VISUAL ANALYSIS OF HARASSMENT CLASSIFICATION IN TWITTER

Jeniffer David
Master of Computer Science
Dalhousie University
jd@dal.ca

Jiarong Cui
Master of Computer Science
Dalhousie University
jr333501@dal.ca

Fatemeh Rahimi
Master of Computer Science
Dalhousie University
fatemeh.rahimi@dal.ca

September 4, 2020

ABSTRACT

Online harassment is becoming prevalent as a specific type of communication in Twitter. Considering the huge amount of user-generated tweets each day, the problem of detecting and possibly limiting these contents automatically in real-time is becoming a fundamental problem. In our project, we develop a Deep Neural Network model to classify the tweets based on four classes: non-harassment, sexual harassment, physical harassment and indirect harassment. We then use Deep SHAP, a unified approach that explains the output of any machine learning model to interpret the predictions of our classifier. We demonstrate the semantic similarity of words in the tweet by visualizing the embeddings in low dimensions using t-SNE. To provide a quick, high level information of the similarities and anomalies between the categories, the final predictions of the model are summarized into different styles of TreeMaps.

1 Introduction

A main area of focus in machine learning is classifier models that can identify toxicity in online conversations, especially in Social Media where online harassment is becoming prevalent as a specific communication type in Twitter. Considering the huge amount of user-generated tweets each day, the problem of detecting and possibly limiting these contents automatically in real time is becoming a fundamental problem specifically for female figures who have been harassed for a long time and Twitter was incapable of helping them. In our project, we focus on the detection of "Sexually harassing", "Physically harassing", "Indirectly harassing" or "Non-harassing" tweets in the data set. We implement supervised learning algorithm where the given tweet is analyzed by a model and then the appropriate tags are applied based on its content.

Before training the model, we pre-process the tweets to remove harmful noise from the data that may interfere with our model. Several techniques for visualizing multidimensional data sets have been studied since, in general, there is no standard mapping into the Cartesian coordinate systems. In spite of several visualization approaches in literature, there are very few examples of applications where text classification methods are used; self-organizing maps are probably the most used technique of visualization for text classification. Current visualization methods provide some form of guidance, visual or non-visual, for the user when training/testing an Automatic Text Classification system. Typically, the user selects the dataset and tunes the values for some parameters of the classification algorithm - which are often difficult to determine a priori. Usually, users do not obtain any intermediate result and have no possibility to intervene during the execution of the algorithm. In this project, we would use the visualization of text (in our case some “keywords”) to clarify which category of harassment the tweet belongs to, to provide some intuition about the correctness of the model.

The key idea of the project, in addition to classifying the tweets, is to explain the classifier itself. The main challenge in deep learning is the intrepretability of the model where the degree of complexity is directly proportional to the number of layers. By making the model more interpretable, we not only increase its transparency, accountability and trustability, but also make it easier for debugging and training. Several libraries like ELI5 implement the LIME algorithm [4] to visually explain black-box machine learning models using different implementations. In our project, we work with the SHAP library which implements a DeepLift algorithm as an approximation of its values suitable for deep learning models.

The predictions of our model are passed to a TreeMap to understand the most frequent words occurring in each class for the purpose of visually analyzing the quality of the prediction. In a TreeMap, each branch of a tree is denoted by a rectangle, which is then tiled with smaller rectangles representing sub-branches. The area of each rectangle is proportional to a specified dimension of the data, and the rectangle is often colored to show a separate dimension or to categorize the various rectangles within the TreeMap. This allows the viewer to easily see patterns that would be hard to spot in bar charts or area charts.

Before sending the input to the classifier, the words are represented in vector space using word embeddings trained on the twitter corpus. This allows us to visualize the similarity between words in the tweets using their euclidean distances. But word embeddings are generally represented in very high dimensions and so it becomes quite a challenge to visualize them in two or three-dimensions without losing much of the information. To address this issue, we utilize the t-Stochastic Neighbor Embedding technique to minimize the loss of information while trying to reduce the dimensions.

In the end, we present a solid project, that not only visualizes the input embeddings that influence the output predictions of our model but also provides an explanation for the behaviour of the model.

2 Related work

Traditional dimensionality reduction techniques like Principal Components Analysis (PCA) and classical multidimensional scaling (MDS) use linear mapping to transform high-dimensional data to low-dimensional data by keeping dissimilar points far apart. But they fail in keeping similar points closer together in a non-linear manifold representation. To overcome this problem, several non-linear dimensionality reductions were proposed, such as Sammon mapping, Isomap, Maximum Variance Unfolding and Locally Linear Embedding to name a few. But they were not efficient when it came to real-world high-dimensional data. In [8], Laurens van der Maaten and Geoffrey Hinton describe a way of converting a high-dimensional data set into a matrix of pairwise similarities by introducing a new technique, called "t-SNE". This technique is capable of capturing much of the local structure of the high-dimensional data very well, while also revealing global structure such as the presence of clusters at several scales.

Although deep learning models excel in several NLP tasks, their internal states cannot be explained easily without spending a considerable amount of time and effort. In [2], Lundberg et al. propose a unified novel approach called SHAP value estimation methods to demonstrate that they are better aligned with human intuition as measured by user studies and more effectually discriminate among model output classes than several existing methods.

When it comes to visualizing hierarchy, the basic idea of node-link visualization for tree structure data come short when the size of nodes increases. TreeMap was the solution for the problem of visualizing the filled hard disk for efficient usage of space invented in 1991. The term TreeMap describes the notion of turning a hierarchical tree structure into a space-filling map. Through the years the basic idea got expanded and led to many different types. Some are: Slice-and-Dice, BinaryTree, Cushion TreeMap, Squarified, Voronoi, Jigsaw, Circular TreeMaps. They all address a specific problem and try to improve the visualization of the standard TreeMap. The basic idea of Slice-and-Dice TreeMap comes from the emergence of thin, elongated rectangles which is difficult for comparing. Squarified TreeMaps try to address this problem by making sure that the rectangles have a lower aspect ratio and also use the space more efficiently. Cushion TreeMaps try to improve the hierarchical identification by using shadows to show the depth.

3 Methodology

3.1 Machine learning

We chose neural networks provided by the Keras library as our classifier model because it gives us the ability to handle large amounts of data for training and validation. The input to the neural network are embeddings learned using the word2vec [3] framework provided by the Gensim library which generates a vocabulary of 483 words where each word or token is a 50-dimensional vector representation of its features. The output of the embedding layer is fed to the next hidden layer of 50 units having RelU activation. The activations of the last hidden layer are fed to a dense output layer which is a softmax of the probability estimates of each class for a given data point. We use binary cross-entropy loss to

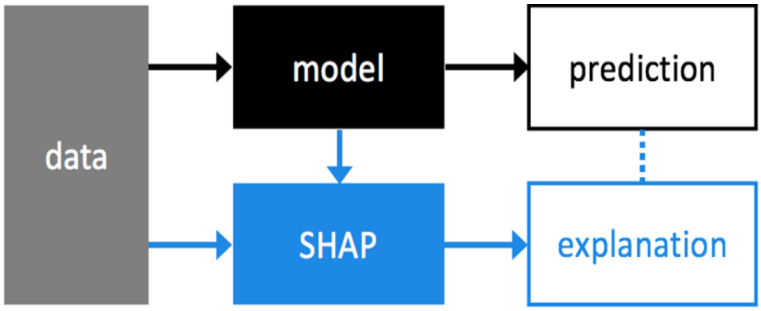

Figure 1: SHAP Explainer

calculate the gradients and optimize the model using Adam optimizer. The trained model weights along with some attributions are the inputs to the SHAP explainer which learns from these weights.

3.2 Visualization

For the visualization, we use three modules: t-SNE for the input, TreeMaps for the output and SHAP for model interpretation.

3.2.1 t-SNE

Consider a high-dimensional data set $X = \{x_1, x_2, ..., x_n\}$. The aim is to map these high-dimensional data points to low-dimensional map points $Y = \{y_1, y_2, ..., y_n\}$. To do this, we convert the Euclidean distances between the two data points in both high and low dimensions to conditional probabilities using their probability density under a Gaussian centered at each point. To overcome the crowding problem and to introduce some symmetry, we calculate the joint probabilities of the high-dimensional data points as: $P_{ij} = \frac{P_{j|i} + P_{i|j}}{2n}$. Then we take a student t-distribution of all low-dimensional map points to calculate their joint probabilities. Finally, we use KL-divergence as the objective function to minimize the cost of mapping the high-dimensional data points to low-dimensional map points and optimize the learning using gradient descent.

For our project, t-SNE tries to map the high-dimensional word embeddings to their corresponding low-dimensional representations to plot them in a 2-D graph. This way we can observe the clusters of words that appearing frequently together since they carry the same meaning semantically.

3.2.2 SHAP

SHAP (SHapley Additive exPlanations) is a unified approach to explain the output of any machine learning model (Fig.1). SHAP connects game theory with local explanations, uniting several previous methods and representing the only possible consistent and locally accurate additive feature attribution method based on expectations [2].

The Deep SHAP algorithm (DeepLIFT + Shapley values) is an enhanced version of SHAP where, similar to Kernel SHAP, we approximate the conditional expectations of SHAP values using a selection of background samples. Lundberg and Lee showed that the per node attribution rules in DeepLIFT can be chosen to approximate Shapley values. By integrating over many background samples DeepExplainer estimates approximate SHAP values such that they sum up to the difference between the expected model output on the passed background samples and the current model output (f(x) - E[f(x)]).

3.2.3 TreeMaps

Three different types of TreeMaps are visualized in the project. In the main page of the project under section 6 we give user the option to choose between Binary TreeMap, Slice-and-Dice TreeMap and Squarified TreeMap. Choosing TreeMap for the visualization of the predictions is because we have hierarchical data in our tweet data set.

Slice and Dice tiling algorithm keeps the order of the values. As indicated in Fig.2, we subdivide either horizontally or vertically. The leaf nodes of the tree are shown in tiles. The first hierarchy which is b3, c3, d10 are divided vertically and the next level hierarchy which is e1, f2, g2,h4 and i4 are divided inside their parent tile but horizontally. And the process goes on until we have all the nodes in the TreeMap.

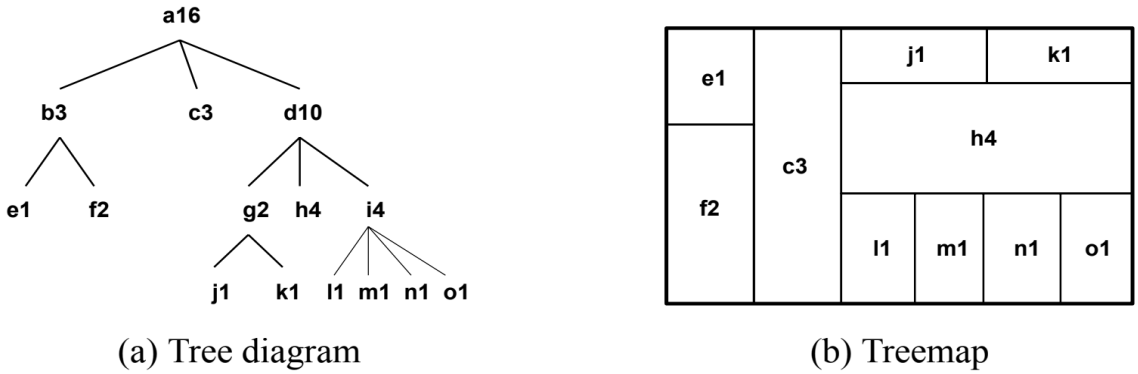

Figure 2: Slice and Dice tiling algorithm.

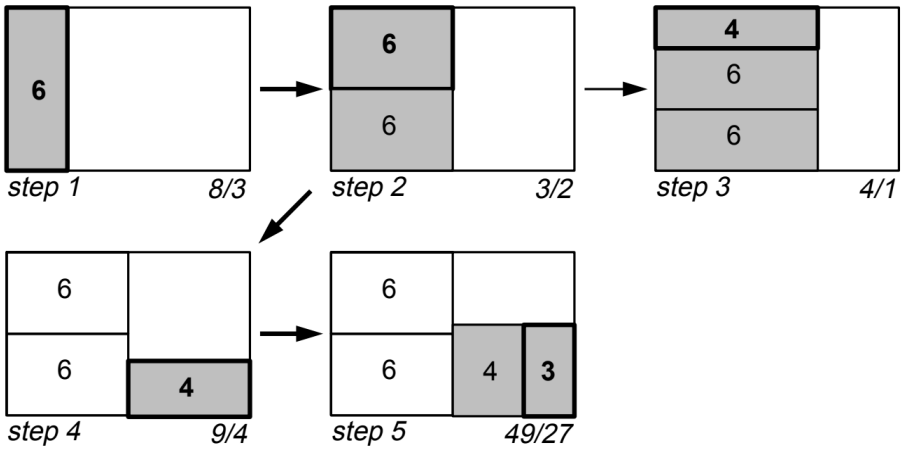

Figure 3: Squarified TreeMap Tiling Algorithm

Squarified TreeMap [1] tiling algorithm is a solution for having rectangles with aspect ratio close to one. The example is indicated in Fig 3. (width is 6 and height is 4). Choosing the division horizontally or vertically comes with the fact of choosing the higher value for width and height in each step. So as in the first step we have a higher value for width, so we divide it vertically. We keep calculating the aspect ratio as per eq. 1 , if aspect ratio gets worse we remove the last tiling and check the width and height and start tiling again with respect to the biggest value of height or width.

$$AspectRatio = \frac{max\,(width, height)}{min\,(width, height)} \tag{1}$$

Squarified TreeMap is better in appearance but it's not suitable in terms of keeping the order of elements. As the order is not important for visualizing the keywords, Squarified would be a good choice. Keywords are the most frequently used terms that appear in the dataset. The frequency is the size of tiles in the TreeMap, meaning the bigger the tile the greater is the frequency of that word in the dataset.

4 Experiments

4.1 Dataset

For the project, we use the data set provided by SIMAH (SocIaL Media And Harassment) competition [5, 6, 7]. This was the first competition on categorizing different types of online harassment language in social media. This dataset has two main classes and three sub-classes with labels indicating if a tweet is harassment or non-harassment and if it's harassment, then it also indicates the type of harassment: indirect, physical or sexual.

Training dataset and validation dataset consist of 6374 and 2125 examples respectively and it is hugely unbalanced. The details of the number of each class is indicated below.

Attributes	Count	Percentage
Non-harassment	3661	57.43%
Harassment	2713	42.5%
IndirectH	55	0.86%
PhysicalH	76	1.91%
SexualH	2582	40.50%

Table 1: The number of Tweets for each class instance in training data.

Attributes	Count	Percentage
Non-harassment	1493	70.25%
Harassment	632	29.79%
IndirectH	71	3.34%
PhysicalH	36	1.69%
SexualH	525	24.70%

Table 2: The number of Tweets for each class instance in validation data.

4.2 Preprocessing

Tweet data usually contain abbreviations, emojis, misspellings, etc.,. Preprocessing facilitates the classifier to emphasize on words that are more relevant for classification. Here are the steps that were taken to reach that goal:

1. Sometimes people start their tweet with “RT” to indicate that they are reposting someone else’s content. As this is not giving us any intuition about the sentiment of the tweets we just delete this word from the dataset.
2. Mentioning starts with "co" in tweets, we make sure to delete both "co" and the next word which is the person who is being mentioned.
3. Finding emojis with regular expression and delete them.
4. Verbs are expanded to their full form (i.e. the verb “i ve” is expanded to “i have”).
5. Replacing abbreviated words such as “idk” to “i do not know”.
6. Removing stop words.

4.3 Framework

The user begins by uploading the training and validation datasets for preprocessing. On completion, the processed tweets are tokenized and converted to bigrams before training the word2vec model. This new w2v model is used to create the word embeddings for the validation dataset. We train the classifier by fixing these word embeddings in the input layers and evaluate the model on the validation dataset.

The embeddings, classifier and the SHAP explainer are all stored in the back-end and retrieved each time for visualization in the front-end through the Flask web application framework. Flask is a web server, so we provide an API in the server-side with related functions. We create API requests from the front-end to the server through HTTP requests in JavaScript. These API calls will get executed on the server and the returned values are used for the different visualizations in the web browser. Fig.4 outlines the flow of data from the user to the model and the interactions of the visual components with the backend server.

4.4 Results

SHAP values of all signalling words for a random tweet is illustrated in Fig. 8. We are using a JavaScript (HTML5) charting library called Anychart. This library has a stacked bar chart which supports negative values as well. This chart plots the positive and negative contribution of each word and shows us the top and worst signalling words for each class denoted by the different colors. The content of the barplot comes from the explanations dervied by the SHAP model on feeding some sample test data with the original prediction. The final interface of our project is shown in Fig.5. Each subsection is shown individually from Fig.6 to Fig.11.

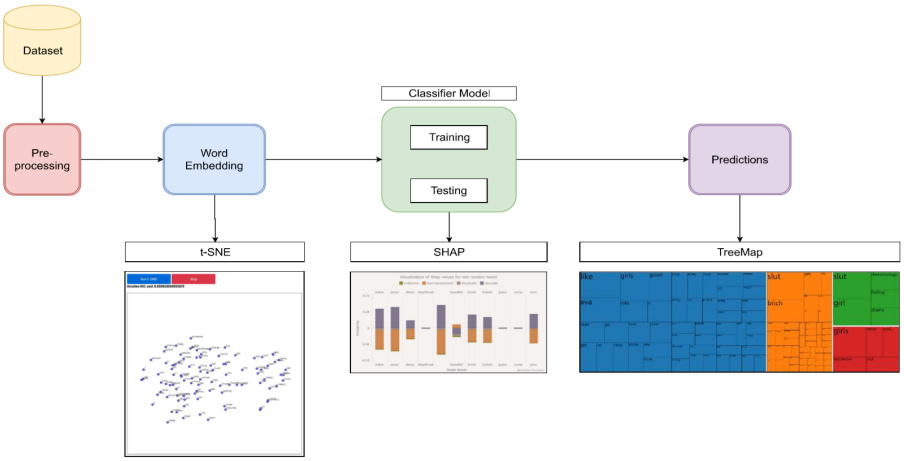

Figure 4: Model Flow Chart

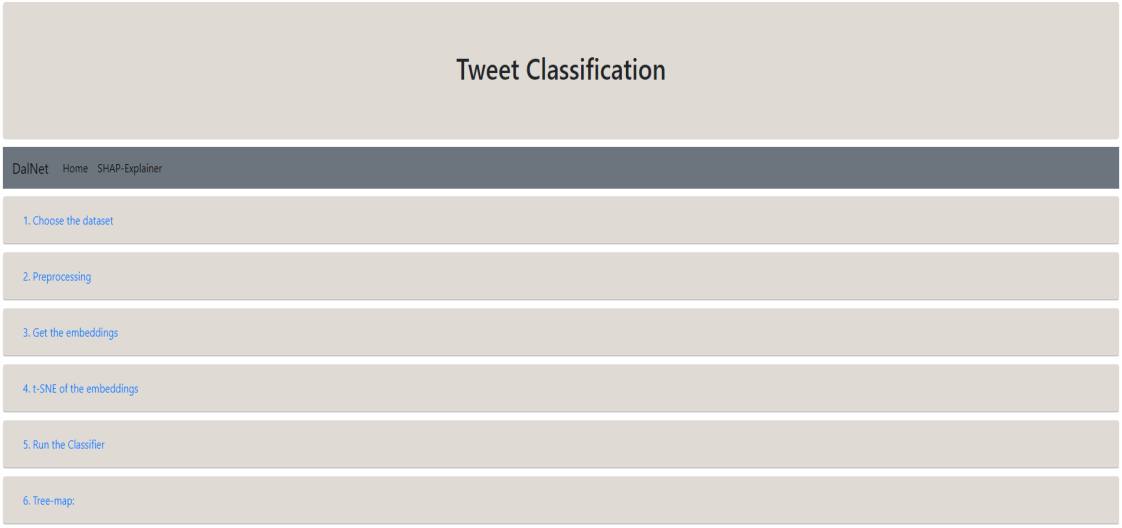

Figure 5: Home page

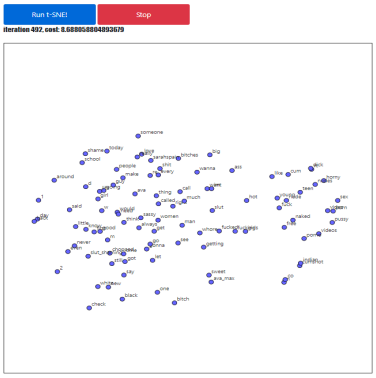

Figure 6: t-SNE of the word embeddings

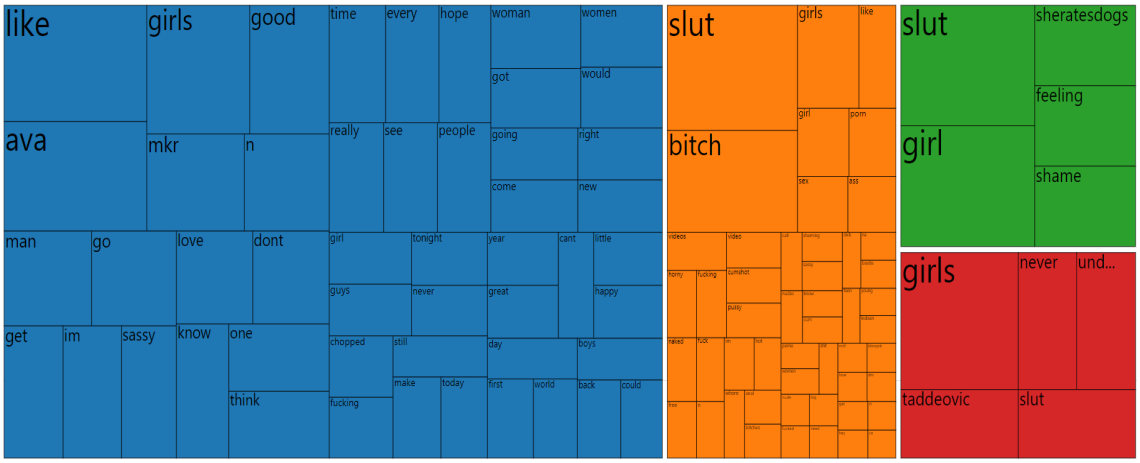

Figure 7: Binary Treemap

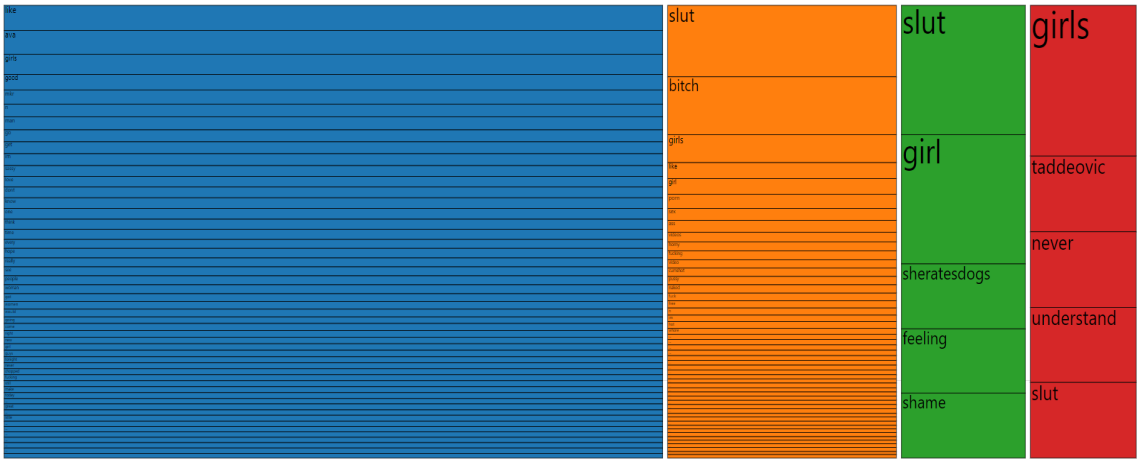

Figure 8: SliceDice Treemap

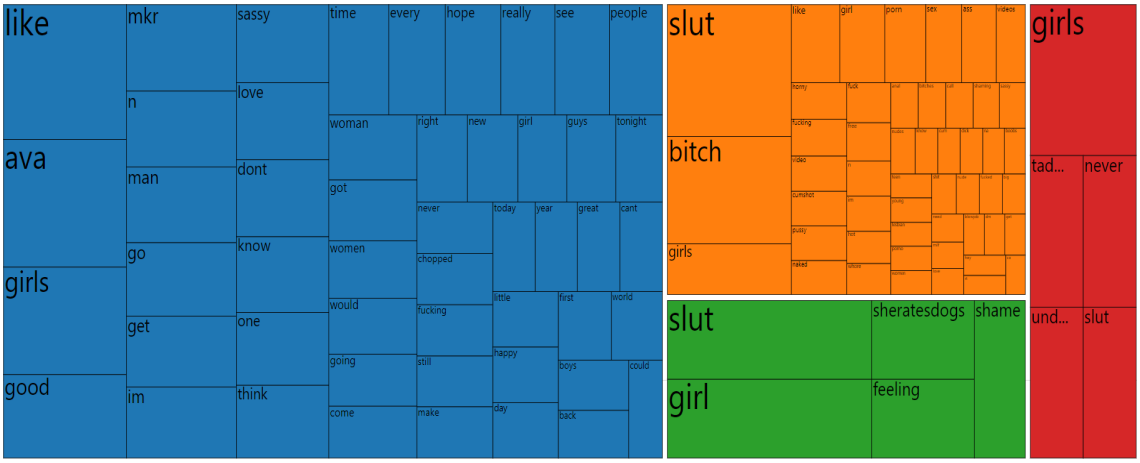

Figure 9: Squarified Treemap

Indirect Harassment	Physical Harssment	Sexual Harassment	Non-Harassment
taddeovic never understand girl s slut shame girls male friends	sheratesdogs feeling also type guy slut shame girl used line	love bby rt like single girl s boob boobs camgirl cumtribute fucked hard fuck hardcor	the girl youhate stubbornn girl sassy soul mailpp someone hate
	kimjonginside girl hasn responded 80 snapchats 4 hours legally obliged call fat slut	noah slut tsundere girl dorky guy pairings	naughtyamerica ava addams francesca le tonights girlfriend
		pics com shauns slut nude teen porn pictures hq full screen drunk girl s anal mid school cock girls skin porn ace	m sorry article balls think great give girl s opportunity see female performer o
		man immediately call girl slut whore bitch etc fucks respect	estroberryboy slut hoe freak got different girl every day week check slut hoe
		yourcutekote mean stupid men consider girl s big breast like slut smart men consider	ve discarded victims girl s color labeled fast precocious survivingrkelly
		sneaky girlfrie1 feeling horny sending surprise anyone retweets slut teenslut bbcslut bigdick whore horny teen fuc	anime fandom check cute girl skimpy clothing love much also anime fandom look
		rasisalilbitch blythehartdowns girl slut shaming acting like grundy stuff archies fault funny	jayleneangelina halloween dress total girl s anything ab

Figure 10: The table shows the predicted labels. Red indicates incorrect prediction. When user clicks on a keyword in the TreeMap, the tweets containing the keyword are automatically displayed in this section under their respective labels.

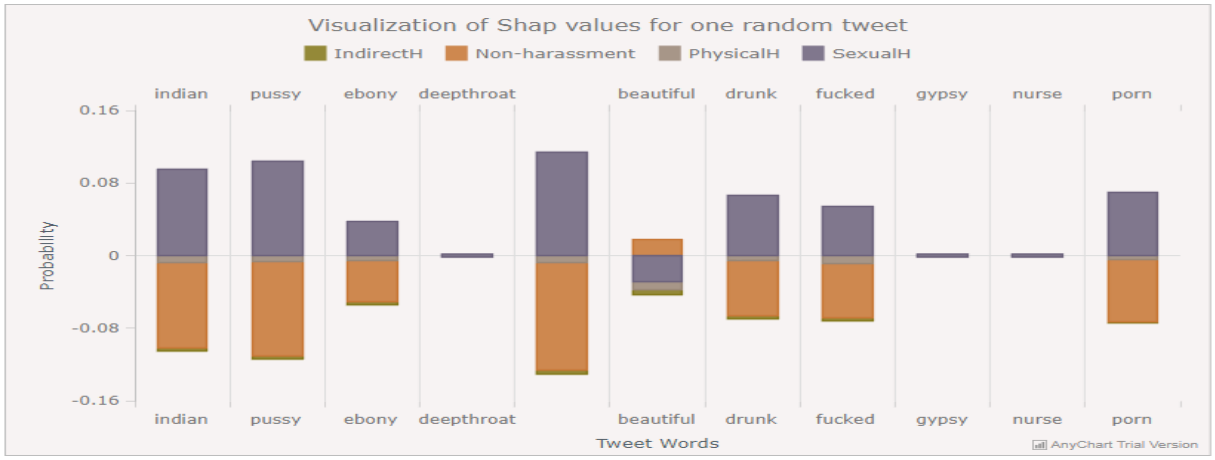

Figure 11: SHAP explainer visualization. The values of the explainer for each word positive or negative is indicated in the stacked bar chart.

4.5 Evaluation

For evaluating the DNN classifier model, sklearn's overall accuracy metric score was chosen. We observed an accuracy of approx. 85% for the entire validation data. We trained the model for 3 epochs with a batch-size of 128 examples.

To evaluate the interpretability of the model, we use stacked bar-plots which uses the shapley values to demonstrate the feature importance assigned to every feature/word in the given tweet denoting the feature contribution. The attributions are the training examples chosen by the user which serves as the background information used by the SHAP model to create explanations.

5 Future work

For future work, we plan to include active learning or Human-Computer interaction component for dynamically training the classifier on input from the user. For example, the user can add new tweets or keywords that were not used for validation and visualize the model on the updated information. We also plan to include more explanatory plots such as summary plots and dependency plots for analyzing the quality of prediction using the feature values.

6 Conclusion

Our project implements an explanatory neural network model for the tweet classification task which visually analyses the input, output and intermediate states of the model using t-SNE, TreeMaps and SHAP respectively. We found the t-SNE to be very useful in mapping high-dimensional information to low-dimensional scatter plots and which it easier to understand the orientation of each word in the dataset. Our idea behind the use of TreeMaps is to give a simple and clear labelling of the predictions and also capturing some hierarchical information among the frequently used words. We understand from the TreeMaps that similar groups of terms always appear in the same class which justifies the embedding model of our inputs. Finally, we train a separate explainer model which identifies the contributions of each feature for the predicted labels and demonstrate the strength of their signal by means of bar plots in the final visualization.

References

[1] Mark Bruls, Kees Huizing, and Jarke J. van Wijk. Squarified Treemaps. In Willem Cornelis de Leeuw and Robert van Liere, editors, *Data Visualization 2000*, pages 33–42. Springer Vienna, Vienna, 2000.

[2] Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. *arXiv:1705.07874 [cs, stat]*, May 2017. arXiv: 1705.07874.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. *Neural information processing systems*, page 9, 2013.

[4] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv:1602.04938 [cs, stat]*, February 2016. arXiv: 1602.04938.

[5] Sima Sharifirad, Borna Jafarpour, and Stan Matwin. Boosting Text Classification Performance on Sexist Tweets by Text Augmentation and Text Generation Using a Combination of Knowledge Graphs. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 107–114, Brussels, Belgium, 2018. Association for Computational Linguistics.

[6] Sima Sharifirad, Borna Jafarpour, and Stan Matwin. How is Your Mood When Writing Sexist tweets? Detecting the Emotion Type and Intensity of Emotion Using Natural Language Processing Techniques. *arXiv:1902.03089 [cs, stat]*, January 2019. arXiv: 1902.03089.

[7] Sima Sharifirad and Stan Matwin. When a Tweet is Actually Sexist. A more Comprehensive Classification of Different Online Harassment Categories and The Challenges in NLP. *arXiv:1902.10584 [cs]*, February 2019. arXiv: 1902.10584.

[8] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

`https://www.anychart.com/products/anychart/gallery/Column_Charts/Stacked_Column_Chart_with_Negative_Values.php?theme=lightEarth`

`https://radimrehurek.com/gensim/`

`https://keras.io/`

`https://d3js.org/`

`https://keras.io/`

`https://radimrehurek.com/gensim/`

`https://cs.stanford.edu/people/karpathy/tsnejs/`

`https://palletsprojects.com/p/flask/`

`https://shap.readthedocs.io/en/latest/`