Song Era Classifier

Course Project of CS4041 NLP

Final Report

Gazala Muhamed, Shravani Sridhar, Shruti Nair, Aditi Menon
Guided By: Athira P K and Suneera C M

March 27, 2018

Abstract

Our aim is to apply natural language processing and machine learning in order to classify English song lyrics based on decade. We apply recurrent neural network models from a large dataset of song lyrics using Python. Lyrics provide an important part of the semantics of a song, therefore their analysis is fundamental for the development of complete music information retrieval systems. We visualise the performance of the model by extracting the lines and words from the songs which have more weight in the classification task.

As lyrics exhibit a hierarchical layer structure - in which words combine to form lines, lines form segments, and segments form a complete song - we use a hierarchical attention network (HAN) to exploit these layers and further learn the importance of the words, lines, and segments. We will use the Natural Language Toolkit (NLTK) as a platform for creating the algorithms in Python.

1 Introduction

Automatic classification of music is an important and well-researched task in Music Information Retrieval (MIR). Previous work on this topic has focused primarily on classifying songs into mood, genre, annotations, and artists. Typically, one or a combination of audio, lyrical, symbolic, and cultural data is used in machine learning algorithms for these tasks.

Genre classification using lyrics is a natural language processing (NLP) problem. In NLP, meanings and labels are assigned to text; therefore genre classification of the lyrical text can be done nearly accurately. Traditional approaches in text classification have utilised n-gram models and algorithms such as Support Vector Machines (SVM), k-Nearest Neighbour (k-NN), and Nave Bayes (NB). Deep learning has in recent years been utilised in several MIR research topics, including live score following, music instrument recognition, and automatic tagging. However, not much research has looked into the performance of these deep learning methods with respect to the genre and era classification task on lyrics. Thus, we attempt to remedy this situation by extending deep learning approaches to text classification to the particular case of lyrics.

2 Problem Statement

The purpose of our project is to develop a system that will take English song lyrics as input and output the decade the song came out. This will require Natural Language Processing as the system would need to process lyrics that are in a natural language. We utilize a Hierarchical Attention Network (HAN) for the processing and classification of lyrics due to lyrics' hierarchical nature.

The song lyrics are split into layers, and a bidirectional recurrent neural network is applied at each layer to obtain hidden state representations. The attention mechanism from HAN is then applied to form a weighted sum of that layer's hidden representations. We have now passed to a higher layer and can repeat the process until we finally end up with a vector which summarizes the whole song, from which we can then classify the song into its decade.

3 Design

3.1 Data Collection

We have a separate program to collect the top 100 lyrics by year. This will be our input to the HAN.

3.2 Data Preprocessing

Reading lyrics from csv to get the training and validation sets.

3.3 Tokenization

Here we tokenize the lyrics based on words, to convert strings to tokens that can be easily inputed to our HAN.

3.4 Embedding Layer

Here we assign weights to the words for the Word Encoder Layer of HAN.

3.5 Hierarchical Attention Networks

HAN consists of several parts: a word sequence encoder, a word-level attention layer, a sentence encoder and a sentence-level attention layer.

3.5.1 Word Encoder

We use a bidirectional GRU (Bahdanau et al., 2014) to get annotations of words by summarizing information from both directions for words, and therefore incorporate the contextual information in the annotation.

3.5.2 Word Attention

Not all words contribute equally to the representation of the sentence meaning. Hence, we introduce attention mechanism to extract such words that are important to the meaning of the sentence.

3.5.3 Sentence Encoder

We use a bidirectional GRU to encode the sentences.

3.5.4 Sentence Attention

To reward sentences that are clues to correctly classify a document, we again use attention mechanism and introduce a sentence level context vector and use the vector to measure the importance of the sentence

3.6 Fitting the Model and Output

We compile and fit the model to the training and testing sets, with approriate epoch and batch sizes. We output the metrics to the user, and save the model(including weights) to a local file, for loading in the future (without recompilation).

4 Work Done

The work that you have done so far related with the project should be stated clearly. Include separate sections for design details, implementation and results.

We downloaded our dataset from Billboard, which has been publishing a Year-End Hot 100 every December since 1958. The dataset inherently didn't have sentences, so we had to download the PyLyrics library in order to download the lyrics in the right format (in sentences).

4.1 Implementation

In this part you have to include the implementation details.

In our project, we used Keras, which is based on TensorFlow, which are Python packages for machine learning.

We use Tokenizer (from keras.preprocessing.text) to tokenize our data based on the words.

```
tokenizer = Tokenizer(nb_words=MAX_NB_WORDS)
tokenizer.fit_on_texts(texts)

data = np.zeros((len(texts), MAX_SENTS,
     MAX_SENT_LENGTH), dtype='int32')

for i, sentences in enumerate(lyrics):
   for j, sent in enumerate(sentences):
      if j< MAX_SENTS:
         wordTokens =
              text_to_word_sequence(sent)
         k=0
         for _, word in enumerate(wordTokens):
            if k<MAX_SENT_LENGTH
             if tokenizer.word_index[word] <
                  MAX_NB_WORDS:
              data[i,j,k] =
                   tokenizer.word_index[word]
              k=k+1
word_index = tokenizer.word_index
```

The embedding matrix is created using the GloVe corpus, which gives weights to words. GloVe uses pre-trained word vectors.

```
embedding_layer = Embedding(len(word_index)
     + 1,
EMBEDDING_DIM,
weights=[embedding_matrix],
input_length=MAX_SENT_LENGTH,
trainable=True)
```

One layer of the network takes in vectors, applies a bidirectional GRU to find a forward hidden state

$$\vec{h}_j$$

and a backward hidden state

$$\overleftarrow{h}_j$$

, and then uses the attention mechanism to form a weighted sum of these hidden states to output as the representation.

We use two layers in our Hierarchical Attention Network, one for words and the other for sentences.

```
#sentEncoder is passed to the second layer,
     Lyrics_encoder
sentEncoder = Model(sentence_input, l_att)
Lyrics_encoder =
     TimeDistributed(sentEncoder)(Lyrics_input)
```

The TimeDistributed layer is what allows the model to run a copy of the encoder to every sentence in the document.

```
l_lstm = Bidirectional(GRU(100,
     return_sequences=True))(embedded_sequences)
l_dense = TimeDistributed(Dense(200))(l_lstm)
l_att = AttLayer()(l_dense)
sentEncoder = Model(sentence_input, l_att)
preds = Dense(3,
     activation='softmax')(l_att_sent) #3
     classes
model = Model(Lyrics_input, preds)
```

The final layer outputs 3 possible categories corresponding to the three decades - 1980's, 1990's,and 2000's. We use the softmax function in the final layer of our neural network-based classifier. Since we have a relatively small dataset (1800 datapoints), we set the batch size to 200 and the number of epochs to 5.

```
model.compile(loss='categorical_crossentropy',
          optimizer='rmsprop',
          metrics=['accuracy', recall])

#model fitting - Hierachical attention
     network
model.fit(x_train, y_train,
     validation_data=(x_val, y_val),
     nb_epoch=5, batch_size=200)
```

4.2 Results

insert pic here? We found the recall to be ?, which in the ideal case, should be close to 1. We found the accuracy to be ?, which is good considering that we have 3 classes whose random chance is .33.

5 Future Work and Conclusions

We have built a system that classifies English song lyrics by decade using a Hierarchical Attention Network. We have three classes (decades), and we have obtained a training set accuracy of, a testing set accuracy of, a precision of, an fmeasure of, and a recall of. We are using two layers: one for words and the next for sentences, and the results we obtained are of the second layer. Our testing set accuracy is good, when compared to the standard accuracy of 1/3 (0.33).

Our classification accuracy has room for improvement, and one way to increase it could be by using a better corpus (i.e, one having better weights). In our project, we used the GloVe model (an unsupervised learning algorithm) for obtaining vector representations for words from a corpus. Using a more balanced and larger dataset would also improve the classification accuracy. The dataset we used has 1858 entries.

Our project could further be extended to classify song lyrics into the more specific years, rather than decades, and also into genres.

References

[1] Zichao Yang,Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy *Hierarchical Attention Networks for Document Classification*,2016

[2] Alexandros Tsaptsinos *Lyrics-Based Music Genre Classification using a Hierarchical Attention Network*, ISMIR 2017