# Multiclass text classification of dark web content

Dennis van Gilst, 16056302@student.hhs.nl
Dennis van Oosten, 16090187@student.hhs.nl
Wessel Ottevanger, 16014324@student.hhs.nl
Christian Ruigrok, 16034198@student.hhs.nl
Mike Verheijden, 19104367@student.hhs.nl

*In association with The Hague University of Applied Sciences (THUAS) and Netherlands Organization for Applied Scientific Research (TNO)*

**20 January 2020**
Johanna Westerdijkplein 75, The Hague

**Abstract.** This paper describes how a pipeline can be created to classify dark web text content into multiple predetermined topics. The different elements of this pipeline are explained and the steps that need to be taken to train and use a model to classify text content are described. An experiment is set up that combines different machine learning algorithms and shows what pipeline configuration results in the best outcome, also taking neural networks and ensemble learning into consideration. Results show that using a minimal preprocessing method combined with tf-idf vectorization and a trained Linear SVC model is the most effective configuration with a corresponding F1 score of 93 percent.

**Keywords:** machine learning, natural language processing, text classification, dark web

# 1 INTRODUCTION

The internet consists of two different sections, the clear- and the deep web. The deep web is the part of the internet which is not accessible through standard web browsers because it is not indexed and therefore not shown in the output of any search engine query. Within the deep web there is another part of the internet, the dark web, which has become the home for many (cyber)criminals to sell goods or services because it is highly anonymized. [12]

International law enforcement is very interested in this part of the internet where criminals can seemingly do their business with a low risk of getting caught. Therefore the European Union has initiated a project called TITANIUM (Tools for the Investigation of Transactions in Underground Markets) partnering with multiple organizations which aims to get insight and knowledge about the dark web, targeted to help law enforcement track down criminals. One of the goals of the project is to be able to automatically categorize the text on dark web forums and marketplaces to a predetermined list provided by Interpol. To make this possible, a trained model needs to be able to predict the topic of a given text.

The training of such a model is defined as *'multi-class text classification'* since the list of topics provided by Interpol contains multiple topics. [14] To be able to realize this dark web classifier, a pipeline needs to be created that can be fed with a text of varying length and will accurately predict the topic. This paper describes how such a pipeline can be created, looking at different preprocessing, vectorization and model training methods.

The main research question is: *"How can a pipeline be created that classifies dark web text-based content to a predetermined topics list?"*

# 2 METHODOLOGY

## 2.1 Data retrieval

The data on which the model will be trained and validated comes from two different sources. One of which is a dark web dataset of the Agora marketplace, publicly available on Kaggle. From now on, this dataset will be referred to as the *'Agora' dataset*. The second dataset comes from Web-IQ, which focuses on multiple marketplaces and is for internal use at TNO only. [2] [15] This dataset will be referred to as the *'Web-IQ' dataset*.

Both datasets are created by scraping sites from the dark web, meaning that the data is read from the source once and stored for use during this project. This means the data is unaffected by any changes made on the dark web pages during the project.

## 2.2 Interpol topics

The final text classification pipeline has to categorize the topics to a predefined list of topics Interpol is interested in. [14] Therefore, the categories of both datasets were manually mapped to the Interpol category list and preprocessed using an automated script. By doing this, both datasets will have the same categories which allows a more accurate comparison of the performance on the two datasets.

## 2.3 Data visualization

The Agora dataset consists of 109,563 records, spread out over 104 different categories. The dataset has a tree structure that divides main categories into subcategories (e.g. 'Drugs/Cannabis/Hash'). It consists of 14 main categories (see figure 1). As can be seen in the figure, the categories are not well balanced, as 'Drugs' is by far the most represented category.

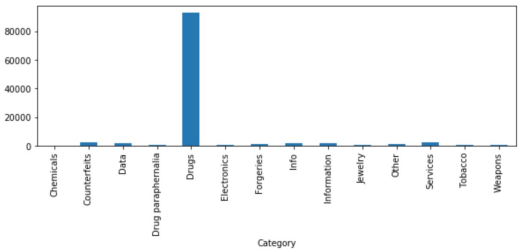



**Fig. 1.** Histogram of the main categories within the Agora set, divided over the full dataset

The Web-IQ dataset consists of 11,589 records, spread out over twelve categories (see figure 2). All these categories can be considered main categories, so they are not divided into subcategories and can be compared to the main categories of the Agora dataset.

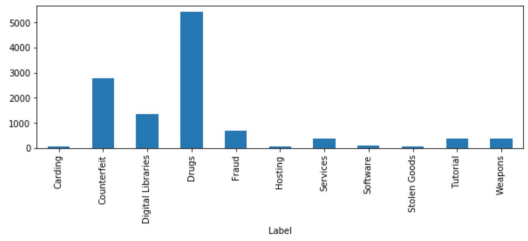



**Fig. 2.** Histogram of the categories within the Web-IQ dataset, divided over the full dataset.

While the Agora dataset has more records than the Web-IQ dataset, the length of the records is capped at 200 characters (see figure 3).

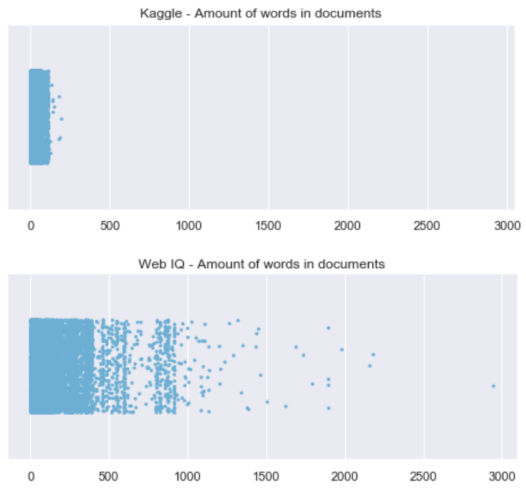



**Fig. 3.** Comparison of the word counts of both datasets

## 2.4 Pipeline

To create a pipeline for Natural Language Processing (NLP) there are a few basic steps that need to be taken. [3] [8] Figure 4 shows a model of the pipeline. The first step is to preprocess the text, basically cleaning the text and make it more uniform for an algorithm to understand. For most model training algorithms, the model needs to be fed with numbers and can not simply work by feeding the raw text data. To do so, vectorization techniques need to be applied to convert the words and descriptions into vectors. The last step is to train a model on a train set and to test its score against a test set.

To use the trained model, new/unseen data is preprocessed and vectorized using the same method used for training the model. Then the vectorized data is being fed into the trained model, which predicts the topic of the text.

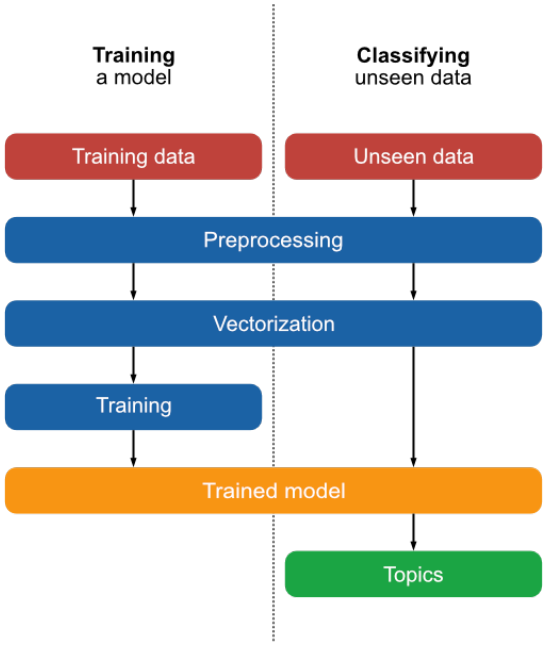



**Fig. 4.** Model of the pipeline

### 2.5 Balancing

Because both of the datasets were heavily biased towards drugs, for some experiments the datasets were balanced so the number of records per category would be equal to prevent the trained models to be biased towards drugs as well. Some experiments were conducted using the same number of records as there were records with the least occurring category and some experiments were conducted copying records so every category had the same number of records as the most occurring category.

### 2.6 Preprocessing

The goal of preprocessing text data is to transform the raw text data to make it more uniform. The preprocessing of text for classification can contribute to a higher accuracy score for the model. [10] There are different strategies for preprocessing, like cleaning the text severely or keeping it mostly in its original state. A preprocessing script was created that allows for easily combining multiple methods.

Combinations (see figure 5) were made with the following methods [4] [9]:

- Lowercase: converts all characters to lowercase.
- Punctuation: removes all punctuation from the text.
- Numbers: removes all numbers from the text.
- Non-ASCII: removes all non-ASCII characters from the text.
- Cut off: because the Agora descriptions have a maximum character length, the last word would be cut off / not complete. This method removes the last word of the text.
- Stop words: removes stop words.
- Lemmatizing: applies lemmatization to all words.
- Stemming: applies stemming to all words.

Lemmatization and stemming work in different ways, the former uses a vocabulary to return the base or dictionary form of a word and the latter simply removes the ending of words to create a stemmed version. Removing stop words removes a bit of information that might or might not be useful for an algorithm to interpret.

The combinations are made based on the fact that the specific differences mentioned above could affect the outcome of the results significantly. Minimal stripping is included because using raw text data might give surprising results.

The reason not every possible combination was used is because this is time consuming and would require more time than was available.

### 2.7 Vectorization

Because a model needs numbers to learn, the text needs to be converted to numbers (also called vectorization or feature extraction). After preprocessing, this is the second step in the pipeline for text classification. This translation from text into numbers is called 'vectorization' or 'feature extraction'. The following vectorization methods were used:

- Bag of Words
- Term frequency - inverse document frequency (tf-idf)
- Word2Vec
- Doc2Vec
- FastText

These methods are all suited for use in NLP and have varying degrees of complexity, which should give a nice range of test options.

### 2.8 Model Training

To train a model, several different algorithms were chosen:

- Decision Tree
- Gaussian Naive Bayes
- K-Nearest Neighbours
- Linear Discriminant
- Linear Support Vector Machine Classifier (Linear SVC)
- Logistic Regression
- Random Forest Classifier
- Linear Classifier trained with Stochastic Gradient Descent (SGD)

| Name | lowercase | punctuation | numbers | non-ASCII | cut off | stop words | lemmatizing | stemming |
|---|---|---|---|---|---|---|---|---|
| Extreme Stripping | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Default stripping (lem) | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| Default stripping (stem) | ✓ | ✓ | ✓ | ✓ | ✓ | | | ✓ |
| Default stripping (none) | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| Minimal stripping | | | | | | | | |

**Fig. 5.** Preprocessing techniques

Hyperparameters will be tweaked for each model trying to improve the scores. [11]

For this project only supervised machine- and deep learning models are used because the ground truth for the data was available and manually labeling clusters would be too time-consuming.

The selection of models is made based on the fact that together they cover a wide range of models from decision trees to regression based approaches.

### 2.9 Neural network

For a broad comparison between models, a deep neural network with Keras and a multi-layer perceptron are selected from Sklearn since this is easy to implement. [7] [11]

### 2.10 Ensemble learning

Another method that is tested is ensemble learning. This means multiple models will be combined into one model to see if they can predict better results when working together. Ensemble learning can be used to reduce the effect of overfitting because having multiple classifiers can compensate for the negative effects. There are multiple methods for implementing ensemble learning. For this project, a variation on the "Bucket of models" approach is used. [1] A bucket of models means to train multiple different models and use the model that is best in classifying the category in question.

### 2.11 Experiment setup

To see which combination of preprocessing, vectorization and model training gives the best result, an experiment is set up that combines all different models and tests them against the different preprocessing and vectorization methods. The outcome of the experiment will be a matrix with scores for all combinations. This experiment is executed twice; once with the Agora data and once with the Web-IQ data.

The scores of the neural network and ensemble learning can be compared to these matrices.

## 3 RESULTS

### 3.1 Agora pipeline comparison

The result of the experiment as described in 2.11 for the Agora dataset can be seen in the comparison matrix in appendix A.

The highest F1 score in the matrix is 93 percent which is achieved by training a Linear SVC model with tf-idf vectorization. For this particular dataset, it seems not all preprocessing methods made a significant difference. The same score occurs with the following three preprocessing methods:

- default stripping
- default stripping with stemming
- minimal stripping

The results of the Linear SVC model trained on the Agora dataset did not improve when using different hyperparameters than the default.

### 3.2 Web-IQ pipeline comparison

The result of the experiment as described in section 2.11 for the Web-IQ dataset can be seen in comparison matrix in appendix B.

The highest F1 score in the matrix is 87 percent which is achieved by training a Linear SVC model with tf-idf vectorization, a random forest classifier with FastText vectorization and

a linear model trained with stochastic gradient descent with tf-idf vectorization. Just like with the Agora dataset, a few different methods scored the same result.

The results of the Linear SVC model trained on the Web-IQ dataset did not improve when using different hyperparameters than the default.

### 3.3 Neural network

With a deep neural network using Keras trained on the Doc2Vec vectors, an accuracy of 76 percent was achieved by tweaking the hyperparameters.

The multi-layer perceptron did not improve on the highest score from the Linear SVC model. An accuracy of 90 percent on the balanced Agora dataset was the highest score the neural network achieved.

### 3.4 Ensemble learning

The results of the ensemble learning scripts did not improve on the results of the best machine learning model. The highest achieved F1 score with ensemble learning was just about 0.5 to 1 percent under the highest achieved result with the Linear SVC model, trained on the Agora dataset.

## 4 CONCLUSION

In conclusion, although the project consisted of researching and experimenting with various machine- and deep learning models the main research question "How can a pipeline be created that classifies dark web text-based content to a predetermined topics list?" was best answered when using one of the simpler models. As mentioned in Occam's razor, *the simplest solution is often the correct one.* [13]

The research found that the best pipeline for classifying text-based content consisted of a preprocessing component that minimally alters the text. The vectorization component used tf-idf and applies n-grams to create shingles of two words to create feature vectors of the used corpus. Finally, a Linear SVC model was trained that can be used to classify new data.

## 5 DISCUSSION

Due to our limited time, we only used supervised ways of training models in our experiments. It might be interesting for future work to explore unsupervised options to see whether clusters can be found in the data which can be linked to topics using either *'Latent Dirichlet Allocation'* or custom word net mappings. [5] This way, the model might automatically pick up on new topics when new things are being sold or discussed on the dark web and combining different topics within existing datasets (like the Darknet Market Archives [6]) will be made easier.

We have used two datasets which differ in structure. The Agora dataset has many rows (approx. 110.000+) but each row does not contain much text, often less than 200 characters. This makes this dataset tall and slender. The Web-IQ dataset has fewer rows (approx. 11.000+) but more text per row, making it short and fat. We found that training a model on one dataset and validating on the other delivered poor results. Interesting insights may be provided when combining the two datasets into one and training a model on this mixed data. We suspect it might improve the scores when validating on less rich data such as the Web-IQ dataset.

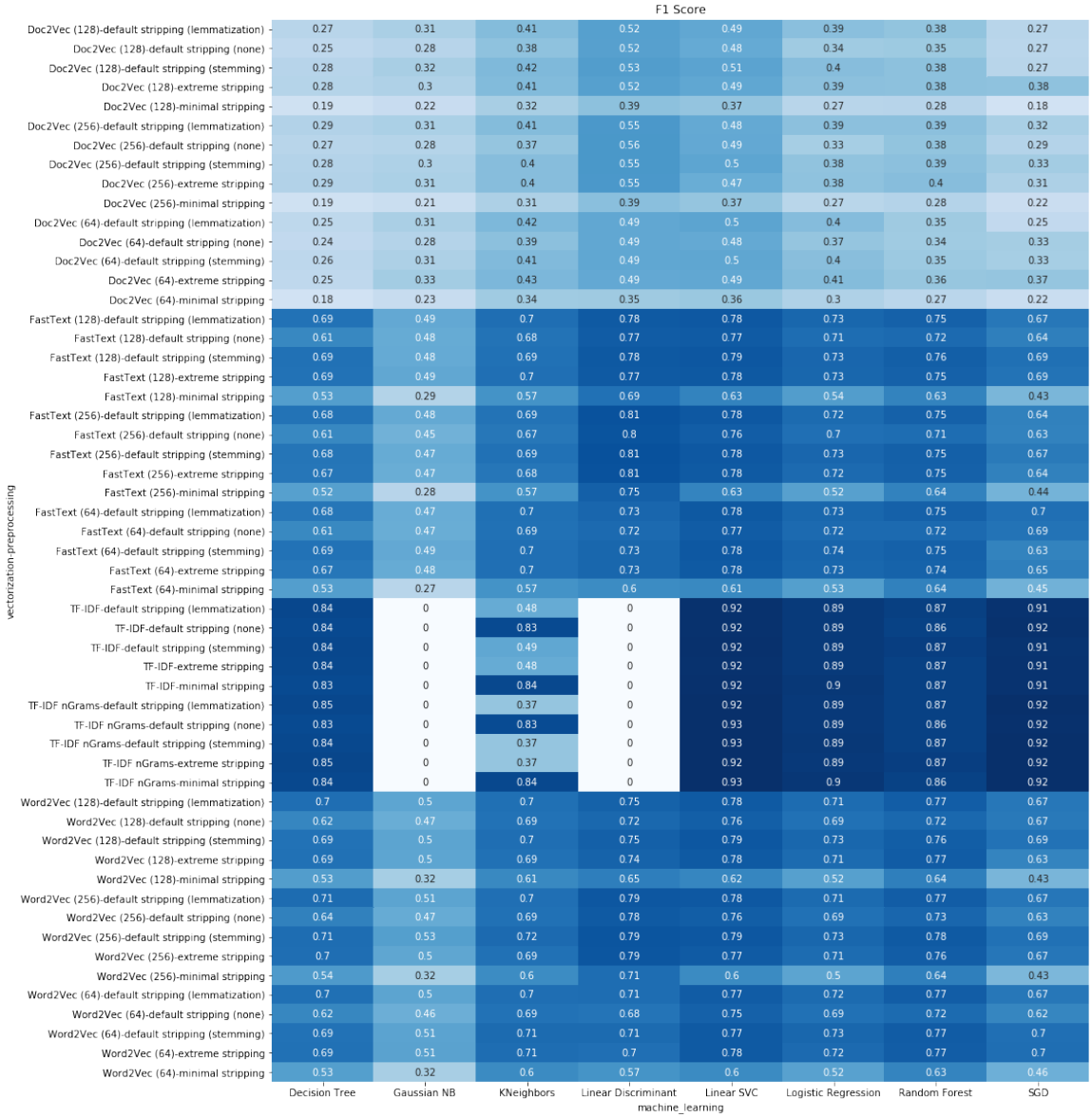
## References


1. Charu C Aggarwal. Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter*, 14(2):49–58, 2013.
2. Philip Bal. Dark Net Marketplace Data (agora 2014-2015). `https://www.kaggle.com/philipjames11/dark-net-marketplace-drug-data-agora-20142015/`. Accessed: 05-09-2019.
3. Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.
4. Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain, July 2004. Association for Computational Linguistics.
5. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
6. Gwern Branwen, Nicolas Christin, David Décary-Hétu, Rasmus Munksgaard Andersen, StExo, El Presidente, Anonymous, Daryl Lau, Delyan Kratunov Sohhlz, Vince Cakic, Van Buskirk, Whom, Michael McKenna, and Sigi Goode. Dark net market archives, 2011-2015. `https://www.gwern.net/DNM-archives`, July 2015. Accessed: 14-01-2020.
7. François Chollet et al. Keras. `https://keras.io`, 2015.
8. Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.
9. Djoerd Hiemstra and Franciska MG de Jong. Statistical language models and information retrieval: natural language processing really meets retrieval. *Glot international*, 5(8):288–293, 2001.
10. Sotiris Kotsiantis, Dimitris Kanellopoulos, and P. Pintelas. Data preprocessing for supervised learning. *International Journal of Computer Science*, 1:111–117, 01 2006.
11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
12. Cath Senker. *Cybercrime & the Dark Net*. Siriusware, 2017.
13. W. M. Thorburn. Occam's razor. *Mind*, 24(2):287–288, 1915.
14. TNO. Interpol topics list. Acquired via mail contact, September 2019. For internal use only.
15. WebIQ. Darknet markets dataset. Acquired via mail contact, December 2019. For internal use only.

# Appendices

# Appendix A

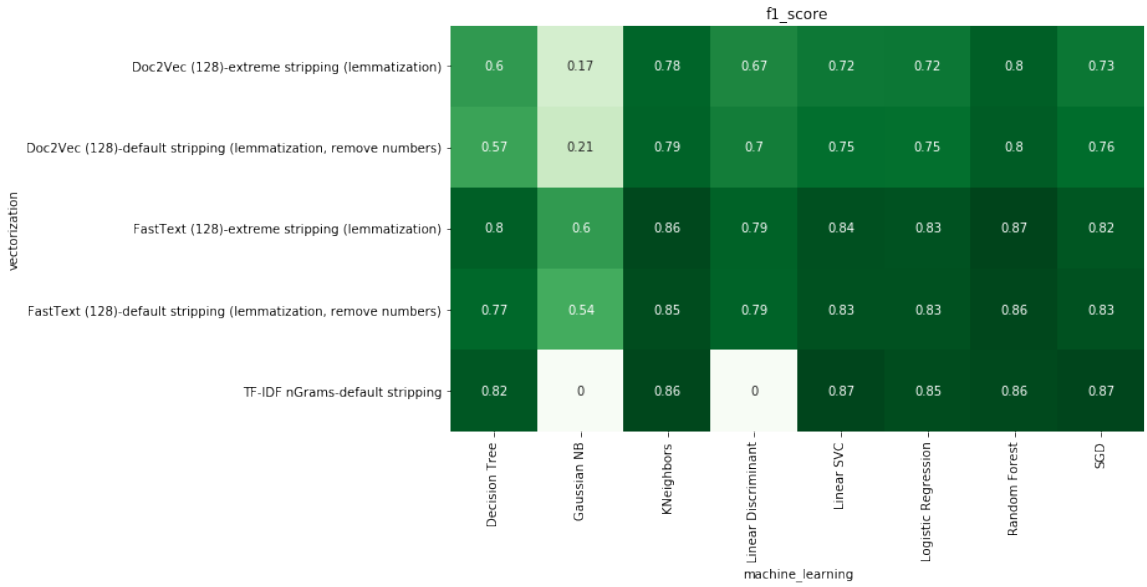
# Agora Pipeline Comparison Matrix




Matrix showing the F1 score of all preprocessing, vectorization and training combinations used on the balanced Agora dataset (500 rows per category).

# Appendix B

# Web-IQ Pipeline Comparison Matrix




Matrix showing the F1 score of all preprocessing, vectorization and training combinations used on the Web-IQ dataset.