System Integration and Implementation II:
Second Written Report

Tsinghua University, Taiwan, ROC

Spring 2018

TOPIC: Cyberbullying Detection in Social Network platforms

DESCRIPTION: Multilingual tool to detect abusive or sarcastic language signaling cyberbullying

ADVISOR NAME: 陳宜欣 (Chen Yi-Shin)

GROUP MEMBER AND INFORMATION:
Eva Virginia Arevalo Rivara (艾怡華)
Cel: 0903200702
E-mail: 93.evar@gmail.com

I. INTRODUCTION:

Bullying is a phenomenon that continuously targets a subject with repeated abuse. The pervasiveness of social media platforms have facilitated communication within and across communities. However, it has also provided a platform for this behavior. This is what we refer to as "Cyberbullying"; ie. the use of violent/abusive speech against a specific target.

Social platforms have indeed provided a way for abusers to infiltrate the personal lives of the targets in a way that wasn't possible before. They have contributed to alienation of interlocutors by removing the context of social interaction from conversations, and thus eliminating the burden of responsibility of one's speech and rendering empathic interactions that generally rule in the realm of personal interaction impossible. A further analysis of this behavior is better suited to the realm of Social Sciences.

However, according to the Anti-Defamation League, although only 28% of students aged 10-18 experienced cyberbullying in 2014, as much as 87% of them have at some point witnessed other people being harassed[1]. 90% of social media-using teens

[1] Statistics on Bullying, Anti-Defamation League, 2013

who have witnessed online cruelty say they have ignored mean behavior on social media, and more than a third (35%) have done this frequently.[2] The abused themselves reported cyberbullied at a rate of 23%.

This prompts a motivation to find a way to automatize this process of detection of cyberbullying, which has been the target of the present project.

II. GENERAL DESCRIPTION:

For this project, we have built an LSTM(Long short-term memory) Neural Network based classifier that is able to identify abusive language in short posts. After training the model the classifier is able to predict if a post from social media uses abusive language or not with 0.94 accuracy.

It also permits to classify abusive, sarcastic, normal and factual(neutral) language with 93%-94% accuracy depending on the label.

This model was trained using 10-fold cross-validation and tested with a set of data.

The architecture has been tested in both English and Spanish, proving that it is portable to different languages.

Implementation and results from tests are further detailed below.

III. BUILDING THE DATASETS:

1. DATA CRAWLER:

I crafted multiple datasets from messages posted on the social media platform Twitter in various languages, namely, French, Spanish, and English. The information was extracted looking for specific keywords.

The construction of the datasets was a little time-consuming due to the fact finding keywords for the hashtags was a little challenging. The nastiness dataset was particularly hard to construct, as creativity was needed to come up with insults and time was spent evaluating those results. Duplicates had to be eliminated and noise had to be filtered to a degree. Sarcastic phrases with less than 4 words were ruled out because they were highly contextual. Reflected sarcasm/insults were filtered searching the database for reflexive pronouns. Tweets in other languages were still present

The sarcasm dataset contains sarcastic language. It was crafted using hashtags such as “#sarcasm” or “#rolleyes”.

The nastiness was crafted using different insults in different languages. I chose

[2] Teens, Kindness and Cruelty on Social Network Sites, by Amanda Lenhart et al.,2011

hashtags that bordered in aggressiveness, such as “#killyourself”, “#drinkbleach” or “#cutYourself”.

The news dataset was extracted by searching tweets from accounts of newspapers, and it is supposed to reflect a much more neutral language.

The control dataset was built extracting any posts, and erasing those that contained some keywords related to sarcasm.

The data crawler is an XML scrapper that stored messages from the social platform Twitter in a MongoDB NoSQL database along with a label indicating which of these groups the tweet belonged to (referred here as label), as well as other attributes such as tweetId, timestamp, etc. for filtering and other purposes.

Below is a table summarizing the tweets obtained in each category, **Table 1.**

	Sarcasm dataset	**Nastiness dataset**	**Control dataset**	**Neutral dataset**
English	21891	17897	21541	15105
Spanish	27534	14240	20675	15279
French	15673	7036	20102	15033

Table 1: Sizes of the datasets collected, according to language and type.

2. PREPROCESSING:

The obtained twitters contained mentions of the form “@somebody”, web addresses, links to pictures and hashtags.

The mentions were replaced by a wildcard <*mention*>, the web addresses were replaced by a wildcard <*link*>, the pictures were replaced by a wildcard <*pic*>.

The hashtags were removed if they were placed at the end of the twitter or followed only by hashtags, links or picture links. The wildcard <*hashtags*> was inserted instead, representing one or multiple hashtags.

The hashtags embedded in the sentence were however kept, only removing #. This is because hashtags occurring mid-sentence often are part of the sentence itself, however, hashtags trailing a sentence are usually only for contextual anchoring purposes.

This preprocessing is needed to filter things that are not part of the language and that are so specific they could introduce noise to our model. Patterns are more easily recognized by generalizing all links, pictures, mentions and hashtags.

3. PREPARING THE DATASETS:

While training the model, I realized that the accuracy for different labels varied greatly when there was a significant difference in the proportion of labels in the dataset. For example, consider the following dataset.

Label	Number of items in dataset	Label accuracy
Sarcasm	25 734	90.68%
Abusive	14 240	63.75%
Control	20 675	91.66%
Fact	15 279	75.40%

Table 2: Correlation of the label accuracy and ratio of items from label.

The label accuracy in the table below **Table 2** is the ratio of correct guesses for the label divided by the total number of instances of that label. The differences between the label accuracy clearly exhibit a relation with the number of items in the dataset.

For this reason, the datasets are filtered, so that the difference between the dataset having less items and the one having the most items is less or equal to 2000.

4. ENCODING:

For the next stage in our process, we need to vectorize each instance of text. In other to do so, we remove all punctuation. Next, we split the sentences into the individual words. We then proceed to build an index of words in the dataset. Next, for each word in this index, we determine its frequency in the dataset. Sorting the index of words in the dataset by frequency, we cut off the top 5% as connector words.

Considering the calculations that will take place afterwards will take more computational resources with higher dimensions, we reduce the size of the word index. The resulting number of words will be referred to as the maximum feature parameter.

Next, we find out the maximum length of these entries. We will build a vector of the size of the longest entry, for consistency. Each dimension or element in the matrix is a real value corresponding to an entry in the index of words. In case the entry is shorter than the longest sequence, we pad with zeros the rest of the dimensions of the vector. An example of a vector might then be [8, 2, 538, 237, 0 , 0,…], where 8 points to the 8th word in the index, 2 points to the 2nd word in the index, etc.

At first, we split the dataset into 3 sets: the training set, the test set and the validation set. The entries are split according to the size of the dataset. The training set comprised 2/3 of the entire data, while the test set and the validation set each comprised 1/3 of the original dataset.

In the next step, we need to build the matrix we will feed to the neural network. The first matrix will have the number of items in the dataset and the maximum length of an entry as its dimensions. The second matrix will have the of items in the dataset and the total number of labels in the dataset as dimensions. **Table 4** below shows an example of the sizes of these matrices for each set for a dataset of size 77 207.

	Data matrix dimension	Label matrix dimensions
Train set	(51729, 63)	(51729, 4)
Test set	(12739, 63)	(12739, 4)
Validation set	(12739, 63)	(12739, 4)

Table 4: Dimensions for the Data and label matrix for each set.

Later on using the 10-fold cross-validation method we used 5/6 of the dataset for training/testing and 1/6 for validation.

IV. WHY LSTM?:

Before moving on to the Architecture of our model, we briefly explain the choice of an LSTM in our classifier. The picture bellows illustrates the difference between units in Recurrent Neural Networks and LSTMs.

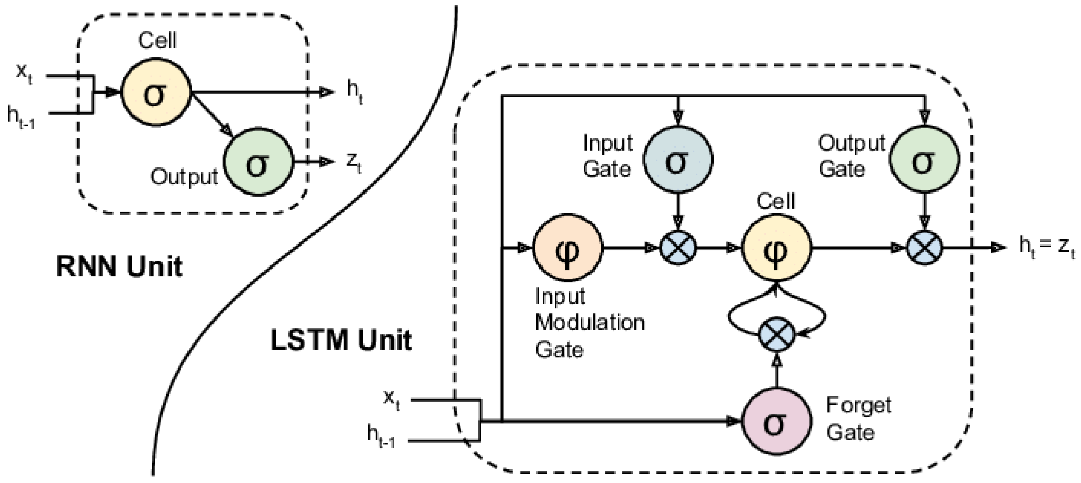


Image 1: Comparison between Recurrent Neural Networks and LSTM units.[3]

[3] Diagram extracted from: Donahue, Jeff & Anne Hendricks, Lisa & Guadarrama, Sergio & Rohrbach, Marcus & Venugopalan, Subhashini & Saenko, Kate & Darrell, Trevor. (2014). Long-Term Recurrent

LSTMs are a type of Recurrent Neural Network that has cell blocks instead of having a single Neural Network layer. Each cell is composed of an input gate, output gate and a forget gate. After applying the activation function (in this case, hyperbolic tangent), the input passes through the "input gate", where a sigmoid operation is applied to it. Then, LSTM will add the data from the input gate to the data in the internal state. Data will then pass to the "forget gate", which also will also apply a sigmoid operation. Finally, as in a normal Recurrent Neural Network, a sigmoid operation is applied before outputting the value.

Summing the input with the internal state allows us to preserve in a certain extent information about the previous state of the Cell. Hence the term Memory in "LSTM",

Applying a sigmoid operation after the hyperbolic tangent reduces the range of the outputs from range [-1,1] to [0,1], thus mapping certain aspects of the input to zero, or "forgetting them" over time. Hence the term short memory.

This type of Neural Network is especially pertinent to text analysis, in which we have a contextual dependency in the sequence of words.

A benefit of LSTM is avoiding the problem of vanishing gradient as well, or not adjusting the weights of the previous elements in the Neural Network. This is important since the focus in language shifts as we speak.

V. CLASSIFIER:

The datasets will be used then to train, test and validate the model.

1. ARCHITECTURE OF THE MODEL:

The Neural Network used in this case will connect 4 layers. The first layer is the Embedding layer, it will extract the features from the dataset for the "real" layers. It will take the data matrix with the prepared above as input and turn it into a three dimensional vector. The third dimension is called the embedding dimension, and it was set to 128 following best practices in similar cases and trial and error.

The Second layer has the purpose of regularizing our model to prevent overfitting. It will just randomly "dropout" to a ratio 0.4 of the new units in the Neural Network. This is with the purpose of obtaining a more general model, and not one that depends/memorizes the particular instances on the train set. Its input and output dimensions are the same.

The third layer is the actual LSTM (Long Short-Term Memory) layer. Its input dimensions are the same as the output of the Embedding layer, determined by the maximum length of the instances of the database and the Embedding parameter. The tensor is reduced to a matrix for output, whose dimension is determined arbitrarily. the LSTM output layer dimension was set to 200 following best practices in similar cases and trial and error.

The activation function used in the LSTM is an hyperbolic tangent activation function. The gradient of this activation function is used to estimate the backpropagated error. For the case when we used 2 labels (abusive and normal), we used logistic sigmoid activation function instead. Dropout was also applied here at 0.2.

The final layer is just a densely connected Neural Network layer, it will apply a softmax activation function to the dot product of the input to this hidden layer from the LSTM layer and the kernel to reduce the size of the output layer to the number of categories or labels, in our case, 4 or 2. I chose a softmax activation function because it uses a probabilistic model instead of a categorical one, which is better suited for our case.

Below is a table summarizing the hidden layers. The output layer's dimension is 4 in the case of the classifier with 4 labels (sarcasm, abusive, normal, factual) and in the case of the classifier with only 2 labels (abusive, normal), only 2.

Layer	Input tensor shape	Output tensor shape
Embedding	(None, max_len)	(None, max_len, 128)
Dropout	(None, max_len, 128)	(None, max_len, 128)
LSTM	(None, max_len, 128)	(None, 196)
Dense	(None, 200)	(None, 4)

Table 5: Summary of the hidden layers in the Neural Network

For the loss function, we use a categorical cross entropy function in the case of 4 labels and binary cross entropy function in the case of 2 labels.

The Neural Network runs with TensorFlow and Theanos for the backend.

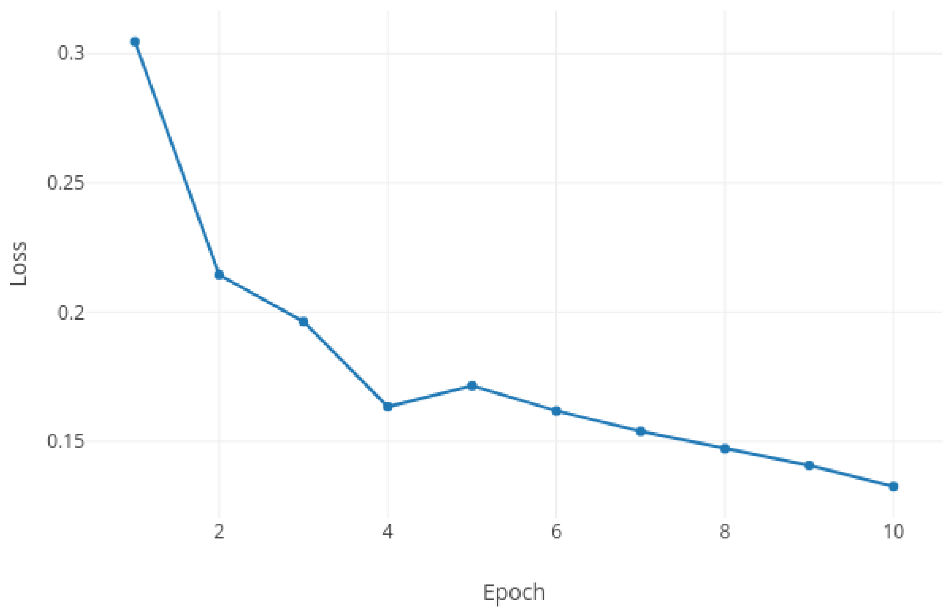
2. TRAINING

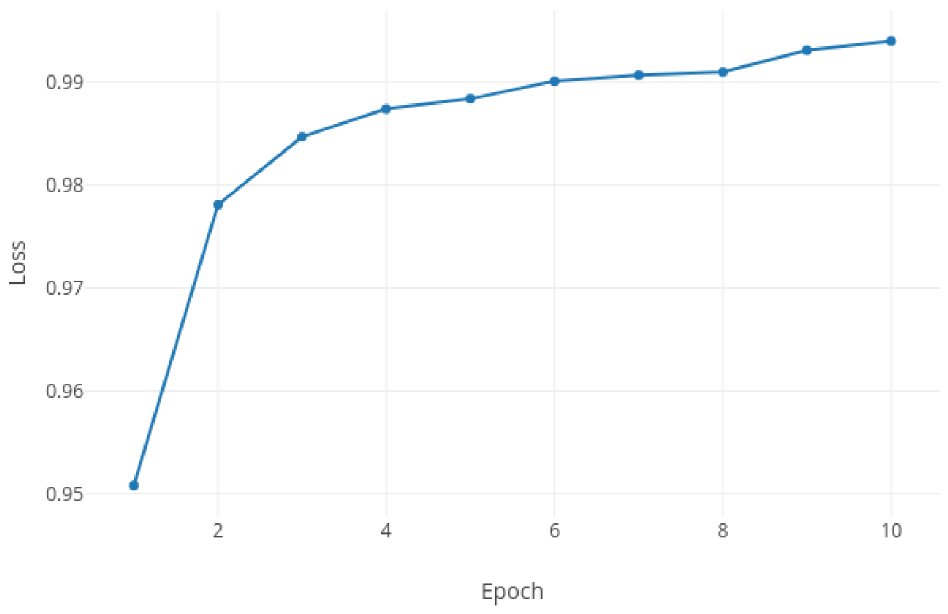
For training, I used the "Adam" algorithm, a gradient-based method for stochastic optimization The batch size was set up empirically by trial and error to 32. The training will be held for a total of 6 epochs, also set up empirically. Beyond 8 epochs, the gain in accuracy is too low, so in order to prevent overfitting, we changed it to 8.

The worst / most dramatic results were by modifying the embedding or LSTM output layer dimensions. In both cases, the accuracy obtained was around 0.92.

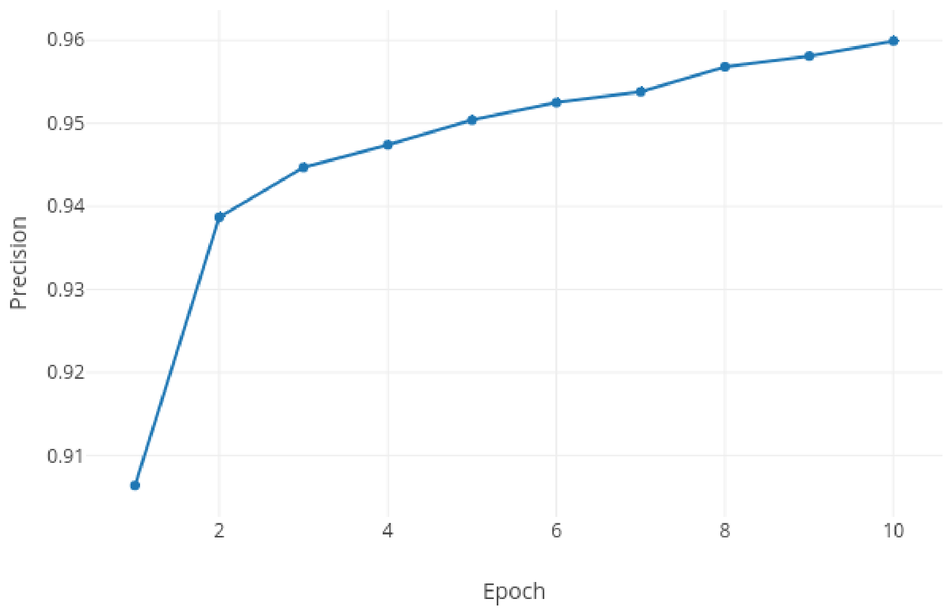
Below are a couple of graphs for the training session of one fold per epoch. They portray Loss, Accuracy, Precision and Recall measurements.

Plot 1.1: Loss by Epoch

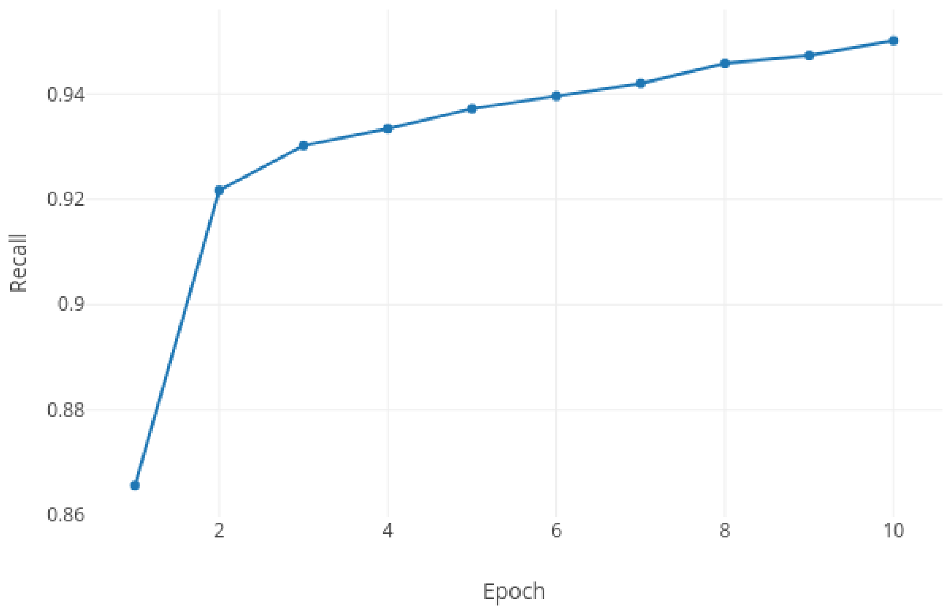




Plot 1.3: Precision by Epoch



Plot 1.4: Recall by Epoch



3. 10 FOLDING CROSS-VALIDATION:

Since the size of the dataset is not very big, I decided to do 10-fold cross validation to get the most out of our data. We have two datasets: the training and the validation dataset. We will partition the training set into 10, train using 9/10 of the set and test using 1/10 of the set. We will repeat this process 10 times, and in each of these the partition selected for testing at the end of the cycle will be different.

Validation is performed at last with the validation dataset.

4. SETTING PARAMETERS

As mentioned above, most parameters were set by trial and error. After training and evaluating a model, the parameters were slightly changed to evaluate the results in loss and accuracy. Below is a summary of the tuning parameters of the Neural Network.

Parameter	**Value**
Maximum of features	4000
Embedding Dimension	128
Dropout rate 2nd layer	0.4
Dropout rate LSTM	0.2
LSTM output dimension	200
Batch size	30
Epochs	8

Table 6. Summary of the tunning parameters of the creation/training of the model

VI. RESULTS:

After the training, we evaluated the model by using the validation test and observing how it classified the data. The results are presented in **Table 7.**

Ta	Correctly tested	Total number	Accuracy
Abusive	2190	2334	93.8303 %
Fact	2281	2451	93.0640 %
Normal	3250	3456	94.0393 %
Sarcasm	3240	3438	94.2431 %

Table 7. Testing results per label after model training

The mean loss and mean accuracy of the models with 2 and 4 labels are listed below.

	2 labels	4 labels
Mean loss	**0.08**	**0.20**
Mean Accuracy	**0.98**	**0.94**

Table 8. Mean loss and mean accuracy

Training in both English and Spanish yield similar results. Training in French was not performed due to time.

VII. PROBLEMS/DIFFICULTIES AND COMPARISON WITH ORIGINAL PROPOSAL:

1. LSTM MODEL

As seen above, the model has a tendency to be more or less accurate guessing a label based on its ratio with other labels in the dataset. That raises a possibility of the existence of a bias, which might affect the performance of the model in real life, since the proportion of sarcasm/abusive language is much less than normal language.

This might lead to more false positives than expected.

2. DIFFERENCES WITH ORIGINAL APPROACH:

Instead of the Embedding layer, the idea was to extract the features using a graph-based approach.

The classifier consisted in comparing graphs extracted from input to patterns already extracted from datasets, in an unsupervised learning schema. Nevertheless, I found it a little difficult to implement and opted for using an LSTM classifier instead, a supervised model, but similar in requirements in terms of datasets.

I originally intending to incorporate the graph part for vectorizing the inputs. However, due to time constraints, I wasn't able to reach these goals.

VIII. CONTRIBUTION:

The model presented in this project is efficient enough to implement a binary classifier of abusive language (either yes or no). The 4-label model is able to further distinguish different "scales" of abusive language.

Previous work in Cyberbullying[4] focus on using Ortony lexicons (eg. Ortony Lexicon[5]), Lists of Profane words or Topic specific Bigrams for text categorization. Other approaches work with Part-of-speech tags that are tightly related to the grammar of a language. This approach, while effective, works only with precompiled lists of word per topic, sentiment and/or language.

This model needs data to be collected for the training, but can is portable to different languages. The implementation proves that the size of the datasets need not be very big as well.

IX. FUTURE WORK:

For future work in the area of Cyberbullying, it would be interesting to explore ways to implement this system. The question of whether or not censure posts with abusive contents poses questions on censorship and freedom of expression. However, giving the option to users to block this type of content gives the option to the person to view that content without having to apply censorship.

Prevention techniques are also to be explored in this case. A possible approaching might be tracking people with tendency to post abusive content and target them as offenders with warnings.

For future work in this specific project, it would be interesting to devise different strategies to embed the sentences that better model the relations in language, such as the graph approach proposed on the first semester.

For future work in the area of language sentiment analysis, a lot of work is being done using LSTMs which are particularly suited for language processing. Other more complex language processing tasks like text comprehension and answering questions about a visual/language context are being explored in **[6]** and **[7]** with a more complex architecture, Dynamic Memory Networks and End-to-end memory networks.

X. WORKFLOW AND ORGANIZATION:

.

1. PAPERS REFERENCED FOR THE RESEARCH PHASE:

[4] Modeling the Detection of Textual Cyberbullying by Karthik Dinakar, Roi Reichart, and Henry Lieberman, 2011

This semester, I continued my goal of reading a paper biweekly. I managed to surpass this goal and read a little bit more than initially expected on this topic and get acquainted with a topic I enjoy. Below are the names of the paper I read and referenced:

1. **C. Hutto and Eric Gilbert, VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text, International AAAI Conference on Web and Social Media, 2014**
2. **Martin Wöllmer, Moritz Kaiser, Florian Eyben, Björn Schuller, Gerhard Rigoll, LSTM-Modeling of continuous emotions in an audiovisual affect recognition framework, Image and Vision Computing, Volume 31, Issue 2, 2013, Pages 153-163**
3. **Hochreiter, Sepp & Schmidhuber, Jürgen, Long Short-term Memory. Neural computation, 1997**
4. **Sundermeyer, Martin & Schlüter, Ralf & Ney, Hermann. LSTM Neural Networks for Language Modeling, 2012**
5. **Z. C. Lipton, D. C. Kale, C. Elkan, and R. C. Wetzel, Learning To Diagnose With LSTM Recurrent Neural Networks, Computing Research Repository, 2015**
6. **Caiming Xiong, Stephen Merity, Richard Socher, Dynamic Memory Networks for Visual and Textual Question Answering, Computing Research Repository, 2016**
7. **Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, Richard Socher, Ask Me Anything: Dynamic Memory Networks for Natural Language Processing, Computing Research Repository, 2015**
8. **Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus, End-To-End Memory Networks, Computing Research Repository, 2015**

2. SCHEDULE FOR THE SECOND SEMESTER:

The **Table X** below show an approximate estimate of the organization of my work this semester.

Task	02.25 – 03.10	03.12 – 03.24	03.25 – 04.07	04.09 – 04.21	04.22 – 05.05	05.06 – 05.20
Research	✓	✓	✓	✓	✓	
Data Collection	✓					

Classifier with Graph approach	✓	✓	✓			
Building the LSTM classifier model			✓	✓		
Getting acquainted with Deep Learning		✓	✓			
Getting acquainted with Deep Learning model tools			✓	✓		
Training the model and getting results				✓	✓	
Applying 10-fold cross validation					✓	
Final report						✓

Table X: Work schedule for Semester2

XI. CONCLUSION

During this process, I have gotten more acquainted with the whole process of doing research on my field and extract knowledge to apply it into other projects. I expect to delve deeper into areas related to Data Science, Natural Language Processing and Artificial Intelligence/Deep Learning in the future, and use the knowledge acquired through this project in my future academic research and personal/professional projects.

XII. ACKNOWLEDGMENTS

Once again, special thanks to PhD candidate Elvis Saravia and my advisor Yi-Shin Chen for helping me with guidance in this project and providing me with the opportunity and the means to carry it.