Song Title Prediction with Bidirectional Recurrent Sequence Tagging

Ryan L. Holmdahl
Stanford University
Stanford, CA 94305
ryanlh@stanford.edu

Abstract

We present the task of predicting a song's title given its lyrics and describe its unique difficulties amongst natural language problems. Finding that most song titles are contained in the song's lyrics, we use a bidirectional GRU network to tag those lyric tokens which are present in the title. Our method improves upon an n-gram extraction baseline by 11.8%. Finally, we discuss opportunities for further work.

1 Introduction

Unlike the documents used by most information extraction tasks (article summarization, question answering), song lyrics are not necessarily expositive in nature and have varied and often unclear objectives. Structurally, songs are distinct in their typical use of repetition, their brevity, their vagueness, and their lack of complete sentences and other typical grammar patterns.

Given these differences, songs-as-documents present an interesting and difficult domain of natural language processing. One task within this domain is title extraction: the prediction of a song's title given the content of its lyrics. Effective models for solving this problem can provide insight into the structure of music and can augment generative song models to provide realistic titles given the generated lyrics.

We approach this problem by first noting that many songs – 79.9% in our dataset – contain their title in their lyrics. Thus, we can treat it as a tagging problem, which is significantly easier than construing it as a sequence-to-sequence problem. Disregarding the ordering of the tokens in a song's title (which can be fairly easily solved after tagging and is not the focus of this paper), we attempt to classify each token in a song's lyrics as a title token or a non-title token.

2 Related Work

Graves and Schmidhuber present the bidirectional LSTM architecture [1]. This architecture uses bidirectional recurrent neural networks (BRNNs), which consist of two separate RNNs fed the same input data in reversed orders [2]. The output of these networks are connected via concatenation or some other means and then surfaced, allowing the final hidden state to carry information from both the past and the future with respect to a particular timestep. The authors augment this structure using the Long Short Term Memory unit, which addresses the inability of RNNs to access information from distant timesteps [3].

Our construction of the problem as one of sequence tagging allows us to leverage the topic's existing literature. Huang et al. present a sequence tagging model which employs a bidirectional LSTM [4]. The hidden states of the LSTM at each timestep, in addition to the input embedding at that timestep

and the outputs of adjacent timesteps, are fed as input to a conditional random field classifier for that token.

3 Approach

We modify the bidirectional LSTM architecture to use the simpler Gated Recurrent Unit (GRU) [5]. Similar to the work of Huang et al., we use this network to generate features for each lyric token, which can then be used to classify it as a title or non-title token.

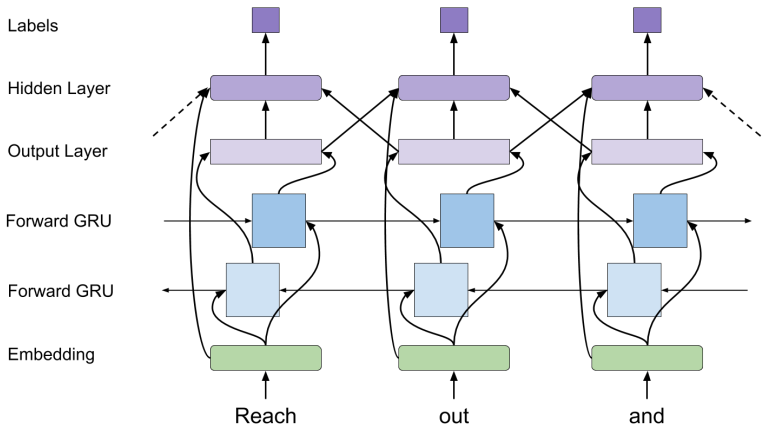

Figure 1: **The bi-GRU architecture.** Each token is transformed into an embedding vector, which is passed into two GRU networks processing the text in opposite directions. At each time step, the hidden states of each GRU for that and adjacent steps are concatenated and, along with the embedding, passed as input to a hidden layer. The hidden layer outputs to a softmax layer, which predicts the token's label.

3.1 Inputs

We create a matrix X whose i-th entry, x_i, is the tokenized lyrics sequence of the i-th song in our dataset of size N. Using a vocabulary V, each entry in the sequence is transformed into a corresponding integer. Each x_i is of a constant length T, with those having fewer tokens being padded with a special token and those with extra having the remainder removed. We refer to the t-th token of x_i as $x_i^{(t)}$.

We build a tensor Y whose i-th entry, y_i, is a vector of label vectors corresponding to the tokens of x_i. If $x_i^{(t)}$ is in the title of song i, then $y_i^{(t)} = [0, 1]$; otherwise $y_i^{(t)} = [1, 0]$. Note that the labeling is irrespective of the number of times that token appears in the title, as long as it appears at least once. That is, if a token j appears at least once in the song title, then every instance of token j in x_i will have a corresponding $[0, 1]$ entry in y_i.

3.2 Embeddings

The integer sequence is used to look up a continuous vector embedding of each token. The embedding matrix is trainable, and is thus updated in each iteration.

Specifically, let L be our $|V| \times D$ embedding matrix, where D is the embedding dimension and $|V|$ is the number of tokens in our vocabulary. We define o_{it} as the one-hot row vector of dimension $|V|$ where the hot index is the value of $x_i^{(t)}$. The embedding of $x_i^{(t)}$, $e_i^{(t)}$, is defined as:

$$e_i^{(t)} = o_{it}L$$

3.3 Bidirectional GRU

The token embeddings e are passed as inputs to a bidirectional recurrent network, with each network consisting of T GRU units. The hidden states of the forward and backwards networks at each timestep are concatenated into a vector $h_i^{(t)}$ and surfaced as input to the classifier.

3.4 Feedforward Classifier

To classify the token at a given timestep t on a given sample i, the hidden state of the bi-GRU $h_i^{(t)}$ is concatenated with the hidden states of the previous and subsequent steps $h_i^{(t-1)}$ and $h_i^{(t+1)}$ and with the embedding vector $e_i^{(t)}$. This vector is then input to a feedforward neural network classifier, defined as follows:

$$v_i^{(t)} = [h_i^{(t-1)}, h_i^{(t)}, h_i^{(t+1)}, e_i^{(t)}]$$

$$\widetilde{h}_i^{(t)} = ReLU(v_i^{(t)}W_1 + b_1)$$

$$\hat{y}_i^{(t)} = softmax(\widetilde{h}_i^{(t)}W_2 + b_2)$$

where W_1 and W_2 are trainable weight matrices and b_1 and b_2 are trainable bias vectors. The final output vector, $\hat{y}$, is of dimension 2 corresponding to the two possible token tags. Special vectors $h^{(0)}$ and $h^{(T+1)}$ are created to act as adjacent vectors for the first and last tokens. These vectors are trainable.

3.5 Loss

To train the network with respect to a given sample i, we minimize the sum of the weighted softmax cross-entropy losses of each term in the sequence, as follows:

$$\sum_{t=1}^{T} w_i^{(t)} CE(y_i^{(t)}, \hat{y}_i^{(t)})$$

where CE is the cross-entropy function, $y_i^{(t)}$ is the correct label for the t-th token of the i-th sample, and $w_i^{(t)}$ is a weight scalar set as a hyperparameter according to the value of $y_i^{(t)}$.

3.6 Other Models

Other models were implemented in the pursuit of this research. Sequence-to-sequence attention models attempted to translate the song lyrics into the title sequence. These models suffered from severe overfitting problems, barely decreasing the development loss while sharply decreasing the training loss. Although these attempts were unable to identify a general pattern for the sequence decoding, future work could apply such models to a larger dataset.

Also implemented was a bag-of-words-to-sequence model. This model converted a song's lyric sequence into a term frequency vector of dimension $|V|$. This term frequency vector was then fed into a hidden layer whose output became the initial state for a decoder RNN. This also suffered from overfitting. In general, the tendency of X-to-sequence models to overfit encouraged us to pursue the tagging approach.

4 Experiments

4.1 Dataset

We combine two datasets consisting of lyric-title pairs. The first pulls information from LyricsFreak and the second from MetroLyrics. After removing songs that do not contain their title in their first T lyric tokens and removing overlap between the datasets, we developed a training set of 100514 songs, a development set of 12578 songs, and a test set of 12658 songs. The songs and titles were tokenized using the Python NLTK word tokenizer.

4.1.1 Vocabulary

We create a vocabulary V of the 50000 most frequent words in the lyrics of our training set. Those tokens present in lyrics or titles that are not present in the vocabulary are replaced with a special token.

4.2 Evaluation

Though we train with respect to the softmax cross-entropy loss, we evaluate our models using the token overlap of the actual title and the predicted title, which we will refer to as the Token Overlap Score.

In particular, let u_i be the token presence vector of dimension $|V|$ corresponding to song i. u_{i_j} is 1 if token j appears in the title of song i and is 0 otherwise. Let $\hat{u}_i$ be the token presence vector of dimension $|V|$ corresponding to the model output of x_i. $\hat{u}_{i_j}$ is 1 if a token with value j was labeled positive in x_i and is 0 otherwise. The Token Overlap Score S is as follows:

$$S(u_i, \hat{u}_i) = \frac{u_i \cdot \hat{u}_i}{\max(\sum_{j=1}^{|V|} u_{i_j}, \sum_{j=1}^{|V|} \hat{u}_{i_j})}$$

This score has several desirable properties. It is on a 0 to 1 scale, with 0 indicating no overlap and 1 indicating full overlap with no excess predictions. The score rewards predictions that have more correct predictions and fewer incorrect predictions such that score is maximized only when the prediction and the ground truth are exactly equal. Using the max of the two vector sums rather than the average does not punish a prediction for including incorrect tokens if those tokens brought the total number of tagged tokens closer to the actual title length.

4.3 Baseline

We used the most frequent n-gram in each song's lyrics as a baseline prediction of its title. Essentially, we find the range of n-gram lengths that scores optimally on the development set and then find its score on the test set. Our specific algorithm is given in Algorithm 1.

4.4 Hyperparameters

We use the first 200 tokens of each song's lyrics; that is, $T = 200$. For a given song i and timestep t, $w_i^{(t)} = 1$ if $y_i^{(t)} = [0, 1]$; otherwise, $w_i^{(t)} = 0.5$. This helps compensate for the label imbalance between title tokens and the more frequent non-title tokens. Our bi-GRU has two layers, and its hidden states in each direction have dimension 256. The hidden classifier layer $\widetilde{h}$ is also of dimension of 256. Data is processed in batches of 64 randomly selected samples and our initial learning rate is 0.1.

4.4.1 Embeddings

To initialize our embedding matrix L, we use dimension 100 GloVe vectors [6]. These vectors were trained on six billion tokens from Wikipedia and Gigaword using the global word-word co-occurrence matrix and capture many interesting properties of words, including relevant nearest neighbors (frog being close to toad) and linear substructures (woman - man being similar to queen

Algorithm 1 Baseline algorithm

procedure BASELINE($devSongs$, $testSongs$)
    $nFloor \leftarrow 0$
    $nCeil \leftarrow 0$
    $bestScore \leftarrow 0$
    for $i = 1; i <= 6; i++$ **do**
        for $j = i; j <= 6; j++$ **do**
            $score \leftarrow$OVERLAPSCOREFORRANGE(i,j,$devSongs$)
            if $score > bestScore$ **then**
                $bestScore \leftarrow score$
                $nFloor \leftarrow i$
                $nCeil \leftarrow j$
    return OVERLAPSCOREFORRANGE($nFloor$, $nCeil$, $testSongs$)
function OVERLAPSCOREFORRANGE($floor$, $ceil$, $songs$)
    $score \leftarrow 0$
    for song in $songs$ **do**
        $maxNgram \leftarrow$ top ngram with length in range [i,j] in song
        $score \leftarrow score+$ token overlap score of $maxNgram$ and song title
    return $score/|songs|$

- king). Those tokens that do not have a corresponding GloVe entry are initialized uniformly at random between -0.1 and 0.1.

4.4.2 Model Selection

We train our model using clipped gradient descent. We select as our "best" model that which has the maximal Token Overlap Score on the development set, which is calculated every 200 iterations. We also evaluate the training and development losses and, if the development loss is not decreasing over a few iterations, we decay the learning rate by a factor of 0.95.

4.5 Results

Figure 2 shows the model loss as iterations progressed. The model ceased to improve after about 30000 iterations, even on the training set. Since the development loss never worsened, increasing the number of parameters via larger hidden dimensions and additional timesteps could be warranted to fit the model better.

The stalling of improvement is corroborated by Figure 3, which describes a slowing of the development Token Overlap Score around 30000 iterations.

We compare the performance of the best model on the development set with the performance of our baseline in Figure 4, where we show an improvement of 11.8% over the baseline.

We can see an example of the model significantly outperforming the baseline in the example in Figure 5. In this case, the song in question lacked many repeated n-grams, so the baseline struggled to select one as the title. Meanwhile, the bi-GRU is still able to identify the key words in the song and present them as the title.

5 Conclusion

We find that the bi-GRU shows significant promise in song title extraction. It outperforms the baseline by a substantial margin, particularly on songs lacking heavy repetition. In future applications, the baseline and the bi-GRU could be applied in tandem, the former to reptitive songs and the latter to more complex songs.

The given implementation of the model does not severely overfit to the training set, indicating that an increase in the number of parameters (tokens read per sample, size of hidden layers, amount of context used by softmax classifier) could be warranted.

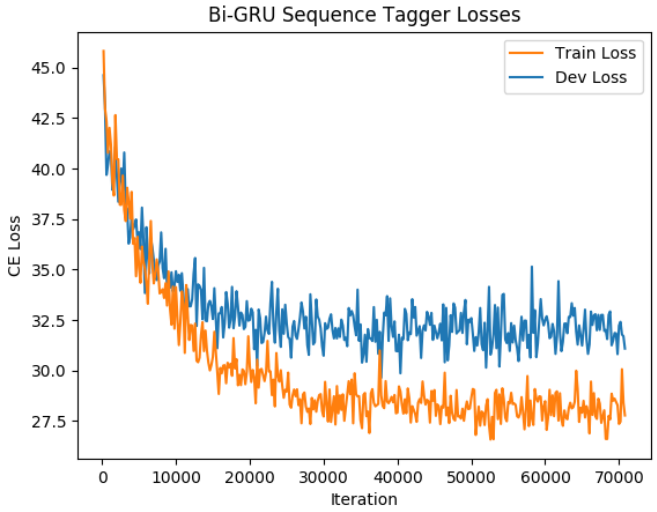

Figure 2: **Model losses over time.** The loss ceases to decrease on both datasets around 30000 iterations.

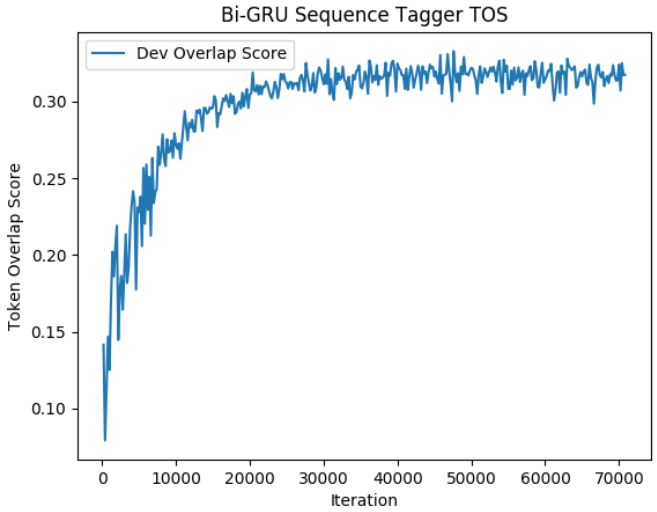

Figure 3: **Development score over time.** The score also ceases to improve on the development set around 30000 iterations.

Model	Test Set TOS
N-Gram Baseline	0.276
Bi-GRU	**0.313**

Figure 4: **Model scores.** The bi-GRU improves upon the baseline by 11.8%.

This research provides a useful foundation for future work. Subsequent efforts could experiment with replacing some of the model components, such as the feedforward classifier layer. Some work could also be done to sort the tagged tokens into a convincing title order. Those implementing generative lyrical models could then begin applying the bi-GRU model to the generation of a title from their lyrics.

Model	Prediction	TOS
N-Gram Baseline	['do', 'you', 'wan', 'na']	0.0
Bi-GRU	['the', 'day', 'school']	0.667

Figure 5: **Example result comparison.** On the test song "Day Off School," which lacks many repeated n-grams, the bi-GRU is able to pick out key terms while the baseline picks one of the more frequent n-grams and is entirely incorrect.

Another means of expanding on this work is to implement a sequence-to-bag-of-words model, wherein the lyrics sequence is used to predict a single bag-of-words vector of dimension $|V|$ rather than to tag each token. Each x, y pair would consist of a lyric token sequence and a vector of dimension $|V|$ whose j-th entry is equal to the number of times j appears in the title. The model would then try to output this vector. This would bypass the current limitation of only being able to process songs that contain their title, as this model could project onto the entire vocabulary, and would also make predicting the quantity of tokens more feasible.

Acknowledgements

Thanks to Gyanendra Mishra for the "380,000+ lyrics from MetroLyrics" dataset, available at Kaggle, and Sergey Kuznetsov for the "55000+ Song Lyrics" dataset, also available at Kaggle. Special thanks also to Kevin Clark, my mentor, and the CS224N teaching staff.

References

[1] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional {LSTM} and other neural network architectures," *Neural Networks*, vol. 18, no. 56, pp. 602 – 610, 2005. {IJCNN} 2005.

[2] P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri, "Exploiting the past and the future in protein secondary structure prediction," *Bioinformatics*, vol. 15, no. 11, pp. 937–946, 1999.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[5] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[6] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, pp. 1532–1543, 2014.