biblo.bib

Sentiment Analysis of Lyrics using NLP and RNN

Robert Rhys Thomas Prudente
Faculty of Linguistics, Philosophy and Theory of Science, University of Gothenburg
Language Technology

January 31, 2020

Abstract

The aim of this project is to analyse the sentiment of song lyrics and train a model that can accurately predict a song's genre based on the sentiment of the lyrics. Two methods will be used, training a model and attempt to learn embeddings as well as using pretrained embeddings. The results from both experiments show some promise however the accuracy is not quite high enough.

1 Introduction

Sentiment is described as the computational study of people's opinions, sentiments, evaluations, attitudes, moods, and emotions and can be represented in a number of ways [?] . In the case of this project we will use the word vector. A word vector is a word represented wit a real value and rely on the distance or angle between pairs of word vectors as the primary method for evaluating the quality of word representations [?] . For example the vector of man and king should be closer together than cat and soap. In this experiment I aim to use both learned embeddings and pretrained embeddings to see how well the model can predict the genre of a song based on its lyrics. The pipeline of the project involves, visually inspecting the data which is in csv format, loading the data into Pytorch's Torchtext library, pre-processing the data, training a model which analyses the sentiment of the song lyrics and finally testing the model by comparing it to unseen data. My hypothesis is that there is enough semantic data in a song lyric and that each music genre in the dataset are contrasting enough that a machine learning algorithm can somewhat accurate assume a song genre based on the lyrics in the song. I have chosen to use the whole song rather than just lines in a song as I feel the song as a whole is more representative than a line however this is also worth testing.

1.1 Torchtext

Throughout the project I will be using Pytorch's Torchtext library in data preprocessing, encoding and training. I found that it significantly speeds up a number of tasks where traditionally one would have to do by hand. It took a while to understand what everything does and there are a few limitations, for example it is not possible to export the training batch like with the Torch Dataloader, which can be pickled, as it contains a generator object. Therefore when running the program the preprocessing, training and testing has to be done in one script. I will describe more the functions of Torchtext throughout the paper.

1.2 Applications

Applications of this kind of experiment include, score aggregating for review sites, social media monitoring, analytic and marketing. If one were to automate such processes it adds to efficiency and time saving. This project has no real application apart from just an experiment applying sentiment to lyrics.

2 Dataset

The dataset used in the project is a web scraping of lyrics from the website Metrolyrics [?]. The datast is formatted into columns consisting of

- index *integer*
- song *name of song*
- year *integer*
- artist *string artist name*
- genre *string*
- lyrics *string containing all lyrics from song*

The columns we are interested in are the *lyric* and *genre* columns. The data contains around 380,000 exampls over 10 song lyrics. There is not an equal number of examples per genre so it will be necessary to noramlise the examples to avoid overfitting as few genres have much larger presence than others.

Therefore to begin with the training data is processed so that before any preprocessing starts we have an equal number of examples from each genre. In this case I am using 5 genres with are as follows pop, rock, metal, country and hip-hop.

I have chosen these genres as I felt that they are a good contrast and also in some sense quite similar. Pop and rock can often be interchangeable therefore it will be interesting to see if the model can tell them apart. Conversely Hip-hop is somewhat different due to the fact that the lyrics are often rapped allowing a greater volume of lyric per song. The dataset requires quite an extensive amount of cleaning as there are more than one language present which can cause issues for training as well as corrupt scrapings and irrelevant text such as headings for sections of songs (e.g verse, chorus). I had attempted to use more than the five genres but these are the genres with the most examples.

The below subsections are a line from each genre of music so we can see how semantically there are differences and similarities.

2.1 Pop

Pop music is typically the music that is created with the intent to sell as much as possible with formulaec songs that fit with what is popular at the moment. The music is usually repetitive, catchy and uplifting.

see your face in my mind like a neon sign

2.2 Rock

Rock music is rooted in noncomformity and and high energy.

If it ain't buzzing like a beehive, kick it into overdrive

2.3 Metal

Metal music, whilst a sub branch of rock music is usually much loader, faster and agressive. Depending on what variation of metal, the lyrics usually envoke ideas of fantasy, violence, pain and long hair.

To whom in a blood drenched cry

2.4 Country

Country music, which is most popular in North America usually has a more nostalgic and simplistic nature to the music and lyrics also focuses somewhat on religion and agriculture. Whilst also overlapping slightly with pop music, the religious and rural features of the music are what distinguishes the genre slightly.
But the good Lord had it planned that way

2.5 Hip-Hop

Hip-Hop is characterised by looped samples and rapping. The genre is quite a divergence to pop and rock music however the genres to overlap to certain extents. The lyrics to hip-hop usually reflect the environment and background of the artist

To feed the 5,000 fellas, hustlers and street dwellers

2.6 Analysis of Dataset

Whilst we can see in these brief cherry-picked examples there are themes that set the examples apart. As humans we find it easy to notice patterns and within these examples one may have different ideas to what each example is about. In project task we give the task to a machine.

3 Preprocessing the data

The first challenge of the process is to inspect the data and see what needs to be done to translate it to a format a computer can understand. One issue with the dataset is that it contained a lot of anomalies and artifacts that are a result of bad encoding and lazy webscraping. We will see to these in the first step with is tokenization.

3.1 Tokenization

The first step is processing the text by identifying all of the tokens present. This is a relatively simple step but the issues come when one decides what consists of a token. One needs to think about what features are import or hindrances to the task. In the case of song lyrics one feature that isn't really required is the punctuation as it doesn't rally provide any semantic meaning. Another issue is as mentioned before the corrupted characters, these can all be removed through meticulous tokenization either making use of NLTK's inbuilt word tokenizer or being more thorough by using regex tokenization which, albeit being much more time consuming, can yield better results. After tokenization the data is outputted to a text file to be further processed.For this project I have used Spacy's inbuilt tokenizer.

3.2 Encoding

For encoding there are a few options, I have used trained embeddings as well as using Glove embeddings of varying dimensions. As I am using Torchtext this is a relatively simple feature to implement. Torchtext creates a vocabulary of all words in the dataset which can be optionally updated to use the Glove vectors.

4 Method

4.1 Generating the Data

Torchtext handles most of the preprocessing and batching. First a field object is created. Within this object we specify the text and the label, this process is basically telling Torchtext what each column in the training data specifies, assuming one is using a csv like file format. The program then preprocesses the csv file using Torchtext's TabularDataset function. There are a number of parameters that need to be taken into account, for example *sequential, use_vocab, lower and include_lengths.* These parameters signify that the data is sequential and requires tokenization, if we require a vocab object, if the text needs to be lower case and if we want to record the lengths of each sequence. These are to be true. We can also specify which tokenizer we will be using, in this case it is Spacy. The script then applies these fields to the data which we are importing from a csv file.

4.2 Generating vocab

We can then generate a vocab object which is saving each word type to a numerical representation. Whether or not to use pretrained vectors can be specified as a command line argument. The vectors are then generated using Torchtext's *build_vocab* command and can then be loaded when training the model

4.3 Batching/Iterators

The data then needs to be put into a format that can be used in the neural network. The data needs to be transformed into an iterator object. For this we can use Torchtext's *BucketIterator*, this allows text with similar sequence length to be batched together. This is useful due to the range of length in song lyrics and another reason that we kept the sequence length from earlier. Therefor we now have batched and padded data that can now be fed into the neural network.

4.4 Model

The model I am using is a recurrent neural network with a gated recurrent unit. I have used this architecture due to the nature of the data. As lyric data can be quite long it is good to use a GRU to act as memory storage. There are a number of parameters than can be adjusted in the training process, for example the size of the batches of training data, the number of epochs and whether or not to use pretrained vectors. The model is adapted from a sentiment analysis tutorial found here [?].

The model is initialised with a number of parameters, additionally I have added the option to use pretrained vectors. When the model has input the sequence is embedded, either with or without pretrained vectors. Depending on the dimensions of the vectors the embedding could be either 50,100,200 or 300 as these are available in Torchtext. When using vectors for the first time Torchtext downloads and caches the vectors to a hidden folder in the project folder. The standard training epochs is ten and this can be changed with command line arguments.

5 Results

I performed the experiment four times, using trained vectors and all dimensions of the Glove vectors (50,100,200,300). The model performed relatively well.