Learning Language Models of Movie Characters

Oğuz H. Elibol
oguz.elibol@gmail.com

Milad Gholami
milad621@gmail.com

Abstract

We investigate learning language models of individual movie characters. We train a recurrent neural net based model on a large dataset of movie scripts with no character specificity to learn a general dialogue model first. Then, we transfer the parameters from this pretrained model to initialize another model and learn a character specific model from a single show. We measure the performance by using perplexity on the general model and relative change in perplexity by switching out character models for the character specific model. Our results suggest that it is possible to learn and evaluate character specific language models.

1 Introduction

Learning language models of individuals can be useful in a number of applications, such as, creating chatbots with distinct and tunable personalities or styles, automatic evaluation of new movie scripts for consistency of characters, discovering similar characters in different settings, or automatic generation of conversations. In this work we investigate learning language models for individual characters in movies (we use the word "character" to refer to movie characters unless otherwise stated).

There is extensive prior work on learning language models, and it is important in a number of classical applications, such as machine translation and speech recognition. Learning language models of characters is challenging because of the limited data on conversations. We approach this problem by learning a general movie dialogue language model from a dataset containing many movies (total of 25,339,337 tokens) then transferring model parameters into another model in which language models for characters are uniquely defined. We demonstrate results by learning from the show South Park, in which main characters have only tens of thousands of words (Cartman: 61k, Stan: 35k, Kyle:31k). In a regular movie the amount of data per character is much more limited hence we picked to work with TV shows. Although we present results on one TV show, the technique we developed can be extended to any show or movie with dialogues.

Specifically, we train a recurrent neural net based model on a dataset containing dialogs from 2496 movies to learn a general model and optimize the performance of this model. Then, we use the learned parameters to initialize a model that has a unique projection layer for each character in the show and use only the data from the show to learn from. Finally we generate sentences using each characters unique projection layer when the model is conditioned on an input sentence. The model can also be used to generate synthetic conversations between characters.

We review the prior art in language modeling, conversation generation and transfer learning in section 2. In Section 3 we provide details about the model we use, and details about the data we used to train our model, and also present the criteria used in choosing data. In Section 4 we provide experimental results and present details on the hyperparameter selection and tuning process, discuss the evaluation of transfer learning, and quantify the performance of language models of characters. We also provide example sentences that were generated by our model in the same section. We conclude by summarizing the results and providing some future directions for the work in Section 5

2 Background and Related Work

There has been extensive work and research on language models. Building on previous work we also employ a Recurrent Neural Net (RNN) based model to tackle this problem based on its success in language modeling as reported in [11]. In addition, we investigate powerful recurrent models such as LSTMs (Long Short Term Memory) [14] and GRUs (Gated Recurrent Units)[8][10] for this task. Different than the prior art, our work focuses on learning unique character models instead of a general language model.

More recently there has been work in modeling conversations using sequence to sequence techniques [15, 17]. Our work utilizes a very similar model as discussed in Section 3, although we use a much more modestly sized model due to limitations in resources, but we add character specificity. As mentioned in [17], although the model performs well on generating conversations there is a lack of consistency and personality. In this work we contribute to the literature by focusing on adding personalities, utilizing different projection layers for each character while retraining the recurrent layer throughout the whole conversation. Although not specifically addressed in our work, the lack of consistency in the conversations can possibly be addressed by using attention based models [7] and will be topic of future research.

We also investigated using parameters transferred from a previously trained model to learn on a different model that captures unique characters. Transfer learning allows a model to perform better than starting from scratch by transferring the knowledge that was learned previously [16]. This has been an active area of research in deep neural nets due to the promise of eliminating the need of retraining from scratch each time. We employ a similar technique in our work and we use a model that has been trained on a large dataset of movies and shows, and transfer those learned parameters into a new model when we are learning language models of individual characters. As discussed in Section 4 this saves significant time in training the character specific model.

3 Approach

We approach solving the problem of learning character specific language models in multiple stages:

1. We learn a general language model of dialogues using a movie script dataset to train a recurrent neural network. This model uses a single projection layer and learning does not account for utterances from different characters but rather learns a general conversational model.
2. We transfer the parameters from the general model and train on specific characters by using data from a movie/show that contains the characters of interest. This model utilizes a separate set of projection parameters for each character. Having a base model allows us to train on specific characters of specific shows without having to retrain on the whole dataset.
3. We use the resulting character specific model to evaluate the language model differences between movie characters by calculating perplexity obtained on a specific characters lines and swapping the character specific projection layer. We also use the model to generate sample utterences conditioned on the previous line.

In this section we discuss the details of the model and dataset that we used in each stage.

3.1 Model

We start by learning a general dialogue language model, then use the learned parameters as initial conditions for learning a fine tune the language model for individual characters. The options and decisions that were made about the model are summarized in this section, and in Section 4 the results supporting these decision are presented.

A language model computes a probability of seeing a series of words together: $P(w_1, ..., w_T) = \prod_{i=1}^{i=T} P(w_i|w_1, ..., w_{i-1})$ [12]. If we restrict the model to the last n word we can approximate the language model as $\prod_{i=1}^{i=T} P(w_i|w_1, ..., w_{i-1}) \approx \prod_{i=1}^{i=T} P(w_i|w_{i-(n-1)}, ..., w_{i-1})$

We use the intrinsic measure of perplexity to evaluate the model. The perplexity is given by:

$$PP(W) = P(w_1, w_2, ..., w_N)^{-\frac{1}{N}} \quad (1)$$

Perplexity in our code was implemented by calculating the mean of the cross-entropy loss and exponentiating it as this gives the same results.

We used pre-trained 100 dimensional GloVe word vectors [13] and allowed backpropagation into the word embedding during training (freezing the layer and random initialization were also tried). Model uses a recurrent neural network with a single 100 dimensional hidden layer GRU cell (LSTM and basic RNN cells were also investigated) and we used a step size of 35 for backpropagating in time (steps sizes 10-45 were evaluated). A projection layer is used with a softmax layer to predict the next word ($\hat{y} = softmax(Uh + b)$, where U is the projection matrix, h is the hidden vector for RNN cell, b is the projection bias). A cross-entropy loss with one hot encoding for the target word ($CE(y, \hat{y}) = -\sum_i^{|V|} y_i log\hat{y}_i$ - where $|V|$ is the vocabulary size) was used during training.

A single projection layer is used while learning a general dialogue model. To learn the character specific language model the parameters except the projection layer (which are the word vectors and GRU parameters) are transferred to another model in which there is a unique projection layer for each character. For example if we had 2 characters in the movie, A and B, we would have two unique sets of projection matrices (U_A and U_B) and biases (b_A and b_B) in this new model. Such a modular model allows us to leverage a pre-trained model to learn on a new movie or show with limited data. Specifically, we investigated the hypothesis that we can capture the character language model by learning different projection parameters for individual characters. The model we employ is illustrated in Figure 1.

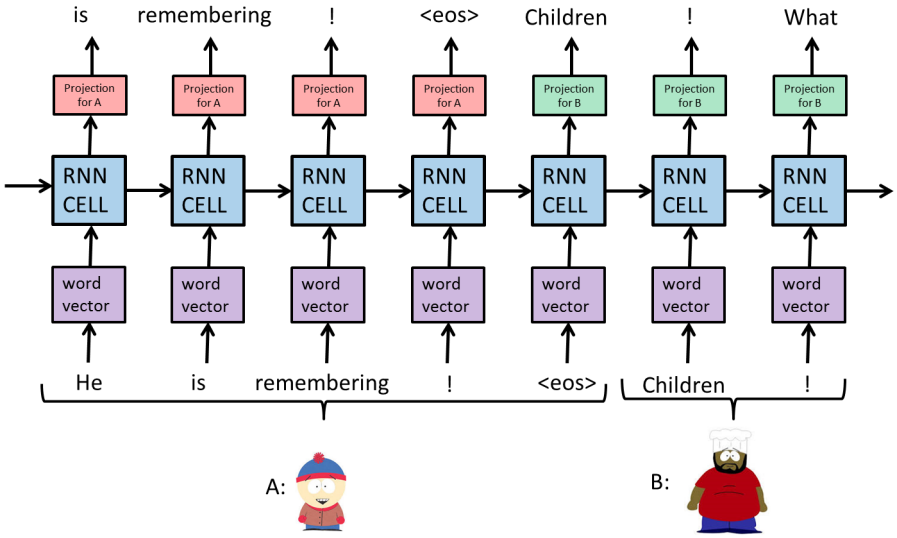


Figure 1: Model used for our experiments for learning a character specific language model. Input words are vectorized using a pre-trained word embedding and the resulting word vectors are fed into a RNN CELL (GRU cell was used). The output word is predicted though the softmax of the projection layer, which is unique to each character. Learning of the unique projection layer is done after transferring parameters from a model that was used to learn from a much larger dataset. A special token <eos> is used to indicate the end of a characters line in the dialogue. A character can utter any number of sentences in their line.

Having a separate projection layer for each character makes our model modular and allows us to transfer weights from a general model. The base model has a single projection unit which learns from conversations without character specialization. Once that model is trained, we can use the parameters to train on specific movies and shows. This eliminates the need to train on the whole conversational dataset all over again, and we limit the dataset only to the movie we are interested.

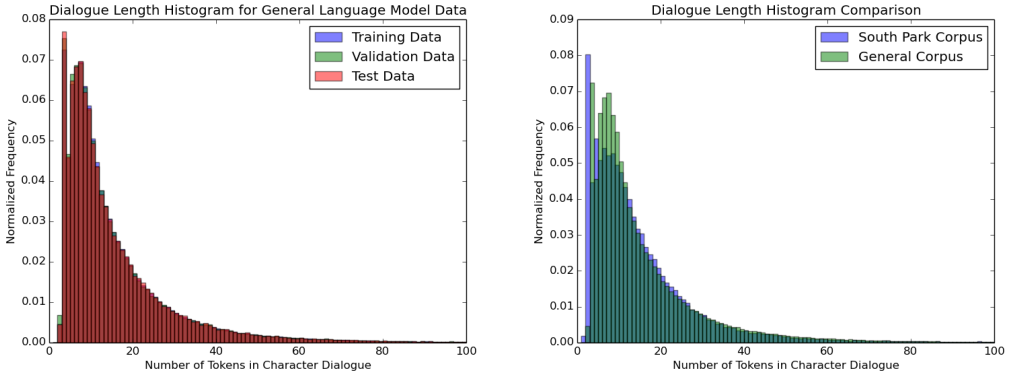
3.2 Data

The dataset was taken from IMSDB [3] which contained 25,339,337 tokens. We denoted the end of the characters dialogue with an <eos> token. Each line can consist of multiple sentences. We picked the dataset because it was the largest available with desired features as discussed below.

The computational complexity of the model is dominated by the number of vocabulary words and the size of the hidden layer ($V \times H$). Thus we decreased the size of the vocabulary by eliminating the tokens that were less than a certain frequency (10 was used in our case) and reduced the size to 34852 unique tokens.

In order to learn individual characters we chose a TV show that had maximal amount of conversations within a relatively few characters. Hence, shows such as Friends [1] and South Park [5] were good candidates because of the length of the shows and consistent main characters that yields high conversation/character. We decided to run our experiments on South Park and learn the language models of the 8 main characters (Cartman, Stan, Kyle, Randy, Butters, Garrison, Chef, others grouped in one).

We also studied the word size of each characters lines in the training dataset for the general language model and the South Park dataset, to ensure that the general language model learned was from a close enough dataset. Figure 2a shows the histogram of the number of tokens in a dialogue for the training, validation and test sets for the general model for movies. We see very similar statistics on the three sets as expected from a well split data set. Figure 2b shows a comparison of the histograms for the training set of the general model and the South Park dataset. We also observe a very similar distribution, although South Park dataset is skewed towards having more conversations with fewer tokens. We went with the assumption that the agreement in the distribution of the datasets is similar enough. We also observe that few characters dominate the conversations (Cartman 16.4%, Stan 9.4%, Kyle 8.4%, Randy 4.1%).



(a) Data used for general language model

(b) Comparison of South Park data

Figure 2: Frequency of the number of token for a dialog in the data sets used in various tasks. (a) Data used to train a general language model - Training Data: 25,339,337 tokens, Validation Data: 3,299,922 tokens, Test Data: 3,143,628 tokens. (b) Data used for learning language models of specific characters in South Park, General dataset: 25,339,337 tokens, South Park dataset: 1,071,136 tokens

4 Experiments

In this section we present data justifying some of the decisions made on the model parameters and the results we obtained on transfer learning and also comparing the language models of characters. Tensorflow [6] was used for implementing the model and most experiments were run on a GPU machine on an Amazon Web Services EC2 instance or on a 4 CPU Intel Xeon machine.

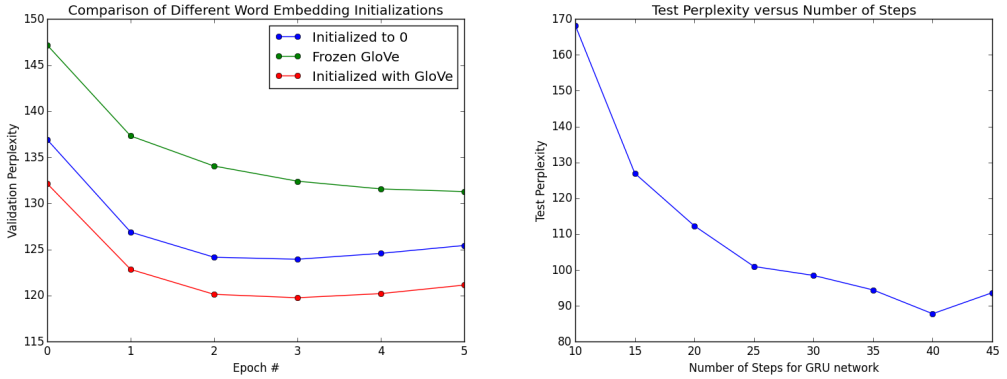
4.1 Model and Hyperparameter Optimization

As shown in the model Figure 1 the RNN cell is a parameter that we have control over and we experimented with using RNN cells, GRU cells and LSTM cells. In past work [9] we have observed that GRU cells perform better than RNN cells, so we focused on GRU and LSTM models on the full dataset. We also experimented with one versus two hidden layers. The results we got indicated that GRU cells performed better than LSTM cells, although this may have been simply due to the lack of proper LSTM tuning. We did pay special attention to tuning, specifically to the forget gate bias and set it to 1 [10], but did not get improved performance. After picking GRU as the RNN cell we also ran a test case with a 2 hidden layer model, however saw an increase in perplexity.

To improve the performance of our model we used pretrained GloVe vectors [4] (glove6B.zip). We chose to use the 100 dimensional embedding after experimenting with 50 and 300. We compared 3 different scenarios on a smaller dataset: - Use the embedding with no changes during training - use the embedding with allowing backpropagation into it and - initialize the word embedding to a random value. We verified the expected result that using pre-trained word vectors with tuning gave the best results. This is expected because back propagation into the embedding will tune the embedding to the specific dataset used. Figure 3a shows the comparison of these cases.

We also experimented with the number of steps to be used for back-propagation in time as shown in Figure 3b. We plotted the text perplexity for each model after training them for 3 epochs on the full dataset and saw that the perplexity leveled off after 35 steps.

Final set of hyperparamets used for the model are presented in Table 1



(a) Word embedding initialization scenarios (b) Number of steps used for backpropagation in time

Figure 3: Model and hyperparameter optimization

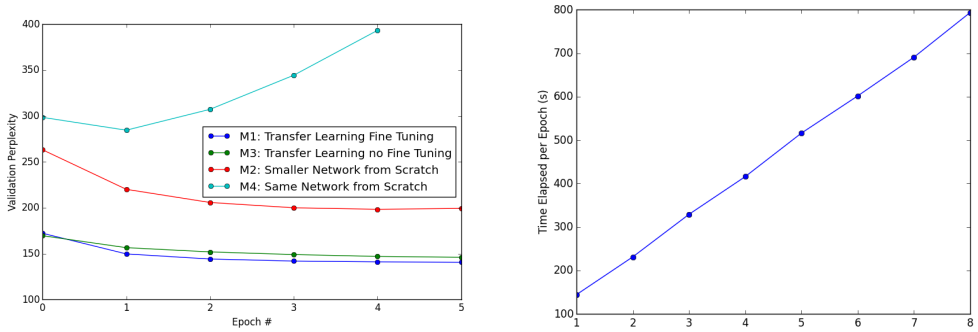
Table 1: Summary of the hyperparameters chosen for the model

RNN Cell	GRU
Hidden Layers	1
Step Size	35
Batch Size	64
Word Vector Size	100
Hidden Size	100
Dropout	0.9
Optimizer	Adam
Learning Rate	0.01

4.2 Transfer Learning

We investigated supplementing the lack of data on individual characters by training on a larger dataset and transferring the parameters over [18]. Literature shows that with no fine tuning (freezing parameters) degrades the performance because simply removing parameters disturbs the co-adapted features. We also observe a similar behavior in which fine-tuning (ability to back-propagate into the parameters) is needed to improve the accuracy of the system as shown in Figure 4a.

It is not feasible to learn the character specific model with the full dataset, because addition of each character increases the training time linearly as shown in 4b, hence transferring over the weights results in significant savings in training time.



(a) (b)

Figure 4: (a) Comparing different initializations using a single projection layer, evaluating the effect of transfer learning by training on the smaller single show dataset. M1: Transferred the parameters and allowed the model to back propagate, M2: Used a smaller network to prevent overfitting on the smaller dataset M3: Transferred the parameter and froze all layer except the projection layer, M4: Used the same network on the smaller dataset with no transfer (b) Training time per epoch of the model as a function of number of unique characters in the model (on AWS GPU EC2) for South Park dataset only.

4.3 Comparing Language Models of Characters

In order to evaluate the character specific language model we ran the model by substituting projection layers of characters with another characters layer and tabulating the change in in perplexity (we expect an increase in perplexity). In Tables 2 and 3, for each character in the first column, we let the model run through the show script but substitute another character language model that is located in the first row. For example, in Tables (2,3), [Row = Cartman, Column = Stan] = (100.86, 80.26) corresponds to the perplexity of the overall model when the language model for Stan is used instead of Cartman's while running the character based model (which result in Stan's projection layer being used for Cartman's lines). Table 2 had a base test perplexity 126.71 and the validation perplexity is given in the table. This model was trained only on South Park data with no parameter transfer. Table 3 was trained again only on South Park data, but this time initialized with pre-trained parameters from the full data. We observe that although the baseline numbers improve, the comparative power of the model for evaluating different characters does not become better. We also observe that as the total number of training data gets smaller for the character it becomes harder to distinguish it from other models. Although we get meaningful differentiation between the models of these characters, there is a clear skew caused by the size of the training data, and fixing this is a part of future work.

4.4 Dialogue Generation

Using our character specific model we generate lines by conditioning the model with an input. The input line is fed into the model to condition the model without monitoring the output. Output line is generated token by token (a token can be a word or punctuation), where the output at each time

Table 2: Validation perplexity on character specific language models after swapping the projection layer of the chacter in the first column with the one in the first row. This model did not use any transfer learning.

	Cartman	Stan	Kyle	Randy	Butters	Garrison	Chef
Cartman	96.34	100.86	101.29	104.74	105.29	111.98	111.33
Stan	96.70	96.34	97.15	99.99	102.02	105.42	104.65
Kyle	96.20	96.58	96.34	99.30	100.58	103.86	102.98
Randy	96.07	96.63	96.59	96.34	97.40	99.05	98.37
Butters	96.01	96.75	96.95	97.39	96.34	99.38	99.41
Garrison	96.13	96.20	96.17	96.30	96.31	96.34	96.44
Chef	95.87	96.18	96.12	96.42	96.93	97.27	96.34

Table 3: Validation perplexity on character specific language models after swapping the projection layer of the chacter in the first column with the one in the first row. With transfer learning.

	Cartman	Stan	Kyle	Randy	Butters	Garrison	Chef
Cartman	77.35	80.26	80.26	82.25	82.73	87.28	85.89
Stan	77.75	77.35	77.89	79.74	80.95	83.43	82.32
Kyle	77.36	77.42	77.35	79.21	79.84	82.24	81.29
Randy	77.28	77.56	77.61	77.35	77.98	79.11	78.49
Butters	77.26	77.70	77.85	78.05	77.35	79.55	79.15
Garrison	77.21	77.25	77.23	77.34	77.33	77.35	77.42
Chef	77.15	77.32	77.31	77.48	77.81	77.99	77.35

step is fed in as the input of the next step until the special token indicating the end of a characters line <eos> is reached. The output is obtained by sampling from the multinomial distribution where the probability of each word is defined as the softmax output of the model [2]. There is also a temperature parameter (logits/Temperature) used to shape the distribution, so with T = 1 we get the standard distribution and with lower temperatures we can give higher importance to the higher probabilities.

Table 4: Output of each character to a specific input at different sampling temperatures

INPUT	i am not sure what is happening . this is not cool ! <eos>		
Character	T = 0.1	T = 0.8	T = 1.0
Cartman	you guys ! <eos>	it billy has gone drank your leg . you know the hundred times up to my name song . by the impossible official jefferson bleeds like science . to be playing people died . he 's been around my life . but maybe i can believe it ... we 're just sick and next unresolved . <eos>	hey kyle . <eos>
Stan	you guys ! <eos>	stop ! <eos>	it 's not ! <eos>
Kyle	you 're not gon na get your ass ! <eos>	everyone stop it ! <eos>	how are come going on with this thing ? ! i wo n't try to get some other thing to get out it , spooked i do n't want you to agree ! you knew butters and stan you have a <unk> ed ? <eos>
Randy	you 're not gon na go ! <eos>	that 's the government n't maintenance cure our for everything . <eos>	stanley , i knew how we got that beer . <eos>
Butters	i 'm not gon na be a little , and i 'm not gon na be a little <unk> . <eos>	it is not wendy . <eos>	oh ... why ? oh that 's a little nice , dad . did ya , i ... turn <unk> out by <unk> , all to the shit is a great nice new night . <eos>

Generating utterences at a low temperature reflects similar perplexities seen for Cartman, Kyle and Stan as the lines generated are very similar. Table 4 shows a table of the sentences generated by characters at various temperatures for comparison. The generated sentences qualitatively reflect the results observed with the character level perplexity comparison (rest of the characters were not included in the table because of space limitations, results shown are representative)

5 Conclusion

In conclusion, we demonstrated the feasibility of learning a character specific language model using recurrent neural nets and showed that it can be used to differentiate between language models of characters. We also confirmed the hypothesis that using large datasets to learn a general model

and transferring these parameters to learn from a smaller set improves the perplexity on character language models. Such a modular approach allows for rapidly learning from new movies without retraining on the full movie dataset.

There are a number of improvement to be made as a part of future work to increase the performance of the model. Although the model was able to differentiate between language models of characters the difference was not very sharp and for characters with fewer lines the performance degraded and this bias will need to be fixed. We also used a relatively modest sized network compared to the conversational model reported in the literature, we believe that with more computational resources, time and better tuning, the performance can be improved substantially. Also, using sequence to sequence models with attention will possibly improve the performance. It would also be interesting to experiment on TV shows/movies with less dialogue data and investigate the effect of the dataset size after transfer learning.

Acknowledgments

We thank the Stanford CS224d Spring 2016 course staff for their valuable guidance.

References

[1] Friends. `http://uncutfriendsepisodes.tripod.com/`. Accessed: 2016-05-15.

[2] Lstm text generation. `https://github.com/fchollet/keras/blob/master/examples/lstm_text_generation.py`. Accessed 2016-05-28.

[3] Movie script database. `http://www.imsdb.com`. Accessed: 2016-05-15.

[4] Pre-trained word embeddings using glove. `http://nlp.stanford.edu/projects/glove`. Accessed: 2016-05-22.

[5] South park. `http://www.imsdb.com/TV/South%20Park.html`. Accessed: 2016-05-15.

[6] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[9] Oğuz H. Elibol and Milad Gholami. Learning language models of movie characters - milestone.

[10] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures.

[11] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Honza Cernocky, and Sanjeev Khudanpur. Recurent neural network based language model. In *INTERSPEECH2010*.

[12] Milad Mohammadi, Rohit Mundra, and Richard Socher. *Lecture Notes Part 4*.

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove global vectors for word representation.

[14] Martin Sundermeyer, Ralf Schluter, and Hermann Ney. Lstm neural networks for language modeling.

[15] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks.

[16] Lisa Torrey and Jude Shavlik. *Transfer Learning - Handbook of Research on Machine Learning*.

[17] Oriol Vinyals and Quoc B. Le. A neural conversational model.

[18] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks.