

Identification and Classification of Toxic Comments on Social Media using Machine Learning Techniques

P. A. Ozoh¹, A. A. Adigun², M. O. Olayiwola³

^{1,2}Department of ICT, Osun State University, Osogbo, Nigeria

³Department of Mathematical Sciences, Osun State University, Osogbo, Nigeria

Abstract- A large proportion of online comments present on public domains are usually constructive, however a significant proportion are toxic in nature. Dataset is obtained online which are processed to remove noise from the dataset. The comments contain lot of errors which increases the number of features manifold, making the machine learning model to train the dataset by processing the dataset, in the form of transformation of raw comments before feeding it to the Classification models using a machine learning technique known as the term frequency-inverse document frequency (TF-IDF) technique. The logistic regression technique is used to train the processed dataset, which will differentiate toxic comments from non-toxic comments. The multi-headed model comprises toxicity (severe-toxic, obscene, threat, insult, and identity-hate) or Non-Toxicity Evaluation, using confusion metrics for their prediction.

Keywords- online comments, toxicity, classification models, TF-IDF technique, logistic regression

I. INTRODUCTION

Over the years, social media and social networking use have been increasing exponentially due to an upsurge in the use of the internet. Flood of information arises from online conversation in a daily basis as people are able to discuss, express themselves and air their opinion via these platforms. While this situation is highly productive and could contribute significantly to the quality of human life, it could also be destructive and enormously dangerous. While discussion or a conversation is opened, it is quite obvious that debates may arise due to differences in opinion. But often these debates take a dirty side and may result in fights over the social media during which offensive language termed as toxic comments may be used from one side. These toxic comments may be threatening, obscene, insulting or identity- based hatred. So, these clearly pose the threat of abuse and harassment online. Consequently, some people stop giving their opinions or give up seeking different opinions which result in unhealthy and unfair discussion. As a result, different platforms and communities find it very difficult to facilitate fair conversation and are often forced to either limit user comments or get dissolved by shutting down user comments completely. This study focuses on building a multi-headed model to detect different types of toxicity like threats, obscenity, insults, and identity-based hate.

Detecting and controlling verbal abuse in an automated fashion is inherently a natural language processing task. Natural Language Processing, (NLP), is a branch of artificial

intelligence that deals with the interaction between computers and humans using the natural language. The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable. Most NLP techniques rely on machine learning to derive meaning from human languages (Deng and Yu¹). Machine learning explores the construction and study of algorithms that can learn from and make predictions on data (Yoshua²). Such algorithms operate by building a model for example inputs in order to make data-driven predictions or decisions, rather than following strictly static program instructions.

Toxic comment classification on online channels is conventionally carried out either by moderators or with the help of text classification tools (Nobata & Tetreault³). With recent advances in Deep Learning (DL) techniques, researchers are exploring if DL can be used for comment classification task. Text classification is a classic topic for natural language processing and an essential component in many applications, such as web searching, information filtering, topic categorization and sentiment analysis (Aggarwal & Zhai⁴). Text transformation is the very first step in any form of text classification. The online comments are generally in non-standard English and contain lots of spelling mistakes partly because of typos (resulting from small screens of the mobile devices) but more importantly because of the deliberate attempt to write the abusive comments in creative ways to dodge the automatic filters.

The contributions of this paper are as follows:

1. This research work will collect and differentiate toxic classified comments from non-toxic comments.
2. This paper will develop a multi-headed model to detect different types of toxicity.

II. LITERATURE REVIEW

Aggression by text is a complex phenomenon, and different knowledge fields try to study and tackle this problem. This analysis of related work focuses on a computer science perspective of aggression identification, a recent emerging area. Currently, the scientific study of automatic identification of aggressive text, using information technology techniques, is increasing. In this study, several related literature are used to express different types of aggression. Some of those are hate (Tarasova et al.⁸), cyber bullying (Adamic⁹), abusive language (Nobata et al.³), toxicity

(Hanson¹⁰), flaming (Waseem et al.¹¹), extremism (Kumar et al.¹²), radicalization (Aggarwal & Zhai⁴), and hate speech (Georgakopoulos et al.¹³). Despite the differences between those concepts, previous research can give us insight into methods to approach the problem of identifying aggressive interactions. Attention is given to the automatic detection of hate speech. For example, Georgakopoulos et al.¹³ provides a short, comprehensive, structured and critical overview of the field of automatic hate speech detection in natural language processing.

This research found a few dedicated works that address the effect of incorporating different text transformations on the model accuracy for sentiment classification. Aggarwal and Zhai⁴ shows the impact of transformation on text classification by taking into account four transformations and their all possible combination on news and email domain to observe the classification accuracy. Their experimental analyses shown that choosing appropriate combination may result in significant improvement on classification accuracy. Nobata & Tetreault³ used normalization of numbers, replacing very long unknown words and repeated punctuations with the same token. Haddadi et. al.⁵ explained the role of transformation in sentiment analyses and demonstrated with the help of SVM on movie review database that the accuracies improve significantly with the appropriate transformation and feature selection. They used transformation methods such as white space removal, expanding abbreviation, stemming, stop words removal and negation handling. Other works focus more on modeling as compared to transformation.

In another study, Bojanowski et. al.⁶ used five transformations namely URLs features reservation, negation transformation, repeated letters normalization, stemming and lemmatization on twitter data and applied linear classifier available in WEKA machine learning tool. They found the accuracy of the classification increases when URLs features reservation, negation transformation and repeated letters normalization are employed while decreases when stemming and lemmatization are applied. Qian et. al.⁷ investigated the effect of transformation on five different twitter datasets in order to perform sentiment classification and found that removal of URLs, the removal of stop words and the removal of numbers have minimal effect on accuracy whereas replacing negation and expanding acronyms can improve the accuracy. Most of the exploration regarding application of the transformation has been around the sentiment classification on twitter data which is length-restricted. The length of online comments varies and may range from a couple of words to a few paragraphs.

III. METHODOLOGY

The characteristics of the data collected for this study is analyzed in this section. This consists of data collected using Jigsaw. A dataset of comments from Wikipedia’s talk page edits is also used. Jigsaw analyses Wikipedia comments (i.e. either toxic or non-toxic), and make the dataset available for those who want to further work on the research. The

contribution of Jigsaw is to develop and illustrate a method that combines crowd sourcing and machine learning to analyze personal attacks. This section also discusses text mining and, also the processing of text carried out using the term frequency-inverse document frequency (TF-IDF) technique. The evaluation of the model is done using confusion metrics.

A. Data Collection and Systems Design of the Study

The Conversation AI team, a research initiative founded by Jigsaw and Google are working on tools to help improve online conversation. One area of focus is the study of negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). So far they’ve built a range of publicly available models served through the Perspective API, including toxicity. But the current models still make errors, and they don’t allow users to select which types of toxicity they’re interested in finding (e.g. some platforms may be fine with profanity, but not with other types of toxic content).

A multi-headed model that’s capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate is developed. An example of a data source by Jigsaw is given in Fig. 3.1.

id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
000099793207770f	Explanation	0	0	0	0	0	0
0001039025c0b60f	D'aww! He matches 0	0	0	0	0	0	0
000112807e002fd	Hey man, I'm really r0	0	0	0	0	0	0
000124101c0b037e	*	0	0	0	0	0	0
000129596546e35	You, sir, are my hero0	0	0	0	0	0	0
000254654475e87	*	0	0	0	0	0	0
0002c0c3e8ac0337	COCKSUCKER BEFORE 1	1	1	0	1	0	0
00031b1e55af7821	Your vandalism to th0	0	0	0	0	0	0
00037261536c51d	Sorry if the word 'no0	0	0	0	0	0	0
00040059c2687caa	alignment on this su0	0	0	0	0	0	0
0005300084f9e6dc	*	0	0	0	0	0	0
0005464e18b50b64	bbq	0	0	0	0	0	0

Fig. 3.1 Dataset from Jigsaw

The systems design of this study is given by Fig. 3.2.

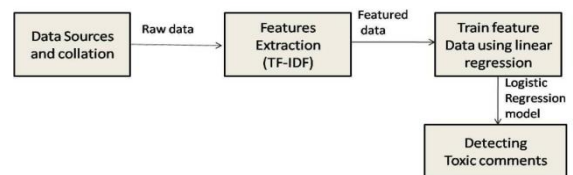


Fig. 3.2 System Design of the Study

B. Description of Proposed Machine Learning Techniques

A brief overview of the Term Frequency-Inverse Document Frequency (TF-IDF) technique, Logistic regression (LR) algorithm, and the confusion matrix are presented in this section. The techniques presented are then used to solve toxicity problems in social media platforms. This section contains a description of these machine learning techniques

1) *Text processing method using Term Frequency-Inverse Document Frequency (TF-IDF) Technique:* The aim of feature extraction is to reduce the dimensionality and eliminate irrelevant features so that efficiency and

performance of the classification algorithms can be improved. Pre-processing helps make for better input data when performing machine Learning or other statistical methods, thus preprocessing aims at converting the free-text review sentences into a set of words and, at the same time, enriching their semantic meaning. There are three subtasks involved in this process. They are part-of-speech tagging, stemming, and meaningful word selection.

The steps for document classification are:

- i. Read Document
- ii. Tokenization, where every word in the document is split into tokens
- iii. Removing stop words, punctuation, or unwanted tokens
- iv. Lemmatization/Stemming, Shorten words to their root stems, e.g. to take walk, walked, walking, walks as Walk etc
- v. Feature vector representation
- vi. Feature extraction
- vii. Learning algorithm

There are three known methods for text processing methods. This include:

Bag-of-words
L²-normalization
TF-IDF

The algorithm of the Term Frequency-Inverse Document Frequency (TF-IDF) is given as follows:

```

Input: Dataset D={D1,D2,...Dn}
Where D->Data set
Di->Document
Output: F={F1,F2,...Fn}
Where F->A set of selected features from D
Fi->is the selected features
Begin
  Load D
  For(i=1;i≤N)// Where N is no of documents
  {
    Perform tokenizing [Di]
    Append the tokens to Tlist
  }
  WC=find length (Tlist) //wc is total no of tokens of the document
  Sc=find (SWlist)
  For(i=0;i<wc)
  {
    For(j=0;j≤Sc)
    {
      If(SWlist (j)==Tlist(i))
      {
        Slist={s1,s2,..sn}
        Eliminate Tlist
      }
    }
  }
  Append Rlist//where Rlist is the stop words removed text
  X=length of NS
  For(i=0;i<x)
  {
    Perform stemming Rlist(i) and store Vlist
  }
  Calculate TFT, IDFT {
  Append the terms to TFlist
  }
  tf - idf is tf - idf(t, d) = tf(t, d) × idf(t, d)
  idf(t) = log2 |D| / { d:t □ d }
  Preparation of feature vector an append FVlist
  For(i=0;i<v)// where V is the stemming word list
  {
    Perform feature vector creation (FVlist)
  }

```

The TF-IDF algorithm has eight phases. They are:

- 1) Dataset is loaded.
- 2) Followed by tokenizing. In this step every word in the document is spilt into tokens, and then stored it as Tlist.
- 3) Prepare the stop words list and store it as SWList.
- 4) Apply the for loop condition to check the stop words.
- 5) If the stored word is present in the document, then delete the word from the document.
- 6) After performing the stop word elimination, do stemming, in stemming process apply null stemmer algorithm to convert the words into their root format. Then store it as Rlist.
- 7) Calculate both TFT, IDFT using $tf - idf$ is $tf - idf(t, d) = tf(t, d) \times idf(t, d)$, $idf(t) = \log_2 |D| / \{ d:t \square d \}$.
- 8) Based on the term frequency, form the feature vector of the document. Store it as Fvlist.

The computation of TF-IDF are as follows:

Term Frequency (TF)= The number of times a word or a term appears in a comment/ the total number of words in the comment.

Inverse Document Frequency (IDF) = $\log(N/n)$

TF-IDF=TF*IDF

Nis the total number of comments.nis the number of comments a word has appeared.

2) *Logistic regression*: The Logistic regression (LR) algorithm is used for supervised learning, and widely used for binary classification tasks. It is a branch of natural language processing (NLP), which is generally thought of as part of artificial intelligence (AI). LR permits obtaining insights about the model, such as observed coefficients.

The LR process is given as follows:

1. Let $p(x)$ be a linear function of x . Every increment of a component of x would add or subtract so much to the probability. The conceptual problem is that p must be between 0 and 1, and linear functions are unbounded. Moreover, many situations empirically see “diminishing returns” — changing p by the same amount requires a bigger change in x when p is already large (or small) than when p is close to $1/2$.
2. Let $\log p(x)$ be a linear function of x , so that changing an input variable multiplies the probability by a fixed amount.
3. Finally, the modification of $\log p(x)$ which has an unbounded range is the logistic (or logit) transformation, $\log(p / (1-p))$.

The algorithm for the logistic regression is given as follows:

```

simulate.from.logr = function(x, coefs) {
require(faraway) # For accessible logit and inverse-logit functions
n = nrow(x)
linear.part = coefs[1] + x %*% coefs[-1]
probs = ilogit(linear.part) # Inverse logit
y = rbinom(n,size=1,prob=probs)
return(y)
}
delta.deviance.sim = function (x,logistic.model) {
y.new = simulate.from.logr(x,logistic.model$coefficients)
GLM.dev = glm(y.new ~ x[,1] + x[,2], family="binomial")$deviance
GAM.dev = gam(y.new ~ lo(x[,1]) + lo(x[,2]),
family="binomial")$deviance
return(GLM.dev - GAM.dev)
}
    
```

3) *Confusion matrix*: A confusion matrix is a technique for summarizing the performance of the classification algorithm. Computing a confusion matrix can give a better idea of what the classification model is getting right and what types of errors it is making. The steps for computing a confusion matrix are as follows:

1. Get a test dataset or a validation dataset with expected outcome values.
2. Predict for each row in the test dataset.
3. From the expected outcomes and predictions, count the number of correct predictions for each class.

IV. RESULTS

This section gives a description of the dataset and their respective categories. It also displays the various comments posted on the social media platform. The requirements for the model automatically detecting categories of toxicity a comment belong are analyzed. These requirements are very essential to implementing the application.

A. Implementation

This algorithms in this study were implemented using Jupyter notebook on ANACONDA IDE written in Python programming language on a personal computer with the configuration of Windows 10 Operating system and i5 processor. The results from implementing the datasets obtained in this study are discussed in this section. Figure 4.1 shows the distribution of the categories: toxic, severe toxic, obscene, threat, insult and identity-hate. From the distribution, the major labels are: toxic, severe toxic and obscene. Fig. 4.2 is used to check for missing values, and count total number of data provided.

```

In [6]: train_df.sample(10)
Out[6]:
   id          comment_text  toxic  severe_toxic  obscene  threat  insult  identity_hate
156859  644262594217c1d  white wrong with the image of 1914. Also you h...  0      0      0      0      0      0
134671  6044e56a308a30b  Last note as his daughter I have been wearing...  0      0      0      0      0      0
154468  ae3e558eca655c   Fucking white trash cunt!Could you also be mo...  1      1      1      0      1      1
157716  a258390ca26d      "ninThis is a short list of users whose pages...  0      0      0      0      0      0
140801  f1ec79c23ba21f0    That is completely absurd. You openly admit lo...  0      0      0      0      0      0
55559  94736536c1b490f    "nin Request n!nH. I would to make use of ...  0      0      0      0      0      0
48858  8229e6c15a2801e    Education n!n! recommend you to get educated...  0      0      0      0      0      0
15519  20456977aba207    Your AIC. Submission n!n!our article submissio...  0      0      0      0      0      0
90164  1159991b05e1c1c    "Ok I added the change suggested. I guess a...  0      0      0      0      0      0
126229  a4b09211ceee6b    I dispute the first 2 instances as personal at...  0      0      0      0      0      0
    
```

Fig. 4.1 Random Data sample

```

In [8]: # check missing values in numeric columns
train_df.describe()
Out[8]:
   toxic  severe_toxic  obscene  threat  insult  identity_hate
count  159571.000000  159571.000000  159571.000000  159571.000000  159571.000000  159571.000000
mean    0.090226      0.019021      0.052962      0.003071      0.049495      0.008066
std     0.294623      0.099401      0.224018      0.055329      0.219990      0.093048
min     0.000000      0.000000      0.000000      0.000000      0.000000      0.000000
25%     0.000000      0.000000      0.000000      0.000000      0.000000      0.000000
50%     0.000000      0.000000      0.000000      0.000000      0.000000      0.000000
75%     0.000000      0.000000      0.000000      0.000000      0.000000      0.000000
max     1.000000      1.000000      1.000000      1.000000      1.000000      1.000000
    
```

Fig. 4.2 Dataset Statistics

The output to check if there is any comment which doesn't belong to the categories (i.e obscene, insult, threat, identity-hate, toxic, and severe-toxic) is given in fig. 4.3. This makes output to be zero(0) i.e. there is no output that does not belong to any category, it is either toxic, severe-toxic, obscene, threat, insult, or identity-hate. If it does fall under any of the categories, it will be 0 all through the category which make it non-toxic comment.

```

In [9]: # check for any 'null' comment
no_comment = train_df[train_df['comment_text'].isnull()]
len(no_comment)
Out[9]: 0
    
```

Fig. 4.3 Number of null comment

The total rows of dataset showing number of rimes a word appears from implementing the algorithm is given in Fig.4.4. In the figure, the number of comments for each combination drops exponentially.

```

In [12]: # Let's see the total rows in train, test data and the numbers for the various categories
print('Total rows in test is {}'.format(len(test_df)))
print('Total rows in train is {}'.format(len(train_df)))
print(train_df[cols_target].sum())
Total rows in test is 153164
Total rows in train is 159571
obscene      8456
insult       7898
toxic        15323
severe_toxic 1599
identity_hate 1418
threat        490
dtype: int64
    
```

Fig.4.4 Total Row of Dataset

The histogram, given in Fig. 4.5, shows the graphical representation of the dataset. Most of the text lengths are within 500 characters, with some up to 5,000 characters long.

```

In [13]: # Let's look at the character length for the rows in the training data and record these
train_df['char_length'] = train_df['comment_text'].apply(lambda x: len(str(x)))
In [14]: # Look at the histogram plot for text length
sns.set()
train_df['char_length'].hist()
plt.show()
    
```

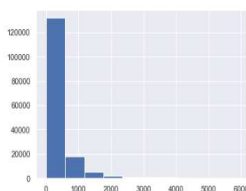


Fig. 4.5 Histogram of Dataset

The correlation matrix of the categories is given in Fig. 4.10. The correlation matrix shows 'toxic' is correlated with 'obscene' and 'insult' (0.68 and 0.65). 'toxic' and 'severe_toxic' have a 0.31 correlation factor. This implies 'toxic' and 'severe_toxic' are not correlated. 'insult' and 'obscene' have a correlation factor of 0.74. This implies 'insult' and 'obscene' are highly correlated.

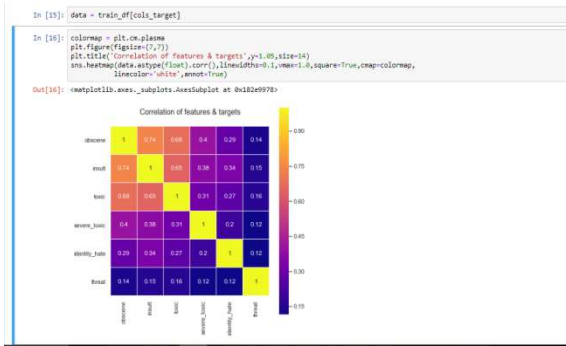


Fig. 4.10 Correlation Matrix of Categories

B. Data Training

Training of the processed dataset is done by solving the problem as a multi-label classification problem. The multi-label classification problem is transformed into a separate single-class classifier problem. This is known as problem transformation. This is achieved using the binary relevance method. The pre-processed dataset is trained using logistic regression, and the training accuracy is given by Fig. 4.11 and summarized in Table 4.1.

```
In [24]: #lets define an array for all the categories
cols_target=['obscene', 'insult', 'toxic', 'severe_toxic', 'identity_hate', 'threat']

In [25]: # import and instantiate the Logistic Regression model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
logreg = LogisticRegression(C=12.0)

# create submission file
#submission_binary = pd.read_csv('submission.csv')

for label in cols_target:
    print('... Processing {}'.format(label))
    y = train_df[label]
    # train the model using X_dtm & y
    logreg.fit(X_dtm, y)
    # compute the training accuracy
    y_pred_X = logreg.predict(X_dtm)
    print('Training accuracy is {}'.format(accuracy_score(y, y_pred_X)))
    # compute the predicted probabilities for X_test_dtm
    test_y_prob = logreg.predict_proba(test_X_dtm)[:,-1]
    #submission_binary[label] = test_y_prob

... Processing obscene
C:\Users\Sophy\Anaconda2\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
to 'lbfgs' in 0.22. Specify a solver to silence this warning.
FutureWarning)

Training accuracy is 0.983173634307
... Processing insult
Training accuracy is 0.975296263105
... Processing toxic
Training accuracy is 0.96377879439
... Processing severe_toxic
Training accuracy is 0.992078761178
... Processing identity_hate
Training accuracy is 0.993045999586
... Processing threat
Training accuracy is 0.998057291112
```

Fig. 4.11 Training Accuracy

Table 4.1 Summary of Training Accuracy and Precision

Label	Training Accuracy	Precision
Toxic	96.38%	0.91
Severe-toxic	99.21%	0.99
Obscene	98.32%	0.95
Threat	99.81%	1
Insults	97.53%	0.95
Identity-hate	99.38%	0.99

The training accuracy for the various comments obtained from social media network is very high. The training accuracy for toxic comments is 96.38%. severe toxic comments (99.21%), obscene comments (98.32%), threats (99.81%), insults (97.53%), and identity-hate (99.38%).

IV. CONCLUSION

Communication is one of the basic necessities of everyone's life. People need to talk and interact with one another to express what they think. Over the years, social media and social networking have been increasing exponentially due to an upsurge (rise) in the use of the internet. Flood of information arises from online conversation on a daily basis, as people are able to discuss, express themselves and express their opinions via these platforms. While this situation is highly productive and could contribute significantly to the quality of human life, it could also be destructive and enormously dangerous. The responsibility lies on the social media administration, or the host organization to control and monitor these comments.

This research work focuses on developing a model that would automatically classify a comment as either toxic or non-toxic using logistic regression. Therefore, this study aim to develop a multi-headed model to detect different types of toxicity like threats, obscenity, insults, and identity-based hate. By collecting and preprocessing toxicity classified comments for training and testing using term frequency-inverse document frequency(TF-IDF) algorithm, developing a multi-headed model will detect different types of toxicity using logistic regression to train the dataset, and evaluate the model using confusion metrics.

REFERENCES

- Deng, A., Yu. D., (2014). Deep Learning. Methods and Applications. Retrieved from <http://research.microsoft.com/pubs/209355/DeepLearning-NowPublishing-Vol7-SIG-39.pdf>
- Yoshua, B., (2009). Learning Deep Architectures for AI (PDF). Foundations and Trends in Machine Learning
- Nobata, C., Tetreault, J., Thomas, A. Mehdad, Y., Chang, Y., (2016). Abusive Language Detection in Online User Content. International Conference on World Wide eb, pp. 145–153.
- Aggarwal, C., Zhai, C., (2012). A Survey of Text Classification Algorithms. In Mining Text Data. Springer, pp. 163–222.
- Haddadi, C., Benevenuto, H., Gummadi, K., (2010). Measuring user Influence in Twitter: The million worldwide fallacy. 4th International AAAI Conference on Weblogs and Social Media (ICWSM), vol. 14, no. 1, p. 8.

- [6]. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T., (2017). Enriching Word Vectors with Subword Information, *TACL*, vol 5, pp.135–146.
- [7]. Qian, M., Sherief, E., Belding-Royer, E., Wang, W., (2018). Leveraging Intra-User and Inter-User Representation Learning for Automated Hate Speech Detection, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 2.
- [8]. Tarasova, Z., Khlinovskaya-Rockhill, E. Tuprina, O., Skryabin, V., (2017). Urbanization and the Shifting of Boundaries, *Contemporary Transformations in Kinship and Child Circulation among the Sakha*. *Europe-Asia Studies* vol. 67, no.7, pp. 1106-1125.
- [9]. Adamic, L., (2016). The small world web, *Research and Advanced Technology for Digital Libraries*, pp. 852–852.
- [10]. Hanson, R., (2014). Foul play in information markets. *George Mason University*, vol. 18, no. 2, pp. 107-126.
- [11]. Waseem, Z., Thorne, J., Bingel, J., (2018). Bridging the gaps: Multi-task Learning for Domain Transfer of Hate Speech Detection. *Online Harassment*, pp 29–55.
- [12]. Kumar, R., Ojha, A., Malmasi, S., Zampieri, M., (2018). Benchmarking Aggression Identification in Social Media, *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pp. 1-11.
- [13]. Georgakopoulos, V., Tasoulis, S., Vrahatis, A., Plagianakos, P., (2018). Convolutional Neural Networks for Toxic Comment Classification, *ACM Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, pp. 35.