Triaging Content in a Mental Health Forum

Ali Nazari
Boise State University
alinazaridaftari@u.boisestate.edu

Benedetta Torsi
Boise State University
benedettatorsi@u.boisestate.edu

Abstract

Automatically triaging posts in a forum aimed at users who struggle with mental health can be an efficient way to provide necessary support. We experiment with three models based on different machine learning techniques and adopt distinct feature sets. We observe that the model, which takes into account the sequential property of the dataset, outperforms other models.

1 Introduction

As the conversation about mental health spreads and its stigmatization is reduced, the importance of timely diagnosis of mental disorders becomes apparent. Mental health problems impact our society greatly, thus it is important to find techniques to identify and analyze such issues, but also to take actions. Providing appropriate support can be a determining factor in the survival of individuals who struggle with mental disorders. The CDC reports that since 1999 through 2017, the suicide rate increased 33% (Hedegaard et al., 2018). Among the mental disorders, depression is often associated with increased risk of suicide[1].

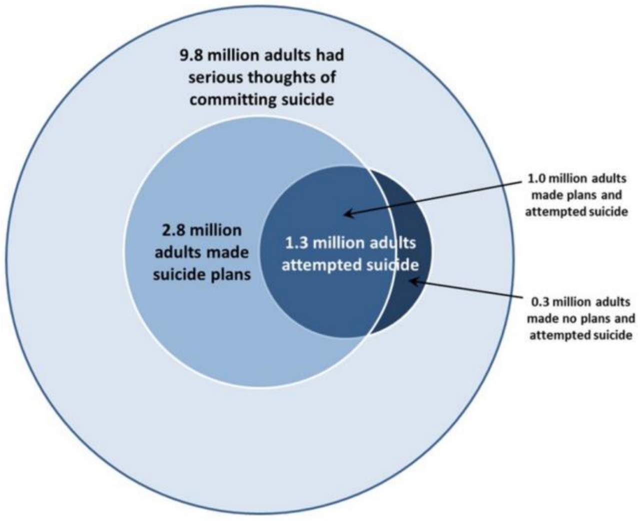

9.8 million adults had serious thoughts of committing suicide
2.8 million adults made suicide plans
1.3 million adults attempted suicide
1.0 million adults made plans and attempted suicide
0.3 million adults made no plans and attempted suicide

Figure 1: Suicidal thoughts among U.S. adults (2017)[2]

2 CLPsych 2017 Shared Task

The CLPsych 2017 Shared Task[3] involves triaging posts from Reachout.com, an Australian mental health forum. On the forum users can anonymously share their experiences. The platform also provides a team of moderators who can intervene to provide support in case of delicate situations such as users expressing suicidal thoughts. As the number of the forum users increase, the moderators have to put in place a triaging system to determine which posts require more immediate attention (Altszyler et al., 2018).The shared task consists of performing the automatic triage of the posts, using the following labels (with increasing priority): *green*, *amber*, *red*, *crisis*. For a description of the annotation process and the ethical regulations required for the task, see Augmenting Online Mental Health Support Services, page 82 (Calvo et al., 2017). The official evaluation metrics for the shared task are accuracy scores. The state of the art for this is presented in the following table.

Team	Recall	Precision	F-measure	Accuracy
Altszyler	0.908	0.903	0.905	0.913
Yates et al.	0.902	0.865	0.883	0.89
French et al.	0.913	0.844	0.877	0.883
Gamaarachchige et al.	0.897	0.829	0.862	0.868
Xia and Liu	0.842	0.871	0.856	0.87
Qadir et al.	0.837	0.87	0.853	0.868
Han et al.	0.875	0.826	0.85	0.858
Vajjala	0.875	0.797	0.834	0.84
Nair et al.	0.793	0.849	0.82	0.84
Miftahutdinov et al.	0.793	0.82	0.807	0.825
Kennington & Mehrpouyan	0.788	0.824	0.806	0.825
Hoenen	0.875	0.647	0.744	0.723
Rose and Bex	0.69	0.575	0.627	0.623

Table 1: CLPsych 2017 results. This metric separates the posts with crisis, red and amber labels from green[4]

The purpose of this project is to predict the

[1] https://www.mentalhealth.org.uk/a-to-z/s/suicide
[2] https://www.nimh.nih.gov/health/statistics/suicide.shtml
[3] http://clpsych.org
[4] http://clpsych.org/shared-task-2017/

degree of urgency of mental condition for members of this depression forum based on their posts. We are using Python 3 programming language throughout this project. We adopt the tools taught in the Natural Language Processing course by Dr. Casey Kennington. After manipulating the data to fit our needs and preprocessing we implemented three main classification models to reach our goal. We start with a Naïve Bayes (NB) model, and then proceed with a simple shallow Neural Network (NN) model. Finally we approach this challenge with a Recurrent Neural Network (RNN) model. The models are evaluated by obtaining accuracy scores for each round of classification. The results will be compared and improved at each step after implementing the advanced classification methods.

3 Dataset

The natural language of the dataset is English. This data was released on May 07, 2017 and provided to our team by Dr. Kennington. The dataset for the shared task consists of 157963 posts. 1188 posts were labeled by human annotators and compose the training set, and 400 labeled posts compose the testing set. We approached this problem as a supervised classification task; therefore only the labeled subsets are used. The data includes:

- Training data: The XML formatted posts that can be used for training the algorithm.
- Testing data: The XML formatted posts that can be used for evaluation.
- labels.tsv: This file contains a tab separated list of post annotations. The columns are: Post (message) ID, Triage label and Fine-grained triage label.
- author_rankings.tsv: This file contains a tab-separated list of author "rankings". The columns are: Author ID and Author ranking on the forums
- author_rankings_summary.tsv: This file contains a tab-separated list of possible user rankings within the forum. It contains Author ranking on the forums, ReachOut status which indicates whether the user is ReachOut-affiliated (staff, a moderator or an invited poster) or a member of the public.

The dataset is quite unbalanced: 60% of the posts are labeled green, 25% amber, 11% red, and 4% are labeled crisis. In order to compensate for this imbalance, we considered generalizing the labels: green instances were labeled with zero, whereas the amber, red, and crisis instances were assigned one. A model trained on such binary labels, however, would not be able to actually triage the posts with the granularity required by the task.

The labeled dataset provided for this task is fairly small; hence we had the idea to use a rule-based approach to extend the green label to the posts written by the authors affiliated with the organization. The assumption would be that the content generated by affiliated authors could be considered as green category.

However, we realized this assumption is not always true. In some instances the post written by affiliated individuals is marked amber or in some cases red. Considering these few cases as outliers, would have increased the dataset size to 16203 posts but the most common baseline (i.e. green label instances) would have increased to 97% as well. We learned that this approach will not improve our models performance since it will create an even more unbalanced data, and consequently the models need to obtain a higher accuracy.

Another approach for handling the small size of the dataset was looking at the history of the authors. The idea was to extract an individual author's activity by looking at the chronology of the posts and try to predict labels for the unlabeled instances based on the noted trend. It is not possible to identify predictable trends in the activity of the users. For example Figure 2 shows the activity of a user in a single day. The posts noticeably fluctuate in the priority within minutes. Thus, it was decided a rule-based approach for predicting the labels based on the history of the author is not a safe practice.

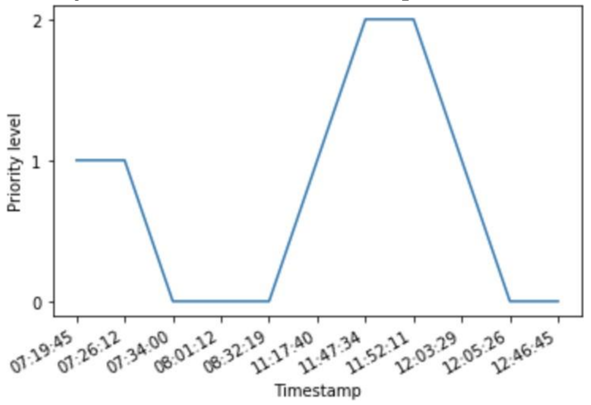


Figure 2: Activity of a user during one day. 0 corresponds to "green" label, 1 to "amber" and 2 to "red"

A further limitation of the dataset was encountered when we attempted to use metadata information as a feature. After examining the testing and training data, it was noted that the ranks assigned to authors in the testing data do not match with the ranks in the training data. For example "*Builder*" rank could be found only in the testing data or "*Mod Squad*" exists only in training data.

The other issue with ranking information is the number of instances of ranks in the two datasets. There are 14 instances of author ranks in training data while there are only 11 instances of the author rank in the testing data. This would cause a mismatch in the input's array-size fed to the models. For the purpose of considering author ranks as a feature set in the neural network model, the number of columns in both testing and training inputs has to match otherwise it is not possible to evaluate the model using the testing data. We had to simplify the ranking labels to resolve this issue. The following assumptions were made in this regard. "*Mod Squad*" equates to "*Mod*", "*Frequent Visitor*" equates to "*Visitor*" and "*Post Mod*" equates to "*Mod*".

The raw data contains informal text in XML format which requires data munging. We need to extract valuable information out of the each file. Minidom was picked to allow parsing the XML files. The Pandas tools will help with data cleansing and storing data in a single dataframe. The data was stored as Pickle files to ease transferring and using the data.

The dataframe was intended to keep the main body of the posts intact for future examinations. BeautifulSoup from "bs4" library was imported to convert the body of posts to string text. Emojis were removed in the parsing of the text. It was decided to not remove the other characteristics during preprocessing, particularly emoticons since they could be helpful to improve sentiment analysis.

4 Models

4.1 Naive Bayes

We started the classification task with a Naive Bayes (NB) classifier. The model was considered one of the baselines along with the most common baseline and the random baseline. This classifier was put forward with the goal of observing what performance could be obtained if multiple features were considered independent of each other. Each instance is a sequence of words (i.e. lemmas, unigrams or tokens) composing a post associated (on the same row of the dataframe) with the rank of the author and the post's triage label. The label is one of the four triage classes, green, amber, red and crisis. The NB classifier models the conditional probability of a label given an instance.

The ranking of the author is useful for this classification task by itself. . The assumption was that authors affiliated to the organization are less likely to generate posts with high priority. For example the probability of finding a post written by an author with "*moderator*" rank to be red or crisis is low. The probability of triaging a post with a particular label given its author's rank is computed and used to classify the posts.

This approach is combined with NB classifier results. Each label's probability is computed using both methods and multiplied for each label per post; then the label with the maximum probability is selected. The assumption is that the words in a post are independent of its author's rank.

4.2 Shallow Neural Network

Preprocessing:

We developed a model to create an embedding per each post using Gensim library. Doc2Vec is adopted for this purpose because it is used to generate representations of paragraphs (Rehurek and Sojka, 2010). The entire training data is included to develop these embeddings which consists of 65'760 posts. The Doc2Vec model converts each post into a vector of size 20. This constituted the first feature set (FS1).

The sentiment of the posts is considered an important element for the classification of the posts. The sentiment scores for each post was computed using vaderSentiment library since it takes into account punctuation, word shape, degree modifiers and some slang words (Hutto and Gilbert, 2014). The result is a set of four scores: *positive, negative, neutral* and *compound* of the first three scores. This feature set in addition to FS1 constructed Feature Set 2 (FS2).

Lastly, we decided to include metadata. In particular, we added the author's rank information. We had to simplify some of author rankings and reduce the count of ranks from 13 to 11 on tainting data as it was pointed out in section 3-Dataset. The SciKit-learn library was utilized to transform these ranking labels to one-hot vectors for each post. This feature in addition to FS2 amounts to Feature Set 3 (FS3).

LabelEncoder and OneHotEncoder from SciKit-learn were used as well to transform the output labels into a categorical array of one-hot vectors.

Defining the Model:

A Neural Network (NN) is implemented and tested adopting different feature sets. Keras library is used to develop the NN. The model is Sequential, which is described in documentations as a linear stack of layers (Chollet and others, 2015). After trying several different activation functions for the NN model, we defined the shallow neural network with a single dense layer of 256 units using a sigmoid activation function.

Following the instructions in the class, we used both binary cross entropy and categorical cross entropy for defining the lost function. The binary cross entropy method resulted in higher accuracy which was in contrast with the fact that the task was not a binary classification. We learned when the user selects binary cross entropy as the loss function and does not specify a particular accuracy metric while compiling the model; Keras infers that the user is interested in the "binary_accuracy"[5]. For the evaluation we needed "categorical_accuracy" which correctly results in lower accuracy compare to binary_accuracy; hence the loss function is defined as "categorical_crossentropy". "adam" optimizer is chosen for this model since it seemed to improve the results.

4.3 RNN

The preprocessing and implementation of the LSTM models were based on a tutorial by Brownlee, 2016 and a tutorial by Thomas, 2018.

Preprocessing:

Keras provides an Embedding layer which converts positive integer representations of words into word embedding (Brownlee 2016). Each unique word in the dataset must be assigned a unique integer index to use the embedding layer. Then the text corpus needs to be reconstituted with sorted integer identifiers (Thomas, 2018). The NLTK tokenizer splits all the posts into tokens then all the tokens are appended together. To form the corpus all the available training and testing posts regardless of being labeled have been incorporated.

The original posts in the labeled training data had to be mapped onto a real valued vector of the unique integers, where each integer is representing a word. The sequence length in each post varies from few tokens to more than 1600 tokens. The longest post in the testing data was 776 tokens and 1616 tokens in the training data. Since the posts have to form an array-shape input, the length of each post is constrained to contain 770 integers. The few posts in the training data longer than 770 characters are truncated and the shorter posts are padded with zero values. The model will learn the zero elements carry no information so indeed the sequences are not the same length in terms of content (Brownlee, 2016).

Data are fed to model generally in small batches. Each batch of data is considered to contain several posts. We tried different batch sizes to see the effect of batch size on the results.

Defining the Model:

The Recurrent Neural Network (RNN) model implemented to represent the instances of classification more accurately as sequences of words. Long Short-Term Memory (LSTM) method is selected for this problem since the size of dataset is fairly small and we did not want to compromise accuracy over time. There are four different types of LSTM models with respect to input and the desired output. These types are many-to-one, one-to-many, many-to-many and one-to-one. We chose the many-to-one approach since our input is characterized by many features and the goal is to obtain one label per instance.

Keras library is utilized to develop the RNN. An LSTM layer is implemented with 100 memory units (a variable parameter) in addition to a dense output layer with four nodes, the same as number of the categories, and a sigmoid activation function to make predictions.

Overfitting is a serious problem in such networks. Large networks are slow to use, making it

[5]*https://stackoverflow.com/questions/42081257/keras-binary-crossentropy-vs-categorical-crossentropy-performance*

difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Avoiding overfitting improves the performance of the neural network. Regularizers such as Dropout allow applying penalties on layer parameters or layering activity during optimization. Dropout randomly sets a fraction rate of input units to 0 at each update during training time, which helps to prevent overfitting.

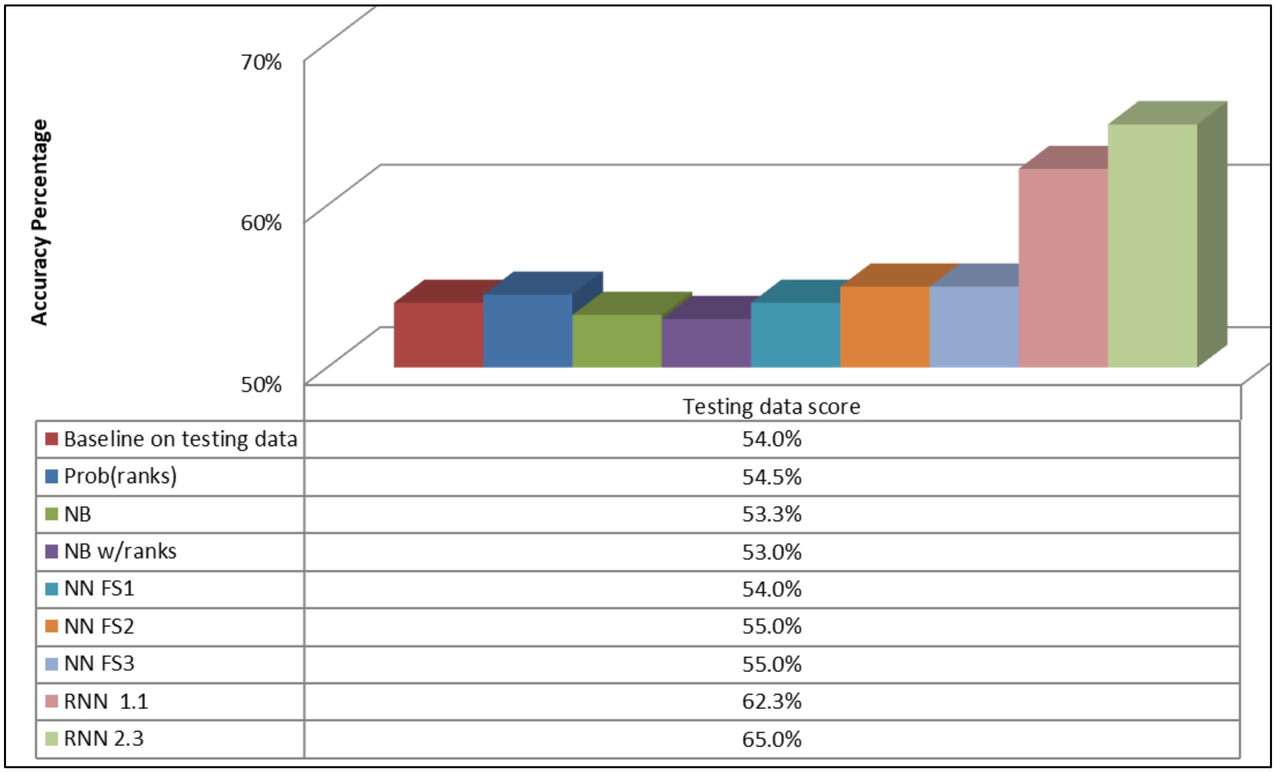
A convolutional (CNN) layer is added ahead of the LSTM layer to observe its effect on the outcome. The data has a one-dimensional spatial structure in the sequence of words and the CNN may be able to pick out invariant features. This learned spatial features may then be learned as sequences by an LSTM layer (Brownlee, 2016). A one-dimensional CNN and max pooling layers is added after the embedding layer which then feeds the consolidated features to the LSTM. A small filter size of 3 is set for the model to not compromise the speed. The pooling layer can use the standard length of 2 to halve the feature map size.

5 Results

5.1 Naive Bayes

The classifier predicted the majority of posted as green category and obtained 53.25% accuracy. Comparing the NB classifier’s performance to most common baseline which is 54% confirms the poor result of this classifier. The relatively small number of instances for other non-green labels in training and testing data sets contributed to this poor performance. Furthermore assuming the words in a post being independent is considered naïve and affecting the results adversely.

Classifying the posts solely base on the probability of the post’s label given its author’s rank resulted in 54.5% which is 0.5% higher than the most common baseline. This result set up the new baseline for the models in future. It proves the necessity of putting author’s ranks to use in this task. Combining the ranking probabilities with the NB classifier did not perform aswe anticipated and dropped the accuracy to 53%.



	Testing data score
🟥 Baseline on testing data	54.0%
🟦 Prob(ranks)	54.5%
🟩 NB	53.3%
🟪 NB w/ranks	53.0%
🟦 NN FS1	54.0%
🟧 NN FS2	55.0%
🟦 NN FS3	55.0%
🟥 RNN 1.1	62.3%
🟩 RNN 2.3	65.0%

Figure 3: Results different classification models for automatic triaging posts. The accuracy percentages are not definite.

5.2 Shallow Neural Network

The NN model was fit to three feature sets. It seemed the results with FS2 and FS3 improved 1% on average compare to FS1 results. Adding metadata features did not improve the system while including sentiment features may have improved the model and predicted other labels besides the most common label, which is essential for the accomplishment of this task. We noted the accuracy score is within 53% to 56% of the testing dataset.

The model was trained for 50 epochs although it was observed changing the parameters of the model did not lead to a consistent and meaningful improvement in the accuracy. The parameters that changed are number of epochs, activation functions, number of layers and number of units in each layer.

We tried to manually initialize the kernel and bias tensors using Keras tools to limit the random behavior of the NN, but this did not improve the consistency of the results. We learned even a small random behavior in the training model propagates into the evaluation results considering the small testing dataset (400 instances). For example if the model in evaluation step predicts only four instances correct compare to the last run, the accuracy will improve 1%. The likelihood of this random change is quite high. When we fit the model to training data, it does not land in the same global minimum on every run; rather it finds a local minimum within an acceptable range of the global minimum.

This randomness in the NN model is referred to as stochastic nature of neural networks. There are several reasons for this behavior. Random order of data observation and random initial weights are some of the main contributors in this case.

5.3 RNN

The RNN model version 1.1 with LSTM layer improved the accuracy by about 10% percent compare to shallow neural network as shown. This improvement owes to the sequential nature of the RNN models. Clearly LSTM has captured the dependency of the features (i.e. words) in the instances of inputs. The accuracy was facing the same issue as the NN model with the consistency and reliability of results due to stochastic nature of neural networks.

Several model parameters were changed to see their effect. Although no meaningful trend was observed in the various accuracies among the models, they were suggesting that using dropout tool generally improves the results.

Using a CNN layer in version 2 of the RNN model seemed to improve the results but it was suffering from inconsistency as well. Our solution was introducing ‘zeros’ initializer for the bias tensor and initialize the kernel weights with a normal distribution as such: mean=0.2 and stddev=0.4. This solution led to improve the results in version 2.3 but did not help with the required consistency of the results.

Model	Version	Testing data score	Model's paramethers				
			embedding_len	batch_len	LSTM_hidden_size	use_dropout	drop
RNN	1	60.50%	30	10	100	F	0.2
RNN	1.1	62.25%	30	10	100	T	0.2
RNN	1.2	62.50%	50	8	100	T	0.3
RNN	1.3	60.50%	50	8	100	T	0.2
RNN	1.4	58.75%	100	8	100	T	0.2
RNN	2.0	61.00%	32	10	100	T	0.2
RNN	2.1	60.00%	32	10	400	T	0.2
RNN	2.2	63.50%	50	8	100	T	0.2
RNN	2.3	65.00%	50	8	100	T	0.2

Table 2: Performance comparison between different models

6 Conclusion

We introduced attempts at compensating the limitation of the dataset used for this task. These limitations had direct effects on the final results of the experiment, yet we were able to conclude meaningful information from them.

Three models were implemented with different machine learning algorithms including Naive Bayes classifier, a shallow neural network, and RNN. Each model was trained on several feature sets. We are confident that the best-performing model is RNN, which takes into account sequential structure of the dataset. Treating the features independent of each other within an instance of input is considered naïve and led to poor results and might not be a valid approach for similar tasks.

The stochastic nature of neural networks forces considering a range for the final accuracy rather than a single definite number. This range is relatively wide for the small testing dataset of size 400.

References

Edgar Altszyler, Ariel J Berenstein, David Milne, Rafael A Calvo, and Diego Fernandez Slezak. 2018. Using contextual information for automatic triage of posts in a peer support forum. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 57–68.

Rafael A Calvo, M Sazzad Hussain, David Milne, Kjartan Nordbo, Ian Hickie, and Peter Danckwerts. 2017. Augmenting online mental health support services. In *Gaming and Technology Addiction: Breakthroughs in Research and Practice*, pages 264–285. IGI Global.

Francois Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

Holly Hedegaard, Sally C Curtin, and Margaret Warner. 2018. Suicide mortality in the united states, 1999–2017. *NCHS data brief*, 330:1–8.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A par- simonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Michelle Morales, Stefan Scherer, and Rivka Levitan. 2018. A linguistically-informed fusion approach for multimodal depression detection. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 13–24.

Radim Rehurek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. http://is.muni.cz/publication/884893/en.

Jason Brownlee. 2016. Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras. Machine Learning Mastery, 26 July 2016, machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/.

Andrew Thomas. 2018. Keras LSTM Tutorial - How to Easily Build a Powerful Deep Learning Language Model. Adventures in Machine Learning, 3 Feb. 2018, adventuresinmachinelearning.com/keras-lstm-tutorial/.