Orca Call Type Sequence Prediction using Deep Natural Language Processing

Kishore Surendra[1], *Christian Bergler*[1]

[1]Friedrich-Alexander-University Erlangen-Nuremberg, Department of Computer Science – Pattern Recognition Lab, Martensstr. 3, 91058 Erlangen, Germany

kishore.surendra@fau.de,christian.bergler@fau.de

Abstract

Orcas(Killer whales), which belong to the dolphin family have been studied extensively by marine biologists due to their significance in the ecosystem. Their behavior, actions and sounds provide a lot of useful information as animal linguistics data.

Complex human language is based on syntax (grammar) and semantics (meaning) and can still be generalized using Deep Natural Language processing techniques for the tasks of text generation, prediction and reproduction. The killer whale sounds on the other hand are perceived more as syntactic than semantic since humans cannot comprehend the stand alone sounds made by them. Therefore, with only the sound patterns in mind (mapped to the behavior), it makes sense to use NLP techniques on animal linguistics.

A language model can be developed and used standalone, such as to generate new sequences of text that appear to have come from the corpus. Language modeling is a root problem for a large range of natural language processing tasks. In this scenario, a similar language model for the killer whale sequences can be built to predict a new sequence of Orca sounds.

In this report, a language model is built on a corpus of sequential killer whale sound patterns and is used to predict the next finite number of actions. These activities are represented in the form of unique alphanumeric characters. Long Short term memory (LSTM) units and GRUs (Graded Recurrent units) have been exploited extensively to train models to learn the killer whale sound patterns. Adding a bidirectional property has been found to improve results. The LSTM networks in particular were found to be the most effective in predicting the killer whale sequences.

Index Terms: Orca, Natural Language Processing, LSTM, GRU

1. Introduction

Orcas are the largest members of the dolphin family which are well studied and whose behavior is well documented in catalogs used by marine biologists [1]. Their acoustic behavior has been studied extensively on the resident fish-eating orcas belonging to the Northeast Pacific regions of North America. These killer whales produce a number of different, group-specific, and social sounds with distinct frequency contours such as echolation clicks , whistles and the pulsed calls - being the most common and intensively studied vocalization of killer whales. The Orchive [2] contains about 20,000 hours of orca underwater recordings, captured via stationary hydrophones in the northern British Columbian Hanson island for over 25 years (1985-2010).

Each of the above mentioned sounds can be considered as the typical sound patterns mentioned in the abstract. Clustering of each denoised audio sample is performed by dividing it into smaller audio files.The output cluster file contains the sequence of clusters in the following format: $\langle Audiofilename \rangle$, $\langle Cluster \rangle$

Clustering was performed either by spectral or K-means methods, which can be specified as a hyperparameter along with the number of clusters [3].This cluster file is used as an input for training the language model.

The same cluster file serves as the raw corpus for input sequence based information retrieval which allows the user to obtain useful information about any valid input sequence of killer whale sound patterns. This information would help save time by guiding us to the actual parts of the tapes containing the sequence of interest.

2. Related Work

Only recently have methods been proposed that go beyond transferring word embeddings in NLP. The prevailing approach is to pretrain embeddings that capture additional context via other tasks. Embeddings at different levels are then used as features, concatenated either with the word embeddings or with the inputs at intermediate layers [4].

An approach called Binary Relevance [5] is widely used due to its simplicity, but it often does not deliver good performance. Intuitively, knowing some labels such as 'sport' and 'football' should make it easier to predict '2018 world cup' and then 'Russia'.

There are several methods that try to capture label dependencies by building a joint probability estimation over all labels $p(y = (y_1, y_2, ..., y_L)|x)$ [6]. The most popular approach, Probabilistic Classifier Chain [7] learns labels one-by-one in a predefined fixed order: for each label, it uses one classifier to estimate the probability of that label given all previous labels predictions, $p(y_l|y_1, ..., y_{l-1}, x)$. Probability of correct classification's well known drawback is that errors in early probability estimations tend to affect subsequent predictions, and can become massive when the total number of label candidates L is large.

Recurrent neural network (RNN) is originally designed to output a sequential structure, such as a sentence [8]. Recently, Zalando [9] has applied RNN to model the sequence of interactions of the users in their webshop. In their work, they use the data from the different sessions of the user to predict the probability that the user will place an order within the next seven days. As inputs, they use a sequence of one-hot encoding vectors which represent past actions as product-views, cart-additions and orders. The RNN model is compared against a logistic regression model with intensive feature engineering efforts over multiple months and used in production systems, achieving a similar accuracy. Additionally, they also provide a way to visualize how the predicted probabilities change over the course of the consumers history.

In [10], collaborative filtering is viewed as a sequence prediction problem and the authors use RNN in the context of movie recommendation. In this case, they create a sequence of ratings of movies to predict which movies the user will watch in the future. They encode the sequence of movies that the user has rated in the past as a sequence of one-hot encoding vectors. In their experiments with the Movielens and Netflix data sets, their method outperforms standard nearest neighbors and matrix factorization, which are typical collaborative filtering methods that ignore the temporal aspect of the data.

3. Methodology

The task of sequence prediction consists of predicting the next symbol of a sequence based on the previously observed symbols. For example, if sound patterns 'A', 'B', 'C', are observed in order, one may want to predict what is the next sound pattern. [11].

There are two steps to perform sequence prediction:

First, one must train a sequence prediction model using some previously seen sequences called the training sequences. This process is illustrated in Figure 1.

The second step is to use a trained sequence prediction model to perform prediction for new sequences (i.e. predict the next symbol of a new sequence), as illustrated in Figure 2.

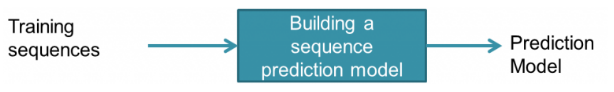


Figure 1: *Building the model.*

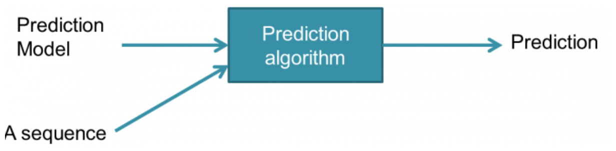


Figure 2: *Making predictions from the model.*

The workflow steps can be categorized into 4 parts:

1. Data splitting

2. Network design

3. Fit network to training dataset

4. Use best model to make predictions

3.1. RNN

A powerful type of neural network designed to handle sequence dependence is called recurrent neural networks. RNNs suffer from short-term memory. If a sequence is long enough, they will have a hard time carrying information from earlier time steps to later ones. So while trying to process a paragraph of text to do predictions, RNNs may leave out important information from the beginning. LSTM s and GRUs were created as the solution to short-term memory. They have internal mechanisms called gates that can regulate the flow of information [12].

3.2. LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture composed of a cell, an input gate, an output gate and a forget gate [13]. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The activation function of the LSTM gates is often the logistic sigmoid function.

3.3. GRU

The GRU is like a long short-term memory (LSTM) with forget gate but has fewer parameters than LSTM, as it lacks an output gate. GRUs have been shown to exhibit even better performance on certain smaller datasets [14]. GRUs use less training parameters and therefore use less memory, execute faster and train faster than LSTMs whereas LSTM is more accurate on dataset using longer sequence.

3.4. Language Model

Statistical Language Modeling, or Language Modeling and LM for short, is the development of probabilistic models that are able to predict the next word in the sequence given the words that precede it.

In a n-gram model, only the previous n-1 words are conditioned on when making a prediction, noting that 'n' denotes the length of context here rather than the total number of words in data. A typical n-gram model is trigram, which separated training set into 3-word groups where the first two words were the evidence for predicting the last word. For a small size word set, n-gram is fast and agile, with fairly good accuracy [15]. LSTM/GRU is more robust and widely used for the ability to preserve long term dependency, and is hence used in this project.

A language model learns the probability of word occurrences based on examples of text. Simpler models may look at a context of a short sequence of words, whereas larger models may work at the level of sentences or paragraphs. Most commonly, language models operate at the level of words.

Language modeling is a root problem for a large range of natural language processing tasks. More practically, language models are used on the front-end or back-end of a more sophisticated model for a task that requires language understanding.

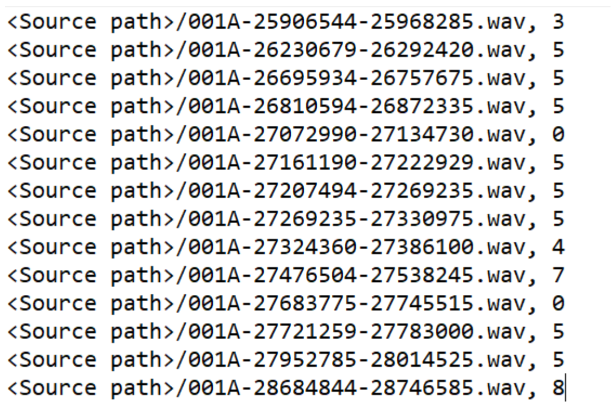
4. Data Basis

4.1. Dataset

The raw dataset is in the form of a text file with each line containing the format : $\langle Audiofilename \rangle$, $\langle Cluster \rangle$.

Each cluster corresponds to a unique sound pattern of the killer whale. A snippet of the raw corpus is as below:

The cluster input file for the language model was created from all audio samples via the recordings between the years 1985 and 2010. So the raw data text files for each of the audio samples are concatenated into one text file. The first 80% of the corpus is used as training data, next 10% as test data and the last 10% of the corpus as validation data, as indicated in Table 1. Overall, there are 465930 sequences/clips.



```
<Source path>/001A-25906544-25968285.wav, 3
<Source path>/001A-26230679-26292420.wav, 5
<Source path>/001A-26695934-26757675.wav, 5
<Source path>/001A-26810594-26872335.wav, 5
<Source path>/001A-27072990-27134730.wav, 0
<Source path>/001A-27161190-27222929.wav, 5
<Source path>/001A-27207494-27269235.wav, 5
<Source path>/001A-27269235-27330975.wav, 5
<Source path>/001A-27324360-27386100.wav, 4
<Source path>/001A-27476504-27538245.wav, 7
<Source path>/001A-27683775-27745515.wav, 0
<Source path>/001A-27721259-27783000.wav, 5
<Source path>/001A-27952785-28014525.wav, 5
<Source path>/001A-28684844-28746585.wav, 8
```

Figure 3: *Raw input data.*

Train set clips	Test set clips	validation set clips
372744	46593	46593

Table 1: *Number of train,test and validation set clips*

4.2. Data Preprocessing

Clustering of each denoised audio sample is performed by dividing the entire tape into smaller audio chunks, and the output cluster file contains the sequence of clusters. This cluster file is used as an input for training the neural network.

A sliding window of length 5 (hyper-parameter) is used to aggregate every 5 consecutive cluster points together from the corpus. So every point in the training data contains 5 entries, and every label in the train labels contains the next entry corresponding to that training data point.

For example : If the cluster file or corpus contains 'N1,N7,N4,N7,N4,N3,....' , then the first element of the training data would be 'N1,N7,N4,N7,N4' and the first element of the training labels will be 'N3' , and so on.

The training data, training labels, test data and test labels are then stored as numpy arrays.

The information retrieval task divides raw data shown in Figure 3 into various lists. Based on the input sequence length, the lists are divided into sublists and only those sublists containing the input sequence are considered, and information such as years, tapes and timestamps of occurrences are recorded in an output text file.

5. Network Model

5.1. Network Design

The numpy arrays of the train data, train labels, test data and test labels are passed into a tokenizer and replaced by the corresponding tokenized numpy arrays. Tokenization is necessary to convert the cluster names which are in alphanumeric forms to unique numbers. The label arrays are converted to categorical format using one hot encoding, and these along with the tokenized training arrays are passed as inputs to one of the following 4 networks :

1. Unidirectional LSTM
2. Bidirectional LSTM
3. Unidirectional GRU
4. Bidirectional GRU

The common architecture for all 4 networks has been shown in Figure 4. The output dimension of the final dense layer is made equal to the vocabulary size, or the number of unique clusters in this case. The loss function used was categorical crossentropy loss due to 21 possible classes, and the optimizer used was adam.

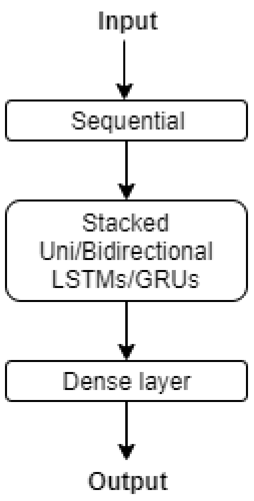


Figure 4: *Network architecture.*

5.2. Training and Evaluation

The training set is reshaped to 3 dimensions as expected by the LSTM/GRU layers. A batch size of 4 was selected for training the neural network for upto 100 epochs, with an early stopping criteria on the training loss as 10 epochs. Finally, the best model was saved using the in-built feature of Keras.

The test data which constitutes 10% of the overall corpus is loaded and reshaped to three dimensions, desirable for the neural network. The number of sound patterns to be predicted are fetched as a hyperparameter. Models trained using one of the four mentioned networks are used to predict the calls with the help of a pre-trained tokenizer. The graphs indicating training accuracy and loss for each of the 4 models have been plotted in Figures 5 to 8.

6. Experiments

The keras in-built tokenizer [16] was used to convert the input clusters into numeric tokens for preparing the train data arrays. This same saved tokenizer is used at the last sequence prediction stage to prepare the test data arrays.

The main hyperparameters are window width, number of clusters to predict and the type of network. Changing the window width did not alter the accuracy much, so a width of 5

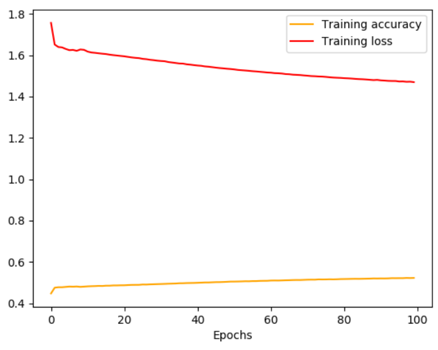


Figure 5: *Unidirectional LSTM model with the plots of training loss(top) and training accuracy(bottom)*

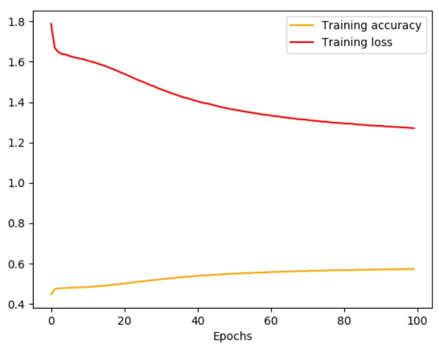


Figure 6: *Bidirectional LSTM model with the plots of training loss(top) and training accuracy(bottom)*

was fixed. The number of clusters was selected as 21 for easier visualization. This number can be altered. The cluster input file for the language model was created from all audio samples via the recordings between 1985 and 2010.

The predicted sequence length is user defined.

7. Results

A snapshot of how the input and predicted sequence look like is shown in Figure 9. The input sequence is shown on the left, and the predicted output sequence of length 3 is on the right. Figure 10 shows how sequence lengths of 1 and 2 can also be predicted.

As stated before, 4 different networks were trained from scratch and used to make predictions. The best model, accuracy and loss for all 4 networks were saved and plotted from figures 5 to 8.

Any number of clusters can be selected while generating the cluster output file. Lesser the clusters, lower is the vocabulary size and the easier it is to train the language model. Here, 21 was selected as the number of clusters, making the sound patterns range from 0 to 20.

For similar number of neurons used per layer, the bi-directional LSTM model gave the best accuracy (approximately 58%) and the least training loss.

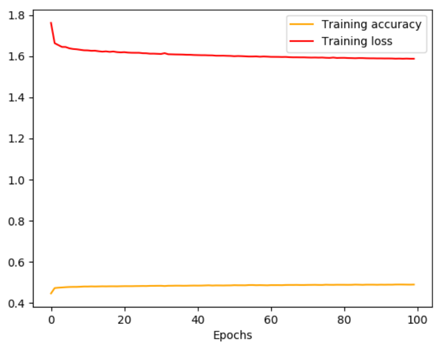


Figure 7: *Unidirectional GRU model with the plots of training loss(top) and training accuracy(bottom)*

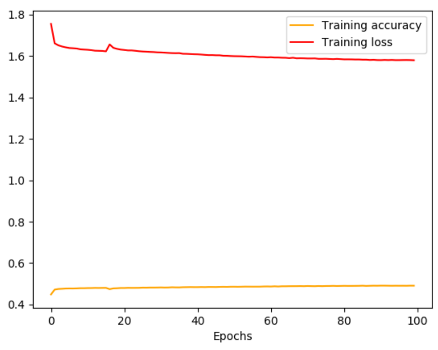
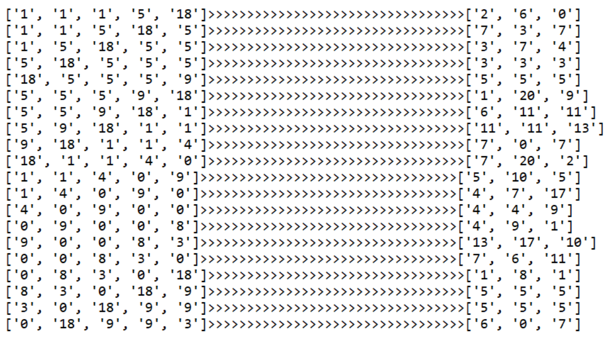
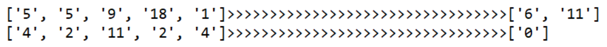


Figure 8: *Bidirectional GRU model with the plots of training loss(top) and training accuracy(bottom)*



```
['1', '1', '1', '5', '18']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['2', '6', '0']
['1', '1', '5', '18', '5']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['7', '3', '7']
['1', '5', '18', '5', '5']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['3', '7', '4']
['5', '18', '5', '5', '5']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['3', '3', '3']
['18', '5', '5', '5', '9']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['5', '5', '5']
['5', '5', '5', '9', '18']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['1', '20', '9']
['5', '5', '9', '18', '1']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['6', '11', '11']
['5', '9', '18', '1', '1']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['11', '11', '13']
['9', '18', '1', '1', '4']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['7', '0', '7']
['18', '1', '1', '4', '0']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['7', '20', '2']
['1', '1', '4', '0', '9']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['5', '10', '5']
['1', '4', '0', '9', '0']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['4', '7', '17']
['4', '0', '9', '0', '0']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['4', '4', '9']
['0', '9', '0', '0', '8']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['4', '9', '1']
['9', '0', '0', '8', '3']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['13', '17', '10']
['0', '0', '8', '3', '0']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['7', '6', '11']
['0', '8', '3', '0', '18']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['1', '8', '1']
['8', '3', '0', '18', '9']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['5', '5', '5']
['3', '0', '18', '9', '9']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['5', '5', '5']
['0', '18', '9', '9', '3']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['6', '0', '7']
```

Figure 9: *Sample predictions.*



```
['5', '5', '9', '18', '1']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['6', '11']
['4', '2', '11', '2', '4']>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>['0']
```

Figure 10: *Two sample(top) and one sample predictions.*

Figure 11 depicts the overall histogram of the triples. The frequency of occurrences of the triples are plotted against their corresponding occurrences. Logarithm (base 10) of the frequency was used instead of the direct frequency value due to large variations between frequencies of low occurrences as compared to frequencies of higher occurrences. Figure 12 illustrates the same histogram, but enlarged for the top nine occurrences and without a log frequency scale. It clearly indicates that most of the triples are less frequent.

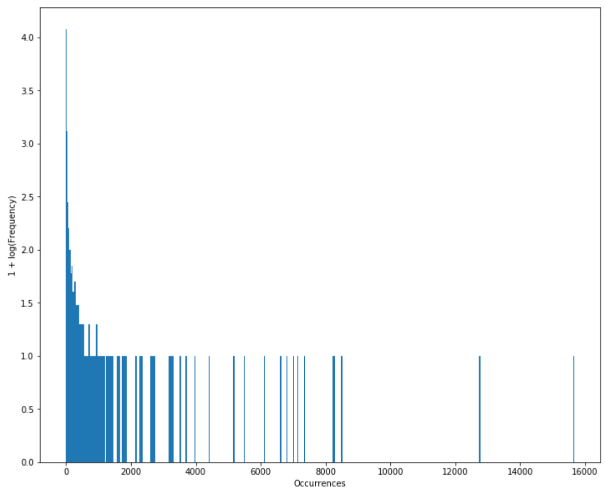


Figure 11: *Overall histogram of occurrences of triples*

Triples	Number of occurrences
(0,0,0)	15653
(20,20,20)	12750
(8,8,8)	8504
(12,12,12)	8275
(2,2,2)	8243

Table 2: *Top 5 most frequent triples and their occurrences*

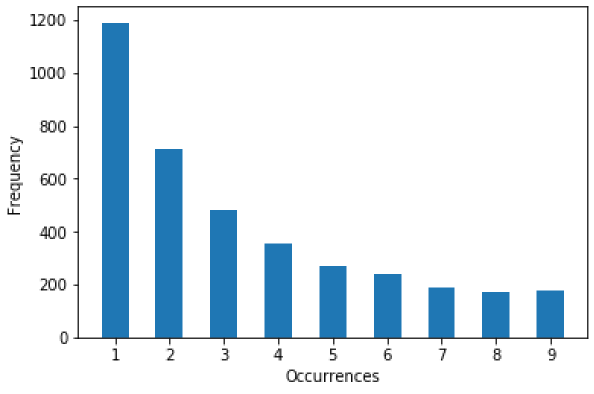


Figure 12: *Histogram of top 9 occurrences*

Table 2 indicates the top five most frequent triples. Triple (0,0,0) was the most frequently occurring triple.

The final feature of the project is that, given an input sequence of sound patterns of any length, a text file containing the following information is generated :

- Which years this sequence can be found in
- Which tape contains the sequence
- The timestamp in the corresponding tape within which this sequence occurs

```
Sequence: 3,4,5


Year: 2009

        Tape: 143A

                Timestamp(s): 1542.9-1544.9

                Timestamp(s): 1586.1-1587.8

                Timestamp(s): 1587.9-1589.9


Year: 1993

        Tape: 300B

                Timestamp(s): 2042.9-2044.5

                Timestamp(s): 2061.9-2063.6

                Timestamp(s): 2065.9-2067.6
```

(a) Without time threshold

```
Sequence: 3,4,5


Year: 1993

        Tape: 300B

                Timestamp(s): 2042.9-2044.5

                Timestamp(s): 2061.9-2063.6

                Timestamp(s): 2065.9-2067.6
```

(b) With time threshold of 20 seconds

Figure 13: *Information retrieved for input sequence (3,4,5)*

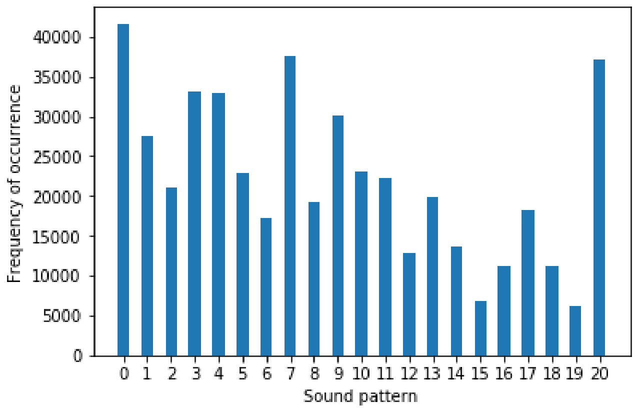
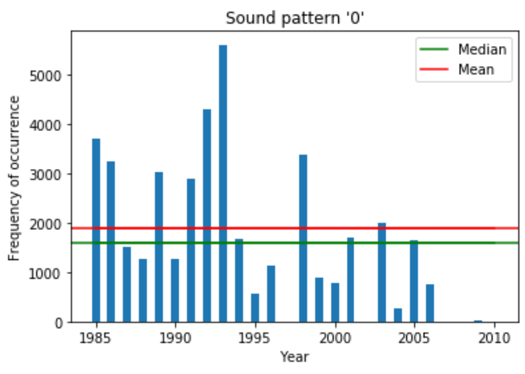
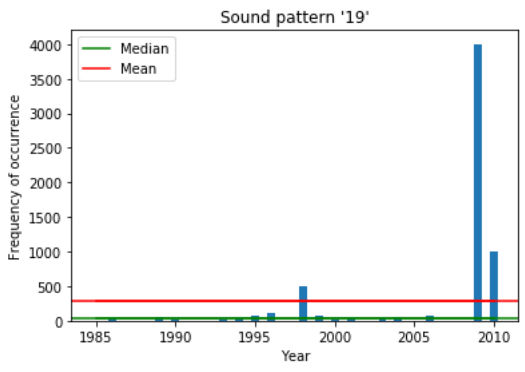


Figure 14: *Histogram of sound patterns and their occurrences*

Figure 13(a) shows a snapshot of the generated text file for the sequence (3,4,5) without an input threshold time. Under a tape, each of the 3 timestamp details corresponds to the start and end time instances within which the corresponding



(a) Frequency distribution of sound pattern '0'



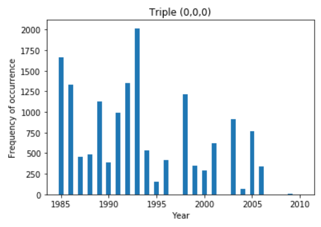
(b) Frequency distribution of sound pattern '19'

Figure 15: *Histogram of sound pattern occurrence statistics vs years of occurrence*

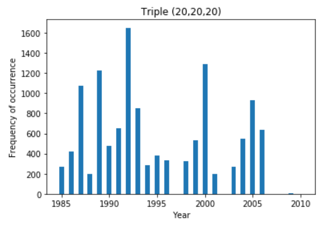
sound pattern is heard. So if the input sequence is (1,2,3,4,5), then each tape would consist of five timestamps.An additional option is to provide a time threshold T_{th} as an input to the script and fetch only those timestamps where each element of the input sequence are within T_{th} seconds of the next element. Figure 13(b) represents the information retrieved for the same sequence (3,4,5) with a threshold time of 20 seconds. We see that this sequence under year 2009 disappears from the text file because the time difference between the beginning of element '4' and the end of element '3' is more than 20 seconds. If a 'n' length sequence is passed as an input to the script, and even one element is more than T_{th} seconds apart from the previous element, the entire sequence is scrapped in the generated text file.

Figure 14 which depicts the histogram of sound patterns vs the frequency of their occurrence indicates that sound pattern '0' occurred the most frequently and sound pattern '19' was the least frequent. Clear differences in the occurrences of these two patterns over the years is illustrated in Figure 15. Sound pattern '19' has mostly been captured in the years 1998, 2009 and 2010, while '0' has been consistently captured over the years. This further explains the low mean and median values for the inconsistent pattern '19' and relatively higher values for the consistent pattern '0'.

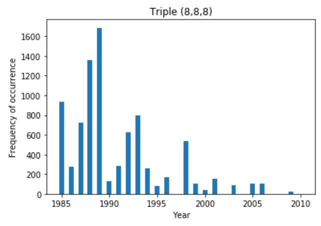
Histograms of the top five triples tabulated in table 2 are illustrated in Figure 16 against the years of occurrence. While triples '0,0,0' and '20,20,20' are consistently distributed over all the years, triple '2,2,2' has a lower distribution in the initial years and triple '8,8,8' has a lower distribution in the final years. Triple '12,12,12' occurred mainly in 2001, and very sparsely in other years.



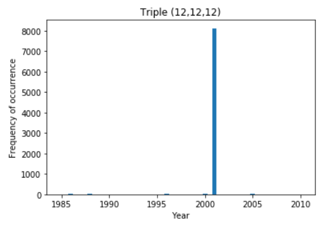
(a) Frequency distribution of triple '0,0,0' over the years



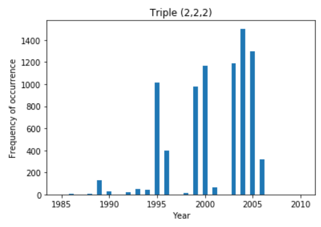
(b) Frequency distribution of triple '20,20,20' over the years



(c) Frequency distribution of triple '8,8,8' over the years



(d) Frequency distribution of triple '12,12,12' over the years



(e) Frequency distribution of triple '2,2,2' over the years

Figure 16: *Histogram of top 5 predicted triples' occurrences vs years of occurrence*

8. Conclusion and Future Work

Overall, the Bidirectional models produced better results than their corresponding unidirectional counterparts, with lower training loss. The performance of LSTMs are seen to be better than GRUs in this experiment despite a small dataset.

There is now a possibility to find specific patterns from information about when a particular sequence occurs, in which tapes and during which intervals of the tapes. These patterns can then be correlated with the corresponding video recordings indicating the action performed by the killer whale. The information retrieval scheme along with mapping audio data to behavior can enable us to learn both syntactic and semantic patterns, thereby leading to better decoding of animal language.

This work has proven that the principles of Deep NLP can be applied to less intelligent creatures and achieve relatively good or even better results due to the limited vocabulary size of animal language/actions as compared to humans.

Pattern discovery algorithms inspired by Starner and Herzing [17] can be used to analyze the killer whale sounds and extract meaningful features that a person might miss or overlook.

Studying behavioral patterns of animals may be enhanced by comparing the sounds made by them with their video at that time instant of producing the sound. Similar neural networks can be designed to build language models for other animals to improve our understanding of their behavior patterns. This would in turn bridge the gap between both species.

9. References

[1] H. Schröter, E. Nöth, A. Maier, R. Cheng, V. Barth, and C. Bergler, "Segmentation, classification, and visualization of orca calls using deep learning," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 8231–8235.

[2] S. Ness, H. Symonds, P. Spong, and G. Tzanetakis, "The orchive: Data mining a massive bioacoustic archive," *arXiv preprint arXiv:1307.0589*, 2013.

[3] M. Schmitt, "Deep feature learning and clustering - a fully unsupervised approach for identifying orca communication patterns," Master's thesis, 2019.

[4] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[5] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[6] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 2005, pp. 195–200.

[7] W. Cheng, E. Hüllermeier, and K. J. Dembczynski, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 279–286.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[9] T. Lang and M. Rettenmeier, "Understanding consumer behavior with recurrent neural networks," in *Workshop on Machine Learning Methods for Recommender Systems*, 2017.

[10] R. Devooght and H. Bersini, "Collaborative filtering with recurrent neural networks," *arXiv preprint arXiv:1608.07400*, 2016.

[11] P. Fournier-Viger. (2016) An introduction to sequence prediction. [Online]. Available: http://data-mining.philippe-fournier-viger.com/an-introduction-to-sequence-prediction/

[12] M. Nguyen. (2018) Illustrated guide to lstms and grus: A step by step explanation. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

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

[14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[15] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent lstm neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, 2015.

[16] Keras. Text preprocessing. [Online]. Available: https://keras.io/preprocessing/text/

[17] H. Hodson. (2014) Dolphin whistle instantly translated by computer. [Online]. Available: https://www.newscientist.com/article/mg22129624-300-dolphin-whistle-instantly-translated-by-computer/