automated methods to detect explicit content in song lyrics. In *Proceedings of the international conference on recent advances in natural language processing (RANLP 2019)* (pp. 338–344). Varna, Bulgaria: INCOMA Ltd. URL:https://www.aclweb.org/anthology/R19-1039. doi: 10.26615/978-954-452-056-4_039..

Fell, M., Cabrio, E., Korfed, E., Buffa, M., & Gandon, F. (2020). Love me, love me, say (and write!) that you love me: Enriching the WASABI song corpus with lyrics annotations. In *Proceedings of the 12th language resources and evaluation conference* (pp. 2138–2147). Marseille, France: European Language Resources Association. URL:https://www.aclweb.org/anthology/2020.lrec-1.262.

Fix, E., Hodges, J., & U.S. of Aviation Medicine. (1951). Discriminatory analysis: Nonparametric discrimination, consistency properties. USAF School of Aviation Medicine. URL:https://books.google.it/books?id=VN07ngEACAAJ..

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th conference of the European chapter of the association for computational linguistics: Vol. 2, Short papers* (pp. 427–431). Valencia, Spain: Association for Computational Linguistics. URL:https://www.aclweb.org/anthology/E17-2068.

Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G. (2004). Multinomial naive bayes for text categorization revisited. In *Proceedings of the 17th Australian joint conference on advances in artificial intelligence AI'04* (pp. 488–499). Berlin, Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-540-30549-1_43. URL: https://doi.org/10.1007/978-3-540-30549-1_43.

Kim, J., & Yi, M. Y. (2019). A hybrid modeling approach for an automated lyrics-rating system for adolescents. In L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, & D. Hiemstra (Eds.), *Advances in information retrieval – 41st European conference on IR research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I* (pp. 779–786). Springer Vol. 11437 of Lecture Notes in Computer Science. URL: https://doi.org/10.1007/978-3-030-15712-8_53. doi: 10.1007/978-3-030-15712-8_53..

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1746–1751). Doha, Qatar: Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1181. URL:https://www.aclweb.org/anthology/D14-1181.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st international conference on machine learning – Vol. 32 ICML'14* (p. II-1188–II-1196). JMLR.org..

Mahedero, J. P. G., MartÍnez, A., Cano, P., Koppenberger, M., & Gouyon, F. (2005). Natural language processing of lyrics. *Proceedings of the 13th annual ACM international conference on multimedia MULTIMEDIA '05* ('05, pp. 475–478). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1101149.1101255. URL: https://doi.org/10.1145/1101149.1101255.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review, 33*, 1–39. https://doi.org/10.1007/s10462-009-9124-7. URL: https://doi.org/10.1007/s10462-009-9124-7.

Sparck Jones, K. (1988). A statistical interpretation of term specificity and its application in retrieval. In *Document retrieval systems* (pp. 132–142). GBR: Taylor Graham Publishing.

Vaglio, A., Hennequin, R., Moussallam, M., Richard, G., & d'Alché-Buc, F. (2020). Audio-based detection of explicit content in music. In *ICASSP 2020–2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 526–530).

Vanni, L., Ducoffe, M., Aguilar, C., Precioso, F., & Mayaffre, D. (2018). Textual deconvolution saliency (TDS): A deep tool box for linguistic analysis. In *Proceedings of the 56th annual meeting of the association for computational linguistics (Vol. 1: Long papers)* (pp. 548–557). Melbourne, Australia: Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1051. URL: https://www.aclweb.org/anthology/P18-1051.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 1480–1489). San Diego, CA: Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1174. URL: https://www.aclweb.org/anthology/N16-1174.