Mircea Maria-Madalina, ICA, 246/2

Project Documentation

Project: Chrome Extension for Emetophobia Trigger Detection and Substitution

Github: https://github.com/MadalinaMircea/TriggerFinder

Project Description: The project is made up of two parts: the server and the client. The server is a Python (Flask) server that receives calls containing texts from websites and returns the texts censored to remove phrases that can be triggering for emetophobes (people who suffer from a specific phobia of vomiting). The client is a Chrome extension written in HTML, CSS, and JavaScript that makes calls to the server and replaces the texts on web pages with the texts returned by the server.

NLP Task: The server uses an ensemble of artificial neural networks that have been trained to identify triggering phrases,

Application flow:

- The Flask server is started and the ensemble is loaded from files.
- When the webpage loads, the extension replaces all the text with “The text is being examined” and makes calls to the server for each tag on the page.
- The server receives the text and begins to process it.
- The text is processed using a sliding window approach. The window is 4 words wide. This algorithm also ignores HTML tags so that they are not modified. Each 4-word phrase is sent to the next step.
- The text is normalized (see “Normalization”)
- The text is tokenized and lemmatized (see “Tokenization”)
- The text is encoded (see “Embedding”)
- The ensemble is used to predict a value for the phrase.
- If the value is lower than 0.5, the sentence is returned as-is. If the value is greater than 0.5, the letters in the sentence are replaced with asterisks.
- The processed text is returned to the client.

- The client replaces the text on the webpage.

Dataset: The dataset is made up of 600 instances, 300 positive examples (triggering) and 300 negative examples (not triggering). Each instance is made up of up to 5 words. The dataset was collected from Facebook posts over the course of 2 weeks.

Normalization: The white characters on the ends of the text are stripped. The letters are kept and turned lowercase. The spaces and apostrophes are kept. All of the other characters are removed.

Tokenization: For training and prediction, the tokenizer used is the TweetTokenizer from nltk. The set of stopwords from nltk is used to tokenize, with the exception of the word "up", which was deemed too important for the task at hand to remove from the phrases. The lemma of each word is obtained from the nltk wordnet through the "morphy" method.

Embedding: The Stanford GloVe twitter embeddings were used. These word vectors were trained on billions of tweets. The 100-dimensional vector was chosen. This was then transformed into a pickle binary file using Google Colab. The pickle file was utilized from then on out.

Input: Phrases made up of up to 5 words

Expected output: A number in [0,1], 0 if the phrase is not triggering, 1 if it is.

Actual output: A number in [0,1], the input is considered triggering if the output is >= 0.5 and not triggering if it is < 0.5

Machine Learning Models: Many models have been trained and tested and the best accuracy and loss were obtained with a Bidirectional Long Short Term Memory network used for regression.

Other tested models: classification with ANN, LSTM, and BiLSTM with different architectures, regression with ANN, LSTM, and BiLSTM with different architectures.

Choosing the Best Model: The models mentioned above were trained and tested using 10-fold cross-validation. In order to choose the winning model, I computed a weighted average of each model for all of the 10 folds, and the model with the best accuracy and loss was chosen. These accuracies can be seen in the figure below. Nr. 3 was chosen because the accuracy was over 97% with only 0.02 loss.

	A	B	C
1	data_2 glove_100 ann stopwords no up CV epoch_100	0.938333327	0.07811134
2	data_2 glove_100 ann stopwords no up CV epoch_150 lr_0_001	0.939999989	0.063035519
3	data_2 glove_100 bilstm stopwords no up CV epoch_100	0.976666662	0.028680518
4	data_2 glove_100 bilstm stopwords no up CV epoch_200 dropout lr_0_0001	0.965833321	0.040919585
5	data_2 glove_100 bilstm_2_layers stopwords no up CV epoch_200 dropout lr_0_00005	0.962499994	0.040133117
6	data_2 glove_100 lstm_1_layer stopwords CV epoch_100	0.964999998	0.038441581
7	data_2 glove_100 lstm_2_layers stopwords CV epoch_100	0.963333318	0.036721471
8	class glove_100 bilstm stopwords no up CV epoch_150	0.979999992	0.294667984
9	class glove_100 bilstm_2_layers stopwords no up CV epoch_150	0.972499999	0.374570083
10	class glove_100 lstm_2_layers stopwords no up CV epoch_150	0.970000002	0.378273888

Model Architecture: The input layer contains 5 neurons, one for each word in the sentence. If the sentence has less than 5 words, it will be padded with 0 to the right. Then, an embedding layer is used, which encodes the input using word2vec. A bidirectional LSTM layer is next, with 100 units and a dropout rate of 0.25. It is followed by a TimeDistributed layer with 200 units. Then, the data is flattened with a Flatten layer. Finally, the data passes through a fully connected "Dense" layer and into the output layer. This architecture can be seen in the figure below.

```
Model: "model_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_10 (InputLayer)        [(None, 5)]               0
_________________________________________________________________
embedding_9 (Embedding)      (None, 5, 100)            85600
_________________________________________________________________
bidirectional_9 (Bidirection (None, 5, 200)            160800
_________________________________________________________________
time_distributed_9 (TimeDist (None, 5, 100)            20100
_________________________________________________________________
flatten_9 (Flatten)          (None, 500)               0
_________________________________________________________________
dense_28 (Dense)             (None, 100)               50100
_________________________________________________________________
dense_29 (Dense)             (None, 1)                 101
=================================================================
Total params: 316,701
Trainable params: 231,101
Non-trainable params: 85,600
_________________________________________________________________
```

Training: 10 models with the above architecture were trained using cross-validation: Each model split the dataset into 90% for training and 10% for testing. The training was performed for 100 epochs per fold, shuffled, using mean squared error for error computation and adam optimizer. All of the models seemed to learn very fast, within the first 10 epochs, but a number of 100 epochs was still used to ensure the loss would be as low as possible. I tried to use a lower learning rate and a larger number of epochs to slow down the learning process, but the evaluation metrics obtained were lower than the previous ones.

Ensemble Architecture: After choosing the winning model architecture, each one of the 10 trained models was given a weight based on the accuracy it achieved on its fold. The ensemble then loads these 10 models and their respective weights and the predicted value of the ensemble is obtained by computing the weighted prediction of the 10 models.

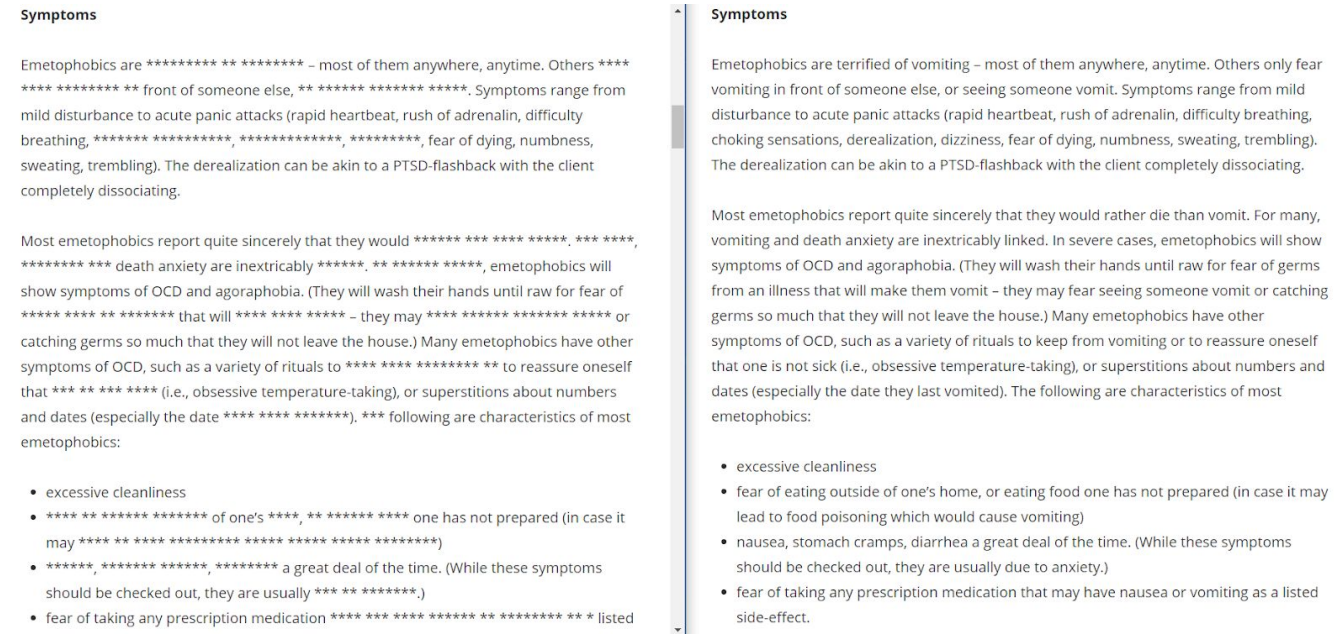
Results: The winning model architecture had an averaged accuracy of 97.6% and an averaged loss of 0.028. The accuracy of each of the 10 models can be seen in the figure below (the first column in the model number, the second column in the weight, and the third column is the accuracy on its fold. The models are ordered ascendingly by accuracy.)

	A	B	C
1	8	0.1	0.933333
2	1	0.1	0.966667
3	2	0.1	0.966667
4	3	0.2	0.966667
5	4	0.2	0.966667
6	9	0.2	0.966667
7	5	0.2	0.983333
8	6	0.3	0.983333
9	7	0.3	0.983333
10	10	0.3	1
11			
12	Total		0.976667

After creating the ensemble, it was manually evaluated on the whole dataset, obtaining the values in the figure below. The “Accuracy” in this computation used the abovementioned threshold value of 0.5 to determine if the predicted value is correct or not.

	A	B
1	Mean squared error	0.006194
2	Root mean squared error	0.078705
3	Mean absolute error	0.027937
4	Accuracy	0.995

In use on an actual webpage, the application behaves like this (the right side is “Before”, the left side is “After”:



Symptoms

Emetophobics are ********* ** ******** – most of them anywhere, anytime. Others **** **** ******** ** front of someone else, ** ****** ******* *****. Symptoms range from mild disturbance to acute panic attacks (rapid heartbeat, rush of adrenalin, difficulty breathing, ******* **********, *********, *********, fear of dying, numbness, sweating, trembling). The derealization can be akin to a PTSD-flashback with the client completely dissociating.

Most emetophobics report quite sincerely that they would ****** *** **** *****. *** ****, ******** *** death anxiety are inextricably ******. ** ****** *****, emetophobics will show symptoms of OCD and agoraphobia. (They will wash their hands until raw for fear of ***** **** ** ******* that will **** **** ***** – they may **** ****** ******* ***** or catching germs so much that they will not leave the house.) Many emetophobics have other symptoms of OCD, such as a variety of rituals to **** **** ******** ** to reassure oneself that *** ** *** **** (i.e., obsessive temperature-taking), or superstitions about numbers and dates (especially the date **** **** *******). *** following are characteristics of most emetophobics:

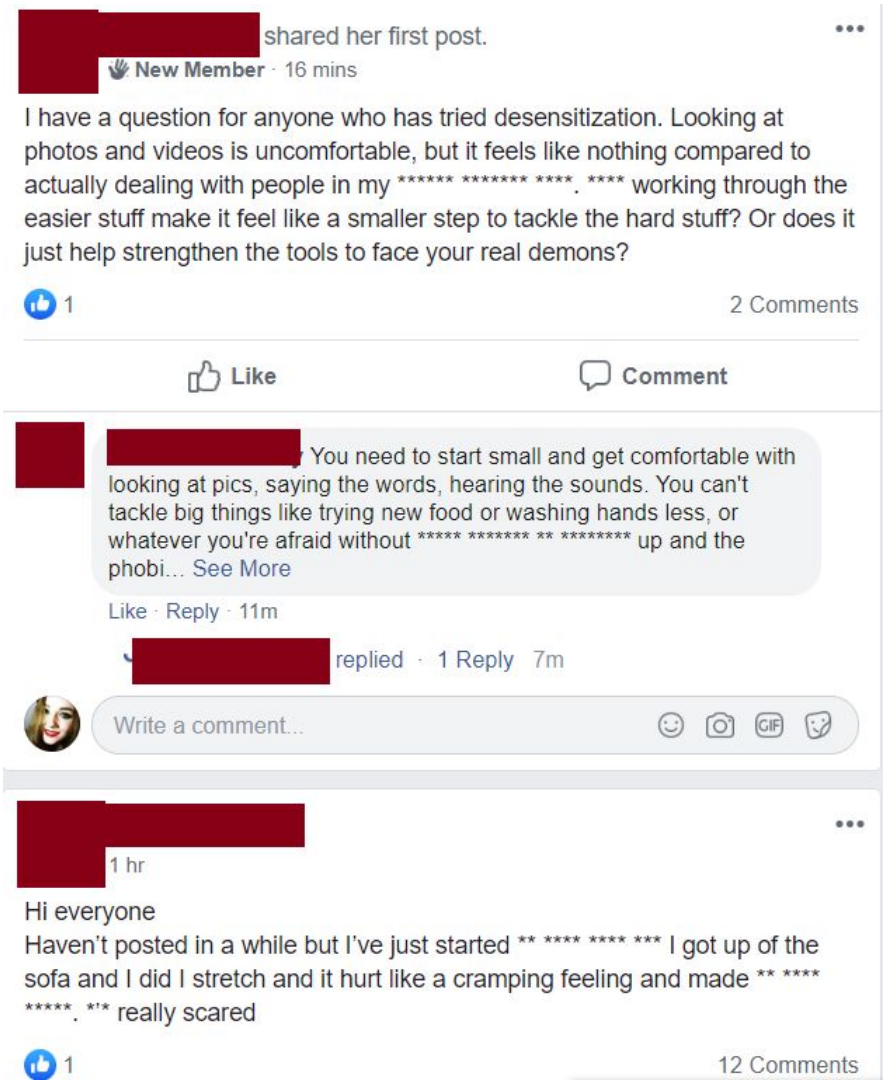
- excessive cleanliness
- **** ** ****** ******* of one’s ****, ** ****** **** one has not prepared (in case it may **** ** **** ********* ***** ***** ***** ********)
- ******, ******* ******, ******** a great deal of the time. (While these symptoms should be checked out, they are usually *** ** *******.)
- fear of taking any prescription medication **** *** **** ****** ** ******** ** * listed

Symptoms

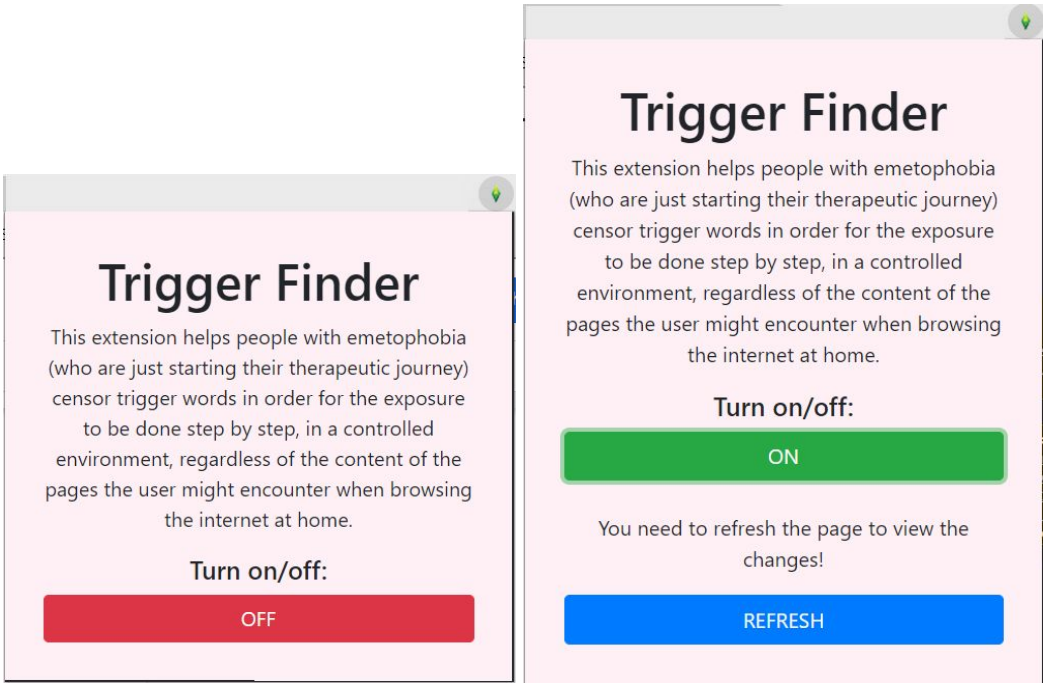
Emetophobics are terrified of vomiting – most of them anywhere, anytime. Others only fear vomiting in front of someone else, or seeing someone vomit. Symptoms range from mild disturbance to acute panic attacks (rapid heartbeat, rush of adrenalin, difficulty breathing, choking sensations, derealization, dizziness, fear of dying, numbness, sweating, trembling). The derealization can be akin to a PTSD-flashback with the client completely dissociating.

Most emetophobics report quite sincerely that they would rather die than vomit. For many, vomiting and death anxiety are inextricably linked. In severe cases, emetophobics will show symptoms of OCD and agoraphobia. (They will wash their hands until raw for fear of germs from an illness that will make them vomit – they may fear seeing someone vomit or catching germs so much that they will not leave the house.) Many emetophobics have other symptoms of OCD, such as a variety of rituals to keep from vomiting or to reassure oneself that one is not sick (i.e., obsessive temperature-taking), or superstitions about numbers and dates (especially the date they last vomited). The following are characteristics of most emetophobics:

- excessive cleanliness
- fear of eating outside of one’s home, or eating food one has not prepared (in case it may lead to food poisoning which would cause vomiting)
- nausea, stomach cramps, diarrhea a great deal of the time. (While these symptoms should be checked out, they are usually due to anxiety.)
- fear of taking any prescription medication that may have nausea or vomiting as a listed side-effect.



User guide: To use the extension, it needs to be loaded in the Chrome extensions, using the developer tools. In order for the calls to the server to be allowed by the Chrome CORS policy, another extension, called Moesif Origins and CORS changer, needs to be installed (this is the simplest way I found so far). The extension is pretty slow on large webpages. The installation and usage instructions will soon be available in the README.md, on github. The chrome extension looks like this:



The On/Off toggle turns the extension on and off. The refresh button is displayed when the on/off status is changed and it refreshes the page when clicked.

Future improvements: The application is very strict with phrases containing words like "stomach" or "throw" in contexts that are not triggering. Thus, the dataset needs to be improved, to include negative mentions of some words and phrases. It also sometimes has trouble with excerpts that begin with a triggering word, probably because most of the instances in the dataset contain the triggers on the last 2 positions. These are, however, improvements that need to be brought to the dataset, not bugs from the model/ensemble itself.

External implementations used: ideas about embeddings and Keras usage from https://github.com/akshayuppal3/language_vision

Tools used: Keras with Tensorflow for model generation and training, the nltk library for the TweetTokenizer and wordnet, numpy, Google Colab for the generation of the word2vec pickle

Internal implementations: I manually implemented the preprocessing, the data shuffle and split, the cross-validation training, the ensemble, the ensemble prediction and evaluation, the sliding window algorithm, the server routes.