CS230

Detecting Release Decade From Song Lyrics

Joe Sommer
jsommer1@stanford.edu

Abstract

The way people speak changes over time, and can be reflected in pop cultural media such as books or songs. In this project, I apply a couple of different neural network models - one simple, and one based on Long Short-Term Memory (LSTM) models - in order to try to classify the decade a song was released in solely by its lyrics. While I did not achieve significant success with my models, my approach shows potential for success using more robust techniques.

1 Introduction

While there are ways to find out the age of the materials of a physical book or document, there may be situations when people such as linguists or historians would like to figure out the original publication date of a text through the writing alone. Since vernacular speech changes with time, and because song lyrics tend to reflect how people speak colloquially, my aim is to try to train neural networks to predict the decade in which a song was written in just by its lyrics.
The input to my models are a vectorized version of the song's lyrics, meaning each word is represented by the index of that word in a larger corpus of text. I then pass that two different neural networks – a simple, regular one for my basic model, and a more complex one made of LSTMs for my sequential model. I then find which decade, between the 1960's and the 2010's, the song was most likely released during.

2 Related work

Upon looking for previous attempts at this challenge, I found that I could not find anything exactly like my project. It seems, however, that there has been plenty of work done regarding sentiment classification, and I believe that there are subtleties involved in detecting sentiment that could also be applied to detecting something as nuanced as a specific decade's vernacular. In their research, Radford et al. have found that LSTM models could be used to efficiently determine sentiments involving high-level concepts, and that a particular unit from these models could generate sentiments as well [1]. Peng et al. found that RNN-based models could be used to detect sentiment based on character radicals of the Chinese writing system, which can then lend themselves to sentence-level sentiment classification [2]. Finally, Kalchbrenner et al. found that there are ways to use CNN models to capture long-term dependencies, which could be useful when involving entire songs or longer texts [3]. These examples are but a handful of products of research showing the ability of neural network models to capture subtleties from text, which are seem promising in the context of what I am attempting.

CS230: Deep Learning, Fall 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Dataset and Features

3.1 Billboard's Hot 100s

One dataset was of Billboard's Hot 100s from 1965 to 2015, scraped and compiled by another researcher [4]. After cleaning the data, I was left with 4,853 usable songs with their full lyrics represented as a string and their release years. This dataset provided a fairly even distribution of sequential data, which made it useful to train LSTM-based models. Each song's lyrics were then used to create a vectorizer corpus, and converted into an array of numbers representing each word's position in the corpus. While the sequential data seemed useful, its small size led me to look for a larger dataset.

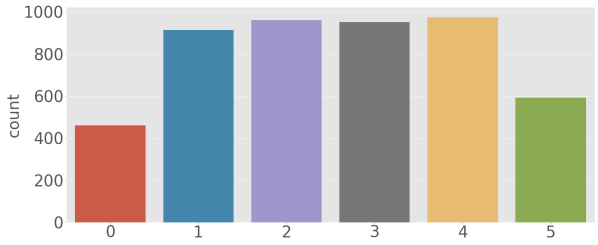

Figure 1: Billboard Dataset Distribution

3.2 Million Song Dataset

The second dataset I used was a subset of the Million Song Dataset [5]. From here, I gathered songs with release years available from the musiXmatch Dataset, and gathered 154,527 songs released between 1965 and 2015 [5]. These songs' lyrics were only available in Bag-of-Words format, so I could not use them for my sequential models. I still created a non-sequential vector of each word being used and vectorized that as well to use as input for my model. This dataset was significantly larger than the Billboard's one, but I later found that the distribution was incredibly skewed towards songs released between 2000 and 2010.

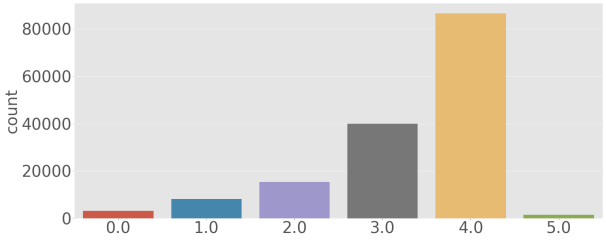

Figure 2: Million Song Dataset Distribution

4 Methods

I implemented two different models: a simple network to use as a benchmark and to see the effect of the larger Million Song dataset, and a more complex one to try to take advantage of the sequential data in the Billboard's dataset. Both models encode the release decade into a one-hot vector using Keras, then feed their respective networks' outputs into a Softmax layer [6].

4.1 Base Models: Billboard and Million

As a baseline to compare my more complex sequential models to, I decided to implement a very simple neural network using Tensorflow [7]. This "network" was just a single hidden layer that

condenses the input to 10 units, then sends that to the Softmax layer to determine the decade. I used trained this base model on both the Billboard's data and the Million Song data to compare the effects of a larger dataset.

4.2 Sequential Models: 2-Stack and 3-Stack

My more complex models were revolved around two details: taking advantage of the sequential Billboard's data, and trying to reduce overfitting from the small dataset size. Thus, my two Sequential models were based around stacking bidirectional LSTMs (BLSTMs) in tandem with dropout, in order to apply regularization. In an attempt to see if there was any important information about lyrics in the context of their previous and next words, I chose to use BLSTMs instead of regular LSTMs. I trained one model, 2-Stack, with two BLSTMs, and another, 3-Stack, with three. 2-Stack was trained by alternating a BLSTM with a Dropout layer twice before outputting to the Softmax layer. 3-Stack added a third BLSTM to the output of 2-Stack before going to the Softmax layer.

5 Experiments/Results/Discussion

5.1 Hyperparameters

I trained each model according to these hyperparameters:

- Batch Size : 10 (Base), 20 (Sequential)
- Epochs : 75 (Base), 130 (Sequential)
- Learning Rate : 0.001
- Beta 1 : 0.9
- Beta 2 : 0.999

I started with a batch size of 10 and 75 epochs for the base models since their validation accuracies seemed to stagnate quickly. The sequential models took longer both in terms of computation and for validation accuracies to converge, so I decided to double the batch size and increase the number of epochs.

5.2 Loss

Since I was training for multi-class classification, I decided to use Categorical Cross-Entropy as my loss function. Categorical Cross-Entropy is defined as [8]:

$$L(y,\hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * log(\hat{y}_{ij}))$$

Figure 3: Categorical Cross-entropy Loss Function

5.3 Metrics

As a general way to describe a model's performance, I chose accuracy as the primary quantitative metric. Given that this is a 6-class classification problem, I decided to qualitatively look at the general spread of wrong classifications within the confusion matrices as well. I reasoned that even if the accuracies of two models are similar, perhaps one could be more closely centered around the correct class, while the other could be spread out more.

5.4 Results

All four models did not yield fantastic results, but were also not terrible. Randomly guessing a decade would give about a 16.67 percent accuracy, so considering the relative simplicity of the models and the shortcomings in the datasets, I would say that these results are not too bad. Below is a table of the results:

Model	Train Accuracy (%)	Test Accuracy (%)
Billboard	99.26	37.77
Million	67.23	52.53
2-Stack	99.18	40.11
3-Stack	79.25	37.71

Figure 4: All model accuracies

The Million model gave the highest test accuracy out of the four, but this should be taken with a grain of salt. The Million Song Dataset was heavily skewed towards the 2000's decade, with over half of the data in that category alone. The confusion matrix also shows that this model is essentially training to guess 2000's for about half the time, just mimicking the poor distribution of training data. Therefore, I don't think that this was the best model.
2-Stack yielded the second highest test accuracy, with 40.11 percent. While it produced higher accuracy than 3-Stack, there are interesting details to be found in their confusion matrices. 2-Stack's mis-classifications are spread out more among the 6 decades, whereas 3-Stack's mis-classifications are spread more closely around the actual decade itself. This may imply that the additional BLSTM is managing to capture something that centers the prediction more closely around the actual year.
Finally, it is interesting to note that a single-layer neural network is able to still yield a higher accuracy than 3-Stack. Perhaps the sequential information that remains in the vectorized song lyrics already contains a good amount of information that can be captured easily by neural networks.

Billboard's Hot 100s

	Actual					
Predicted	**1960's**	**1970's**	**1980's**	**1990's**	**2000's**	**2010's**
1960's	**20**	30	11	5	6	6
1970's	64	**127**	95	58	40	21
1980's	36	58	**99**	60	38	30
1990's	20	24	50	**110**	41	19
2000's	5	17	29	35	**129**	34
2010's	4	10	14	11	35	**65**

Million Song

	Actual					
Predicted	**1960's**	**1970's**	**1980's**	**1990's**	**2000's**	**2010's**
1960's	**25**	61	64	127	194	3
1970's	30	**131**	111	248	294	2
1980's	65	234	**580**	747	964	20
1990's	133	376	757	**2039**	2836	30
2000's	692	1672	3047	8904	**21579**	388
2010's	0	0	1	1	4	**0**

2-Stack

	Actual					
Predicted	**1960's**	**1970's**	**1980's**	**1990's**	**2000's**	**2010's**
1960's	**32**	26	20	19	8	8
1970's	64	**127**	97	41	28	16
1980's	36	57	**96**	51	25	14
1990's	8	27	30	**84**	19	8
2000's	7	19	43	74	**177**	61
2010's	2	10	12	10	32	**68**

3-Stack

	Actual					
Predicted	**1960's**	**1970's**	**1980's**	**1990's**	**2000's**	**2010's**
1960's	**1**	0	0	0	0	0
1970's	86	**118**	64	27	14	9
1980's	45	108	**132**	75	37	21
1990's	15	30	78	**127**	67	42
2000's	2	10	24	50	**171**	103
2010's	0	0	0	0	0	**0**

Figure 5: Confusion Matrices

6 Conclusion/Future Work

My sequential models were able to produce better results than expected, considering limitations, but were nothing fantastic. However, there are glaring issues of not having enough data available and training on skewed data. Therefore, the most immediate next step to this project would be to try to amass a significantly larger amount of properly distributed data. Also, as the confusion matrices for 2-Stack and 3-Stack showed, there may be potential for improvement if more complex models are used. It would make sense that a robust, pre-trained model such as Google's BERT could potentially provide much higher accuracies than a couple of BLSTMs thrown together. Finally, the classes themselves could be tweaked as well. An issue that this current classification problem could have is differentiating between songs that were released within a year or two of each other, but fall into different categories. For example, will a song from 1979 have significantly different lyrical content than a song from 1980? Thus, it might also be worthwhile adjusting the classifier definitions as well.

References

[1] Radford, Alec, Rafal Jozefowicz, and Ilya Sutskever. "Learning to generate reviews and discovering sentiment." arXiv preprint arXiv:1704.01444 (2017).

[2] Peng, Haiyun, Erik Cambria, and Xiaomei Zou. "Radical-based hierarchical embeddings for chinese sentiment analysis at sentence level." The Thirtieth International Flairs Conference. 2017.

[3] Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences." arXiv preprint arXiv:1404.2188 (2014).

[4] https://github.com/walkerkq/musiclyrics.

[5] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[6] https://github.com/keras-team/keras.

[7] Abadi, Martín, et al. "Tensorflow: A system for large-scale machine learning." 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016.

[8] https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy.