University of Amsterdam

Forum post classification to support forensic investigations of illegal trade on the Dark Web

Research Project 2

Diana Rusu
diana.rusu@os3.nl

Supervisors:

Martijn Spitters
martijn.spitters@tno.nl

Stefan Verbruggen
stefan.verbruggen@tno.nl

TNO, Media & Network Services

September 13, 2015

Abstract

Cybercrime is facilitated by the multitude of Internet services. It is a flourishing field which affects governments, businesses and citizens that are using the Internet as a service. Cybercriminals take advantage of Internet features, such as anonymity and encryption, to commit illegal acts. Consequently, it is important for law enforcement authorities to have at their disposal intelligent tools that would support them in investigations.

This research aims to provide forensic analysts a quick way to determine the topics currently being discussed in the underground marketplace forums. To achieve this, we investigated if it is feasible to exploit semantic word representations in order to build a classifier.

With time being a limited resource in many forensic cases, we aimed to boost the training process of the classifier, as it involves labour intensive annotation work. For this purpose we made use of Word2Vec a state of the art technique. We proceeded on evaluating the classifier and the training model by determining their accuracy. For these tasks, methods such as Cosine Similarity and Support Vector Machines were employed. The results are discussed and solutions are proposed in order to improve the process.

Acknowledgements

I would like to express my gratitude and appreciation to my supervisors, Martijn Spitters and Stefan Verbruggen, for the valuable feedback, guidance and patience during this research. All the input, advices and critics led to a constructive path towards the project completion.

I want to also thank TNO for providing all of the necessary resources for this project within short notice. It was a great opportunity to work closely with the Dark Web team and get familiar with their research efforts.

For suggestions and feedback on the report, I wish to express my sincere thanks to Dragos Laurentiu Barosan and Stefan Boronea.

Contents

1 Introduction

This chapter will introduce the main reasons for investigating and analysing marketplace forums data that resides under the DarkWeb. It also contains the main research questions this research will try to answer and the previous work performed regarding the topic of this project. Considering that Darknets are intended to be anonymous platforms, ethical issues will be examined in the last subsection of this chapter.

1.1 Motivation & Scope

The Internet is a daily used system which features services such as instant communication, business markets and social networking. These capabilities are of interest not only for law abiding citizens, but also for criminals[1]. One of the key targets of the Internet users that run illegal businesses is to protect their anonymity.

In order to communicate freely with their customers, without the risk of being traced, they make use of tools such as TOR[2] or I2P[3], known collectively as Darknets[4]. These tools have been misused by people involved activities such as illicit drug and weapon trade, money laundering, hacking services, child pornography, and even assassination services to thrive on the Internet[5]. A lot of these trading activities and services take place on underground forums and marketplaces, which reside under the DarkWeb. Therefore, it is an urgent matter for law enforcement authorities to have at their disposal intelligent tools that would automate data analysis gathered from the Dark Web, data required for cybercrime investigations.

As time is limited in many forensic investigations, getting a swift thematic insight of what is being discussed under the marketplace forums would provide support to forensic analysts when dealing with large collections of discussion data about illegal trade. An assessment of the aimed categories of data could be performed for further investigations.

Keyword matching, used in regular expressions, would be insufficient for several reasons. First of all, it would be time consuming to define all possible keywords. Lets take as an example one of the classes useful in investigations,

namely *hard drugs*. Names for each type of drug might change or a new type of drug could be introduced on the market. The context in which this type of drugs is mentioned and discussed, however, might be similar. Therefore, by using a method that will take into account the context for the documents (posts) we could enhance the discovery of possible drug types and the new ones which will be introduced. Another reason why keyword matching is not sufficient is the fact that some of the posts under DarkWeb might contain codified language (e.g. the ISIS community form of communication under the DarkWeb [6]). However, even if the discussion style is changed, again the context in which the drugs or weapons are discussed will remain similarly close. A particular problem is that of words with multiple meanings. A drug may have a name which out of context would be considered irrelevant. This could result in either ignoring relevant information or in gathering a lot of irrelevant data.

The aim of this research is to investigate how semantic word representations can be exploited for classifying sparse, short forum posts on marketplace discussion forums, using a (small) labeled training data. By *sparse* forum posts we refer to data that is less topic focused and not so consistent compared to traditional documents. It becomes difficult to discover the topic of interest when this forum posts are very short and they do not provide word co-occurrence[9].

We have experimented and tested techniques, such as Cosine Similarity[23] and Support Vector Machines[32], by making use of the word2vec[7] model, as it has shown promising results for modeling the underlying semantics and syntax of large collections of unstructured text [8]. We performed the k-fold cross validation technique to evaluate the constructed training model. As well, we evaluate the classification task by calculating the accuracy of the outcome results for Cosine Similarity. The dataset used for examination and assessment comprises of aggregated posts from multiple discussion forums associated with Tor marketplaces (e.g. Agora, Evolution, Silk Road, BMR, etc).

1.2 Research Questions

The main question that this research will try to answer, in the context of grouping DarkWeb marketplaces forum posts into relevant categories useful for forensic investigators, is:

Can semantic word representations be used to boost the set-up and training process of forum post classifiers?

For the above question we have posed the following subquestions:

- *What methods can be used to exploit the word representations for classifying sparse, short forum posts on discussion forums using few training examples?*
- *What is the accuracy of the proposed methods and how can they be improved?*

1.3 Related Work

Classifying data, such as forum posts, has been intensively investigated during the past few years. Therefore, there is a lot of literature around this field, especially in the sentiment analysis area. Some of the previous publications have performed similar work, on different datasets, but having a similar approach.

One of the most important works related to this research, is the paper of Xuan-Hieu Phan et. al [9] which is using LDA(Latent Dirichlet allocation) for topic modelling, that also produces a contextual model of the data, inferring the topic distribution for the test documents from the model.This topic distribution is then used as a feature set for the classifier in a similar manner to that which this research does. In addition to the previous mentioned research, we will make use of Word2Vec model to produce a contextual model.

Another work related to this research is the report published by Duyu Tang, Furu We et. al [10], where they present a method which learns word embedding for sentiment analysis on twitter posts. They make a comparison between

the Word2Vec model and their defined sentiment specific word embedding (SSWE) method which encodes sentiment information in the continuous representation of words. In our research we will make use of Word2Vec, as our goal is to extend the training sets for the classifiers. Also the dataset on which the testing will be performed is crawled from DarkWeb marketplace forums, not from Twitter.

Furthermore, the research conducted by Yoon Kim [11] performs several experiments with convolutional neural networks built on top of the Word2Vec model, with small tunings on hyperparameters, to improve the performance on sentence classification. In his research, Yoon Kim makes use of publicly available word2vec vectors, which were trained on over 100 billion words from Google. There is an available repository, containing publicly available code on Github[1] which can be tested. It it an important work and it could be inspected closer for improving our later defined classification task.

One important research on classifying Internet forum user posts[12] has been performed by Sumit Bhatia et. al. In the experiment conducted, they made use of the Ubuntu forum discussion posts as their primary dataset and categorized posts in classes such as question, repeat question, clarification, suggest solutions, positive feedback and negative feedback. In this research we had a similar approach.

1.4 Ethical Considerations

In this research we performed several tests by making use of word representations. The dataset, on which the experiments were conducted, was provided by TNO and includes posts aggregated from different DarkWeb marketplace forums. As this data is hidden and protected by encryption or passwords, under the DarkNets, we took into account user privacy. Therefore, we did not use any personal information about the users that posted the data.

[1] https://github.com/yoonkim/CNN_sentence

2 Theoretical background

This chapter will briefly introduce the main concepts and theoretical aspects which concern the techniques chosen, namely Word2Vec [7], Cosine Similarity [23], Support Vector Machines [32] and k-Nearest Neighbour [35]. For a comprehensive approach, the strength and weaknesses of each of the methods aforementioned will be outlined.

2.1 Document Classification

Classification is a method that assigns documents to a corresponding class [19]. There are two main approaches used for categorizing documents: supervised or unsupervised method. The purpose of supervised classification is to build a model using a training data set which consists of documents for which the categories are known a-priori. Once the model is built, it can be used to predict the categories of input documents for which we do not yet know the outcome classes. In contrast, unsupervised approach is used in clustering where there is no model involved, but a *similarity method* to determine clusters of data. In this research we use a combination of supervised and unsupervised techniques: unsupervised to create the word embeddings, which will subsequently be used as the features in a supervised classification task.

In order to classify and make predictions for new data entities, we need to determine a set of features which will differentiate between the intrinsic characteristics of each document. In our case we used the word2vec technique, that for an input corpus, will output a set of vectors. These vectors will have associated a specified number of features for each word vector [15]. We chose word2vec as the main technique for building the classifier as it aims to learn the meanings behind words[7]. It will capture the similarities and relationships, including both semantic and syntactic, incorporated in the features learnt without human supervision. The patterns arise automatically in the training process.

2.2 Word2Vec

Some of the natural language processing tasks use for word representations the score associated to those words across a given document or set of documents.

One known method is TF-IDF[16] or Term Frequency - Inverse Document Frequency, that will provide the importance of the words without taking into account their semantics and syntactic meaning in a given context.

Another method is Bag Of Words[20] which counts the number of words appearing across a document and then it builds vectors according to the frequency of the corresponding words. So, the similarity between some given documents will be determined from the composed vectors. Neither of these methods aforementioned take the semantic meaning of a context into account. Bag of Words building up the necessary information based on the vectors that measure the frequency of the words.

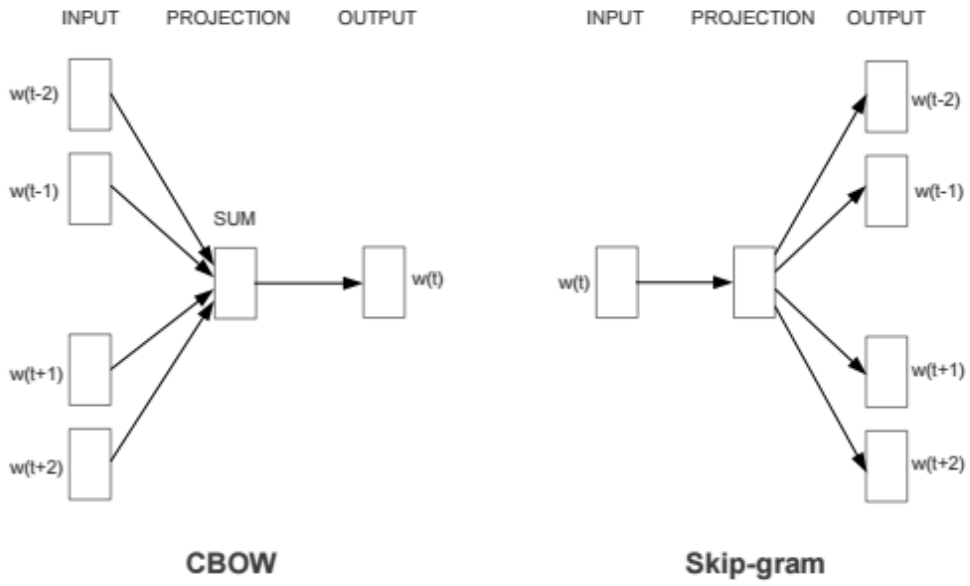

Figure 2.1: CBOW and Skip-Gram algorithms representing the semantic and syntactic meaning. Image Source: Tomas Mikolov et. al *Efficient Estimation of Word Representations in Vector Space*[21]

Word2Vec[7] technique comes with a solution that will take care of the encoded semantic meaningful information. Instead of counting words, it will represent each word as a vector of features corresponding to different characteristics. It achieves this by making use of one of the following architectural methods: CBOW (or Continuous Bag-Of-Words)[7] or Skip-Gram[13]. As training algorithms there is a selection between hierarchical softmax or negative sampling.

Both of these methods train the model in order to build a vocabulary of word representations with a rich number of features or characteristics that take into account semantic information and context. The difference comes from the algorithm they use to accomplish their goal. CBOW uses multiple context words in order to predict a target word, while Skip-gram uses one target word in order to predict the context. Figure 2.1 illustrates these two methods.

These models are considered *shallow* neural models as they have a single hidden layer. Skip-gram performs better in situations where only a small training set is available since multiple training instances can be created from a small amount of data. This is the reason to chose Skip-gram further for building the model needed in the classification task.

A well-known example, used to show analogies when performing arithmetic operations on word representations, is *king–man+woman=queen* [22]. What we are actually asking is: if *man* is related to *woman* then what is *king* related to? The algorithm will output *queen*, which means that it actually knows the difference between genders. It achieves this by learning relationships between words automatically.

2.3 Cosine Similarity

Cosine Similarity is used to measure the similarity between two elements(e.g. documents) in a vector space [23].

If the cosine angle formed by the vectors, when measured, is equal to 0°then those vectors coincide and have a similarity value of 1, meaning they have the same direction. If the two vectors are diametrically opposed then they have a similarity of -1.

Cosine similarity between two vectors, x and y, can be calculated using the following formula[2]:

$$CosineSimilarity(x, y) = \cos(\theta) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

In the above formula the numerator contains the dot product (i.e. inner product) and the denominator will have the vectors norms. If the vectors x and y will be non-negative, similarity will be bounded between [0,1]. For our purposes, the space will be comprised of vectors representing forums posts. These will result from averaging the word representations contained in the posts.

2.4 Support Vector Machines

Support Vector Machines (or short SVM) [32] are a set of supervised classification methods which predict new data by constructing a model out of a training dataset. This model contains the vector representations of the entities from the training data. Prediction for unknown data is based on decision of hyperplanes for which there are defined boundaries [34]. More precisely, SVM uses hyperplanes [33] for classification purpose by splitting the data. For example, given an annotated training input, it will output an optimal hyperplane that will be used in classifying new documents. This is known as *linear classification*. To find an optimal boundary when separating data and the best hyperplane, the largest margin between two classes is chosen. Therefore, the best hyperplane is the one that has the largest separation between the nearest vector pointing to it.

For this project, a multi-class strategy was chosen, in which we made use of the one-vs-rest approach, which means that the classification task will categorize posts into multiple classes. A post will belong to a specific chosen class only. In this case n class models were trained.

[2]http://brenocon.com/blog/2012/03/cosine-similarity-pearson-correlation-and-ols-coefficients/

The advantages of using this method is that is still effective when the number of dimensions is greater than the number of samples[3]. Another asset is the effectiveness in high dimensional space, which is needed in our research.

One of the drawbacks is that it does not provide direct probabilistic estimates. However, we could calculate these using k-fold cross-validation, which is a model validation technique [17]. We used the cross validation technique to estimate how accurately the predictive model will perform.

2.4.1 k-Fold Cross Validation

Cross validation is a technique used to evaluate the predictive performance of a model[17]. In some situation when a model fails to predict unseen data, an overfitting problem arises. Overfitting [18] occurs in the moment the model contains errors and noise rather than relationships.

With k-Fold cross validation approach, the training data will be split into k smaller sets. For the derived *k folds*, we will train a model on *k-1* of the folds as the training data. The rest of the data is used as test data. The resulting model from *k-1* folds, will be validated over the test set. In order to determine the accuracy of the classifier, we run the cross validation several times, each time with a different arrangement of the data. The average values reported by the k-fold cross validation will show the performance of the classifier.

2.4.2 Precision & Recall

Precision and recall are metrics used to determine and evaluate the classifier output quality[24] measuring the results relevance. In order to compare the output classifier results, there are well known defined terms such as true positives (Tp), true negatives (Tn), false positives (Fp) , and false negatives (Fn). The positives and negatives refer to the prediction of the classifier. They refer on what we are expecting from the classifier to output. While the true and false refer to the some external prediction, also known as *observation*. The following formulas are used to calculate the classifier

[3]http://scikit-learn.org/stable/modules/svm.html

quality [4]:

$$Precision = \frac{T_p}{T_p + F_p} \tag{1}$$

$$Recall = \frac{T_p}{T_p + F_n} \tag{2}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

Precision will evaluate how relevant are the results obtained in the classification task, while recall will measure the truly relevant results obtained. The F1 score is *defined as the harmonic mean of precision and recall*.

2.5 k-Nearest Neighbor

The k-Nearest Neighbour or kNN algorithm is used as a method for instance-based classification [35] purposes. This means that for an unknown given instances (e.g. documents) a distance or a similarity function will be applied for prediction, based on the known instances. The algorithm will decide, based on the metric given by the distance, which is the class that a new instance will correspond. It will predict the class by choosing the k closest data points and it will select the most common class between these points. Majority voting will decide the outcome of the classifier prediction, the points which are the k-Nearest Neighbours.

The strength of this method and the reason to choose it for predicting k-nearest neighbours around some given query, is the simplicity, robustness and predicting power even in the case of a noisy training set [37]. As features for this classifier, the word2vec vectors will be used.

The main disadvantage for this algorithm is the complexity of searching the nearest neighbours for each sample, as the distance has to be calculated, for each query instance, to all the training set data. However, there are some techniques, such as K-D tree [25] short from k-dimensional tree, that could reduce this cost. In this project we made use of a powerful server which

[4] http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

allowed us to overcome this weakness.

An interesting characteristic of this algorithm, is that it does not learn from the training set, but it is using the training set to predict the label class for an unknown given instance.

2.6 Snowball Sampling

Snowball Sampling[26] is a non-probabilistic sampling technique used to *identify possible study subjects*[5], where the subjects are hard to be discovered or they are hidden (e.g. drug addicts, individuals with AIDS/HIV, prostitutes etc). The process of this method is based on a snowball effect, hence its name. In this case the existing study subjects will recruit new subjects based on their relationships, around their acquaintances. Thus, the new formed study group will grow like a rolling snowball.

In this research we will use this technique by having a small defined training set of posts (could be correlated to our study subjects) and check if we could quickly extend this set with new instances in the data which have a similar semantic word representation.

[5]http://dissertation.laerd.com/snowball-sampling.php

3 Experimental Setup

In this chapter the general setup for the environment prepared for the classifier will be described. An overview of the methodology chosen for preparing the necessary data used in the experiments, is elaborated in subsection 3.3. Subsection 3.4 will outline the methods selected for the evaluation.

3.1 Hardware & Software used

The primary work for this research project was conducted on the end-user laptop & on the TNO [27] available workstation. TNO gave access to a server with sufficient RAM memory(i.e. 48 GB), needed for the implementation of the PoC and the conducted experiments.

As for the software side, Python and bash scripting were used for developing the PoC. All the data collected from the Dark Web marketplace forums is stored into a MySQL database [28], which was queried for acquiring the dataset. The operating system on which the experimental setup was tested is Ubuntu trusty 14.04.2 LTS [29].

For the Python code, external libraries such as gensim [30], pickle [31], numpy [36], scikit-learn [32] etc. were used when developing and testing the proposed methods for the classifier.

3.2 Dataset

The aggregated data from different marketplace forums on the DarkWeb has been collected into a MySQL database. TNO provided access to this database, which consists of approximately 21 million posts. A subset of approximately 2 million such posts were selected as input dataset for our project, while the remaining were filtered out as they were marked as spam messages.

For the classification task, in order to speed up the process and get familiar with the DarkWeb possible discussions on the marketplace forums, a taxonomy list containing discovered classes from a previous research was given. Some of these classes, investigated and used in this research, can be seen in *Appendix A*.

Natural language processing performs better when cleaning the data by using filtering & pre-processing tasks. This implies stop-words removal as it pollutes the information density of the data. These stop-words include words such as days of the week, video extensions (e.g exe, avi, rar) or any other word that would interfere with the efficiency of the proposed technique and would be uninformative to the domain of interest.

However, in the context of sentiment analysis, research [39] has demonstrated that it is better to keep the data intact as it carries meaningful context. For example, "!!!" or ":))" express emotions in a context, so it is recommended to not exclude punctuation in this cases. Consequently, we chose to not remove these in this research.

3.3 Methodology

A series of steps were defined in order to structure the approach. These steps are illustrated in Figure 4.1 and explained below:

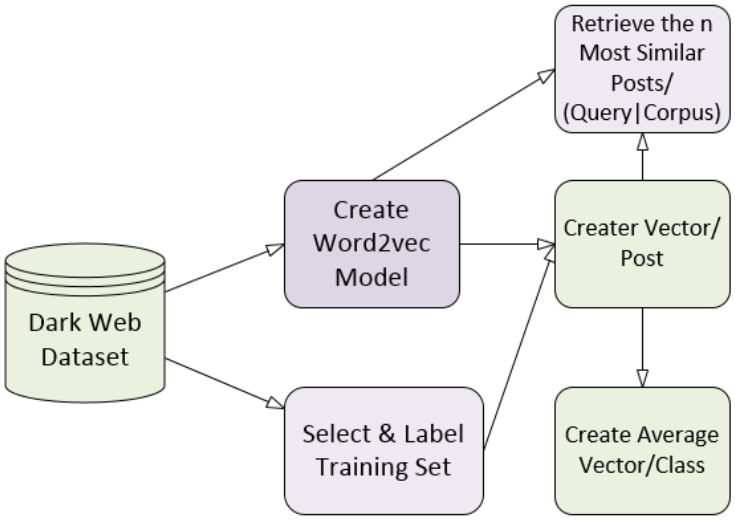

Figure 3.1: The workflow representation of the selected approach into preparing data for building the classifier.

3.3.1 Word2Vec Model

From the entire dataset, a Word2Vec model was created to form a vocabulary comprised of vector representations of words, also called *word embeddings* [7]. These vectors were used as the main features for the classifier. Thus, semantic representations for each word will be found in the constructed model.

```
>>> from gensim.models import Word2Vec
>>> model = Word2Vec.load("word2vec_model_full_300D")
>>> model.syn0.shape
(311366, 300)
```

Figure 3.2: Python request for the vocabulary size of the Word2Vec model

This model consists of 311366 unique words, as seen in Figure 3.2, and their feature vectors have 300 dimensions. When creating the model, the skip-gram method was chosen because, even if it is slower than CBOW, it gives better results and accuracy. The following command was used to create the model:

```
gensim.models.Word2Vec(tokenized_sentences, size = 300, min_count=4, workers=4)
```

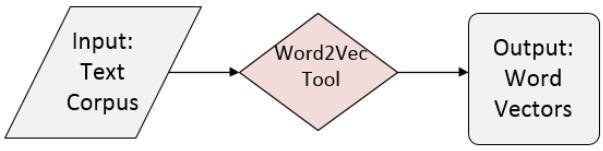

Figure 3.3: Word2Vec transformation flow

We chose 300 dimensions for the feature vectors size in order to build a good model, even if this may result in long runtimes. We selected 4 for the min_count argument, which means that words that are not repeated at least 4 times in the dataset will not be added to the model. So we take into account almost all possible words mentioned and discussed in the forum posts. We ran 4 parallel threads to increase the speed when constructing the model. This will lead to approximately 386% CPU usage, as illustrated in the Figure 3.4.

```
Tasks: 146 total,   2 running, 144 sleeping,   0 stopped,   0 zombie
%Cpu(s): 94.6 us,  2.2 sy,  0.0 ni,  3.2 id,  0.0 wa,  0.0 hi,  0.0 si,
0.0 st
KiB Mem:  47399936 total, 20419928 used, 26980008 free,    84912 buffers
KiB Swap: 48232444 total,     9372 used, 48223072 free. 10766468 cached
Mem

PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
20738 drusu     20   0 9151848 8.290g   8144 S 385.8 18.3  17:55.63 python
  886 mysql     20   0  492648  48732   6684 S   0.3  0.1  25:09.51 mysqld
```

Figure 3.4: CPU usage increases to approximately 386% when constructing the Word2Vec model

3.3.2 Post Representation

In order to create a vector, also called a representation, for each post, an average vector for the word vectors in the post will be computed. The word vectors are generated based on the model we constructed. If the words are found in the vocabulary, then the corresponding vectors will be computed. Otherwise, if a word is not found in the constructed vocabulary, it will be removed and no vector will be assigned. The computed vectors for each post will be used further in the conducted experiments.

3.3.3 Similar Posts

Using the representations of posts, we can retrieve a certain number of similar posts corresponding to a given query. This is required in the experiments when expanding the training set. We could determine the relevance and efficiency of using the word2vec model for this purpose.

A k-nearest neighbour classifier was used to find the most similar posts to a given query. A kNN model for the vectors in memory was created, that matched the given query.

3.3.4 Average *Class* Vectors

For each class discovered, see *Appendix A*, in the training set we had seed posts varying between 2 and 7. In order to measure the classification accuracy, we used *cosine similarity* for the first experiment. Computing an average

vector for each class will help us in determining the distance between a test post and all classes by measuring the angle between their vector representations.

3.3.5 Training Set

From the dataset, 300 posts were selected randomly, as we wanted to rely on the distribution of the data. These posts were labeled accordingly with the class they correspond to. Selecting random data is an important step because we could cancel-out the effect of unobservable factors. This means that we will remove the sampling bias and give to each individual post in the dataset an equal chance to be selected[6].

However, relaying on the data distribution sometimes is not the best solution as there is a high chance to select and form a training set with unrepresentative data. In this case, based on human common sense, we could have queried the database using keywords representative for some class.

We have not included any external posts in the data training set. The randomly selected items in the training set were assigned to 35 different classes. We also chose to include an *other* class for miscellaneous posts that do not fit into any of the taxonomy classes. It was a sufficient training set to get started with our classifier. Compared to our initial dataset, the training set chosen represents only 0.015% of the data(300 posts).

[6]https://explorable.com/simple-random-sampling

3.4 Experiments & Evaluation

For the conducted experiments, we selected the *Cosine Similarity and Support Vector Machines* methods to test the accuracy of the classifier, as illustrated in Figure 3.5. Each step in the diagram will be explained in more detail below.

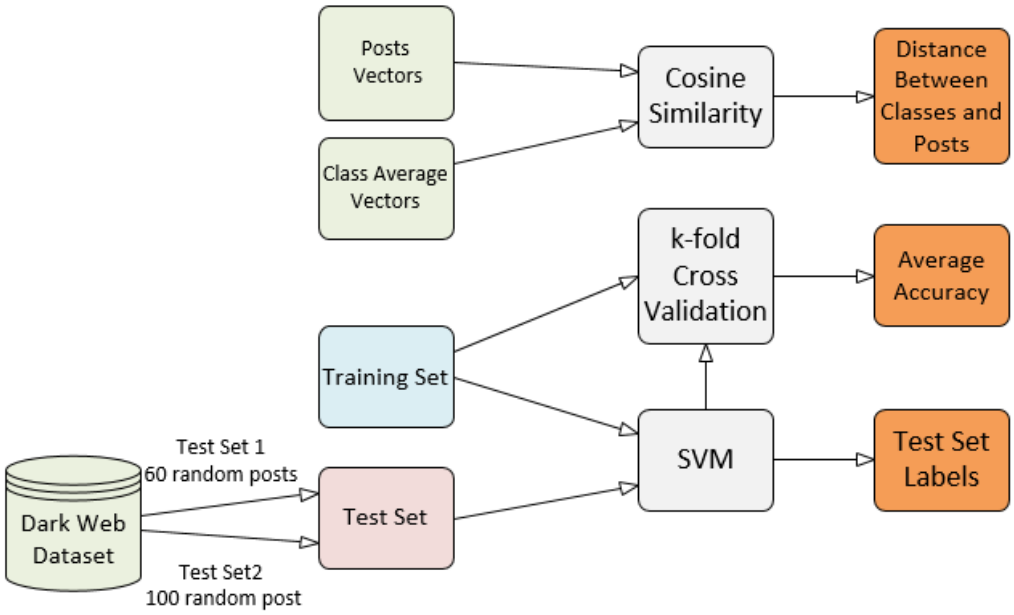

Figure 3.5: The workflow representation of the selected methods, Cosine Similarity & SVM for evaluating the classifier.

3.4.1 Test Set

As we previously built the training set by randomly selecting 300 posts out of the entire dataset, for the test set we want as well to rely on the distribution of the data. For *Test Set 1* we selected 20% data from the 300 posts set, as it was already labeled. Then we subtracted these 20% posts(i.e. 60 posts) selected for Test Set, from the Training Set, so we actually had 240 posts as Training Set initially. We also used a k-fold cross validation technique which split the initial training data in 3-folds (33%, meaning 100 posts as Test Set), respectively 5-folds (implying 20% of data, meaning 60 posts as Test Set) to evaluate the classifier.

If we would have retrieved the posts by using keyword matching for some classes that we considered as relevant, then we could have positively bias the accuracy of the classifier. For example, if we would have selected *bitcoin* as keyword when searching seeds for the class *Bitcoins*, then there is a high chance that this word will be found in all the posts corresponding to the *Bitcoins* class. This was another reason why the training set was chosen randomly and we did not alter the data retrieved.

We chose a second test set of data in order to verify if, when using a (larger) complete training set, the classifier accuracy would increase and if the results obtained using *Test Set 1* will have the same range. The second test set was selected randomly as well, but this time out of the entire dataset, and it comprised of 100 posts. Before selecting the Test Set, we subtracted the posts used for the training set from the complete forum post dataset. We need to mention that in the posts from the test set, new classes were encountered, see *Appendix A*.

We also labeled each post in the test sets. We then compared the predicted class for each post in the test set with the label assigned by a human counterpart.

3.4.2 Cosine Similarity

For the first experiment conducted, we measured the similarity of each post from the test set, with each of the class average vector(retrieved from the training set). This will produce rankings of the classes for each of the test

```
Human label - "hard_drugs"

Post 97 : " fakename wrote : i dont like street deals so i buy only here
and another markets but need a fair deal.I gave you a vendor , whose
pricesare decent for an online market . And there are a shittonne of
vendors online selling the Nijntje pills ... themostseekrit contact
details upon request But I see nothing , no eyes ... no eyes on me . "
-------------------------------------
***********Highest Rank(bottom-up)**************
TOP36: greetings - 0.22749844193458557,
.............................................................
TOP8: trading_scamming - 0.8590390682220459,
TOP7: vendors - 0.8627676367759705,
TOP6: trading_shipping - 0.8668627142906189
TOP5: financial_carding - 0.8688409924507141,
TOP4: hard_drugs - 0.8711443543434143,
TOP3: other - 0.8717963695526123
TOP2: trading_feedback - 0.8815533518791199,
TOP1: trading_recommendation - 0.8951979279518127
```

Figure 3.6: An example of the outcome produced with Cosine Similarity method when testing with Test Set 2 containing 100 posts Sample

instance. We assigned #1 Rank to the class with the highest similarity score and compared it to the *manually* assigned label for each test instance. The accuracy of this method is the percentage of the correct test instances for which the #1 ranked class corresponded to the actual label assigned from *human perspective*.

We could also define a threshold when using Cosine Similarity. For example, in Figure 3.6 we can see that Post 97, taken from the second Test Set(containing 100 posts), mentions "Nijntje pills" which is related to the "hard drugs" class. These pills are also known as "white miffy"s" or "white rabbit", named after a dutch cartoon character[38]. Therefore, when annotating the Test Set, we decided the most prominent class and the most important for which this post falls off, is the "hard drug" class.

However, if we inspect the context of this post more closely, we can see that the post embeds in its context more then just one class. Therefore,

when we applied the Cosine Similarity method over this post we can see that it actually determined with a high precision that this post discusses about "trading recommendation" and it also gives a "feedback" on personal experience. The manually assigned label can be seen in the top 4 assigned classes. Observing the values assigned, we can see that this post is not a "greeting" as it has a very low score, but it does include the classes in the top 4, excluding the *other* class, which we considered this class as a noisy defined class.

Taking all the mentioned aspects into consideration from the example provided, we could define a certain value that could filter out the classes which do not correspond to a post.

3.4.3 Support Vector Machines

As a second experiment we trained a SVM on the training set and tested it with the labeled test set. For the classifier we made use of the LinearSVC or Linear Support Vector Classification method from the sklearn python package, because it has more flexibility for the loss of functions and it is also supposed to scale better.

The training set samples contain an array, X_train, with the size of [n_samples,n_features]. Another input for the SVM algorithm is the y_train array of size [n_samples]. In our case the number of n_samples was 285 because 15 of posts could not be processed as the corresponding words were not found in the vocabulary. The number of features used for the training set corresponds to the number of features defined for our model, more exactly 300. It is important that the y_train size matches the size of n_samples from the first array, X_train.

We have defined as target names(classes) to be predicted for each post, all the classes that were discovered in the training set.

The accuracy for this method will be calculated as in the cosine similarity case, considering the top assigned label. As well we performed a k-fold cross validation method for evaluating the classifier.

3.4.4 Extending the Training Set

A final experiment had the purpose of testing if we could quickly extend the training set by adding instances from the dataset which have a similar semantic word representation. We then examined the quality of the retrieved extended data.

The approach was to semi-automatically extend the initial training set. We accomplished this by using the average vector computed for each class and retrieving the 15 most similar posts to that vector. Firstly, we excluded from the initial dataset the posts that were forming the training set. Secondly, when retrieving the similar posts of each class, we randomly evaluated retrieved posts if they corresponded in the respective class.

An important consideration here is that, when evaluating the retrieved posts, we could observe the snowball effect of the *poor annotation* on the training set. In the snowball sampling we had the property that, for each study subject used in research, the attracted population will be formed based on the acquaintances. In our case, we extended the population based on the selected initial training set. Thus, if we annotated an instance as *hard drugs*, but which is actually a product recommendation, then the instances we retrieve will also get the *hard drugs* label, while they look like a *product recommendation*, as it can be observed in Figure 3.6.

We also re-evaluated the accuracy in case of cosine similarity by using the extended training data. The results are being discussed in Section 4.

4 Results

In this research all the tests were performed by using multi-class classification, meaning that the test instances were categorized in more then two classes. We discovered and labeled thirty-six different classes from the selected training set and assigned each post to the most prominent class corresponding to it, as shown in *Appendix A*. Therefore, the results achieved during the experimental phases were accomplished by using single-labeled classification for each post.

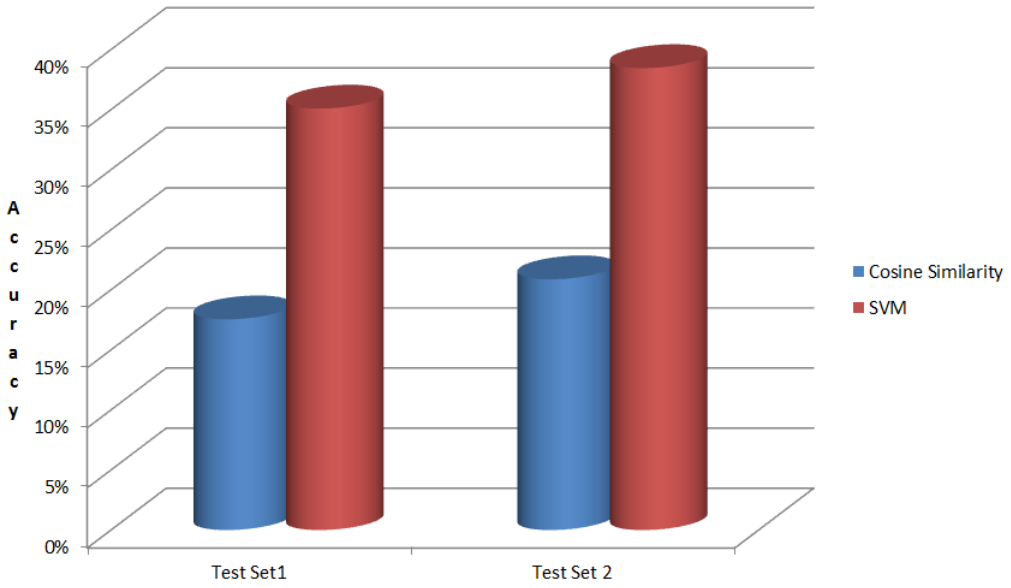

Figure 4.1: Classification accuracy for both methods Cosine Similarity & SVM, tested with two different Test Sets

In Figure 4.1 we can observe that we obtained different accuracy results for the selected test sets and methods. The difference between the accuracy of the two *test sets* comes from the fact that the training set for each of them had a different size. More precisely, for the first *test set* we selected 60 random (labeled) posts from the initial training set which was formed out of 300 posts. These test set posts were then subtracted from the initial training set resulting into a training set of 240 posts.

For the second *test set*, 100 random posts were selected from the entire dataset, excluding the posts chosen as training set. Therefore, it is a normal behaviour for the results to show a better accuracy when we use the (larger)complete training set.

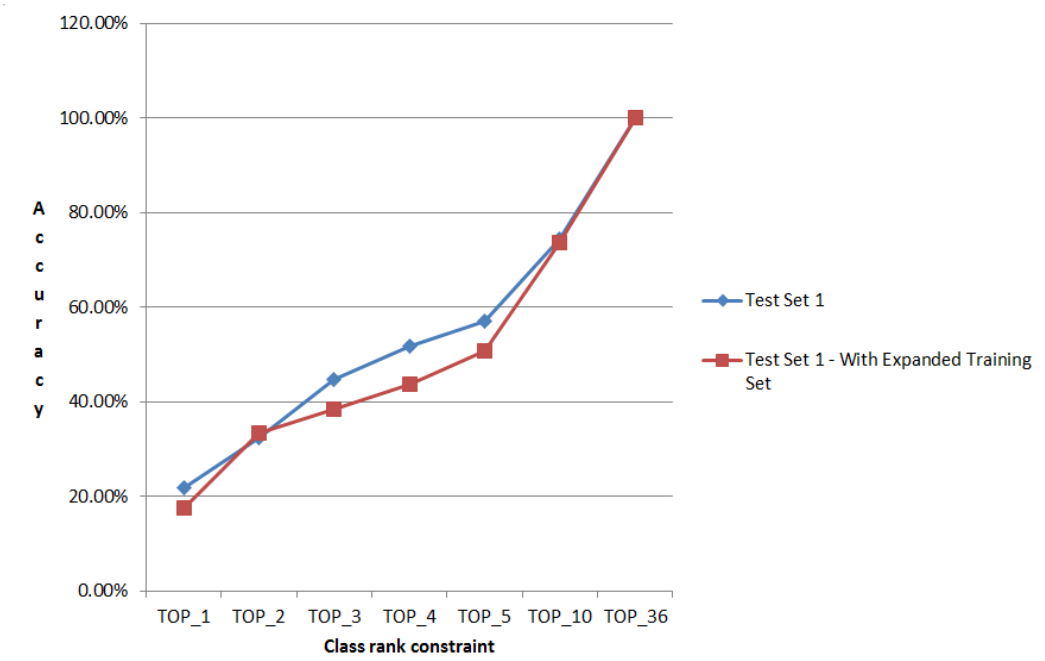

Figure 4.2: Accuracy versus class rank constraints when matching the manually assigned label for the Cosine Similarity method.

We can also observe the difference between the methods that were used in calculating the accuracy. Support vector machines performs better compared to cosine similarity when having the constraint of matching the Rank #1 label. This happens due to the fact that SVM learns from the assigned labeled class for posts in training set, however, with cosine similarity we measured the angle between each post given as test set and the average vector for each class. Thus, we did not train a classifier exactly with cosine similarity. It just gave a semantically space measurement between a vector representation of a test set post and an average vector for each class.

An accuracy of approximately 20 % was achieved for Cosine Similarity. As the result was not satisfactory, a closer inspection of the outcome values was performed. We noticed that in almost all posts from the test set, the highest value corresponding to class ranked as #1 is more accurate then the one assigned manually. Therefore, we decided to relax the constraint on the accuracy and evaluate the *top 5* predicted classes.

The results on how the accuracy increases after this step can be seen in Figure 4.2. We can observe that the results improve significantly, with a 50 % accuracy, if we consider only the *top 4* class matching. This means that the initial approach of annotating the training data with just a single label was not the best methodological choice, as multiple labels can be correct for a post and it is often hard to tell which of the correct labels is the most important, as they can be on different dimensions (content, like hard drugs, or post type, like positive review).

Table 1: The Precision-Recall and F1-score associated metrics to SVM, by running a stratified 3-fold cross validation (95 posts/fold).

	precision	recall	f1-score	support
Average / Total	0.39	0.29	0.32	285

We then evaluated the classifier training model by using a *stratified k-fold* cross validation technique and having SVM as support method. First we decided to split the data into 3-folds smaller sets. Meaning that the *training data* will consist of 200 posts, while the *test data* will comprise of 100 posts(i.e. 95 were only used because the training set consists of 285 posts). The average result including the precision, recall and F1 metrics for this output can be seen in Table 1. The outcome shows a precision of 0.39, which means that an approximately 39% post labeled as belonging to class C does indeed belong to class C. While a recall of 0.29 means that 29% posts from class C were labeled as belonging to class C. The average accuracy for the classifier in case of 3-folds is approximately 28,77%, while with 5-folds we got an accuracy of 27.72 %.

The metrics achieved with stratified 5-fold cross validation can be inspected in the Table 2. Each fold in this case consisted of 57 posts because the training set comprised of 285 posts. Compared to the 3-fold results, precision increases with 0.03, while recall decreases with 0.01. The F1-score associated in both cases, of 3-folds and 5-folds is however, the same.

Table 2: The Precision-Recall and F1-score associated metrics to SVM, by running a stratified 5-fold cross validation(57 posts/fold).

	precision	recall	f1-score	support
Average / Total	0.42	0.28	0.32	285

Considering the experiment performed with *test set 1* on the extended training set, we can observe a lower accuracy in that case, as illustrated in Figure 4.2. The reason for which the re-evaluation produced a sub-optimal result is the data which was added to the initial training set.

We extended the initial training set for each class by finding the closest 15 neighbours to the average vector for a specific class. We then randomly checked if the added posts match with the corresponding class. We observed that when retrieving the closest neighbours, we actually increased the noise in our training set, because some of the posts did not exactly belong to the targeted class. It means that we got the snowball effect of the *single-label* annotation approach. Therefore, in the future work it would be recommended to have *multi-label* annotation. If we inspect the initial performed test for the Cosine Similarity, Figure 4.1, we can see that we achieved only 20 % accuracy. We can estimate that 80 % inaccurate data engaged more faulty data, resulting into a noisy training set.

5 Conclusions

In this research we investigated the usage of semantic word representations, built with Word2Vec technique, on a Dark Web dataset in order to boost the training process by speeding-up the annotation work of forum post classifiers. To achieve this purpose, we selected the cosine similarity and support vector machines methods to measure the accuracy of the classification task results and the k-nearest neighbour method to retrieve the most similar posts.

For the cosine similarity method, when using word embeddings, we achieved only 20% of accuracy from the first run, as detailed in Section 4. In comparison to the cosine similarity, SVM method performed better, with a 39% accuracy.

We also ran a stratified k-fold cross validation technique to measure the accuracy of the training model used for classification task. We tested a 3-folds split on the training data and achieved approximately 28,77% accuracy, while with 5-folds smaller sets we got a result of almost 27.72 %.

The accuracy of cosine similarity improves significantly if we relax the constraints and allow a positive match result in the *top 4* values assigned by the classifier when comparing with the manually labeled classes. The accuracy increases to 50% in this case.

We tested if we could swiftly extend the training set, by adding similar posts corresponding to each class in the training set. We noticed the snowball effect due to the non-optimal *single-label* annotation which populated our initial training set with noisy data.

If we improve a small training set to use correct multi-class labels for each post, it is feasible to use word representations as features for a classifier and extend the training set, in order to get a quick thematic insight over the discussion forums in the Dark Web.

6 Future Work

Considering the small, single-class labeled training set we achieved a satisfactory accuracy result for the classifier. As a future step for improving the classification accuracy, assignment of multi-label classes for each post in the training set could be tested. Due to the highly specialized nature of this topic, having the right knowledge in defining the initial training set is crucial. In this way we could increase the "intelligence" of the classifier. Lastly, for a more precise annotation of the training set documents, at least two people have to review the labeling for each post.

The taxonomy class used for the classifier, comprised of only 36 classes which were discovered when annotating the training set. For the new documents used in the test set, overall model accuracy will decrease if a document contains posts that do not discuss subjects related to the classes defined. Having 5 to 10 seed posts for a more extended taxonomy class would support the increase of the classifier accuracy.

We focused on a supervised method for the classification purpose, which requires manual labeling. To automate this task, unsupervised method, such as clustering could be used to form clusters of data. The available set of classes could be further extended and checked if the clusters formed correspond to a particular class that we are missing. In this way a minimal *human effort* would be required when building a classifier.

We could explore another technique to build a model for the classifier that uses word representation, an extension of word2vec named doc2vec, which is still in a development phase[40].

Considering the results achieved, we would suggest integration with the `darkwebmonitor.eu` portal, as it could prove to be a valuable asset for the analysts using the platform and investigating *illicit* cases.

Appendix A Classes

- Discovered classes in Training Set:

```
"Bitcoins"
"soft_drugs"
"other"
"trading_shipping"
"thanks"
"trading_scamming"
"encryption"
"financial_carding"
"financial_prices"
"security"
"financial_paypal"
"hard_drugs"
"accounts"
"contests"
"apologizes"
"trading_hardware"
"trading_recommendation"
"vendors"
"product_offer"
"hitman"
"trading_feedback"
"marketplaces"
"anonimity"
"greetings"
"hacking_malware"
"financial_money_laundering"
"arrest"
"escrow"
"hacking_ddos"
"hacking_cracking"
"software_recommendation"
"financial_loans"
"religion"
"business_closed"
```

"psychological_help"
"fake_software"

- Discovered new classes in Test Set(100 posts), besides the aforementioned ones from the Training Set:

  "pornography_adult"
  "pornography_child"

References

[1] Europol, *The Internet Organized Crime Threat Assessment (iOCTA) 2014*

[2] The Tor Project, Inc. "Tor Project: Anonymity Online". Torproject.org. `https://www.torproject.org/`

[3] Geti2p.net,. "I2P Anonymous Network". `https://geti2p.net/`

[4] Chacos, Brad, and Brad Chacos. "Meet Darknet, The Hidden, Anonymous Underbelly Of The Searchable Web". PCWorld. `http://www.pcworld.com/article/2046227/meet-darknet-the-hidden-anonymous-underbelly-of-the-searchable-web.html`

[5] Martijn Spitters, Femke Klaver, Gijs Koot, Mark van Staalduinen *Authorship Analysis on Dark Marketplace Forums*, IEEE European Intelligence & Security Informatics Conference (EISIC) at Manchester, UK

[6] Stephen Loiaconi, Sinclair Broadcast Group. "FBI: ISIS 'Going Dark,' New Encryption Poses New Threat". Fox11online.com. `http://fox11online.com/news/nation-world/fbi-isis-going-dark-new-encryption-poses-new-threat`

[7] Tomas Mikolov, Ilya Sutskever et al., *Distributed Representations of Words and Phrases and their Compositionality* , Proceedings of NIPS, 2013

[8] Quoc Le, Tomas Mikolov, *Distributed Representations of Sentences and Documents*, ICML, 2014

[9] Xuan-Hieu Phan, Le-Minh Nguyen, Susumu Horiguchi, *Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections*, WWW Conference 2008

[10] Duyu Tang, Furu We et. al, *Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification*, ACL 2014

[11] Yoon Kim, *Convolutional Neural Networks for Sentence Classification*, EMNLP 2014

[12] Sumit Bhatia, Prakhar Biyani and Prasenjit Mitra, *Classifying User Messages For Managing Web Forum Data*, WebDB 2012

[13] Noam Shazeer, Joris Pelemans, Ciprian Chelba *Skip-gram Language Modeling Using Sparse Non-negative Matrix Probability Estimation*

[14] K.Kanagavalli, S.T.Tharani, *Analysing User Posts for Web Forum using K-means Clustering*, International Journal of Scientific and Research Publications, Volume 4, Issue 5, May 2014

[15] Chris Nicholson, Adam Gibson. "Deeplearning4j - Open-Source, Distributed Deep Learning For The JVM". Deeplearning4j.org.

[16] Tfidf.com,. "Tf-Idf : A Single-Page Tutorial - Information Retrieval And Text Mining". `http://www.tfidf.com/`

[17] Scikit-learn.org,. "3.1. Cross-Validation: Evaluating Estimator Performance — Scikit-Learn 0.16.1 Documentation". `http://scikit-learn.org/stable/modules/cross_validation.html`

[18] Big Data for Commerce - Lokad,. "Overfitting: When Accuracy Measure Goes Wrong". `http://blog.lokad.com/journal/2009/4/22/overfitting-when-accuracy-measure-goes-wrong.html`

[19] Nlp.stanford.edu,. "Choosing What Kind Of Classifier To Use"

[20] Youngjoong Ko (2012) *A study of term weighting schemes using class information for text classification*, SIGIR 2012

[21] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean*Efficient Estimation of Word Representations in Vector Space* , arXiv:1301.3781v3 2013

[22] Multithreaded.stitchfix.com,. "A Word Is Worth A Thousand Vectors — Stitch Fix Technology – Multithreaded" `http://multithreaded.stitchfix.com/blog/2015/03/11/word-is-worth-a-thousand-vectors/`

[23] P.-N. Tan, M. Steinbach & V. Kumar, “Introduction to Data Mining”, , Addison-Wesley (2005)

[24] Powers, David M W (2011). textitEvaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation (PDF). Journal of ”Learning Technologies 2 (1): 37–63

[25] Andrew W Moore, *An intoductory tutorial on kdtrees*

[26] Explorable.com,. “Snowball Sampling - Chain Referral Sampling”, https://explorable.com/snowball-sampling

[27] TNO,. ”TNO - Innovation For Life — TNO”, https://www.tno.nl/nl/

[28] Mysql.com,. “Mysql : The World”s Most Popular Open Source Database”

[29] Releases.ubuntu.com,. “Ubuntu 14.04.2 LTS (Trusty Tahr)”

[30] Radimrehurek.com,. “Gensim: Topic Modelling For Humans”

[31] Docs.python.org,. “11.1. Pickle — Python Object Serialization — Python 2.7.10 Documentation”, https://docs.python.org/2/library/pickle.html

[32] Scikit-learn.org,. “1.4. Support Vector ”s — Scikit-Learn 0.16.1 Documentation” http://scikit-learn.org/stable/modules/svm.html

[33] Docs.opencv.org,. “Introduction To Support Vector ”s — Opencv 2.4.11.0 Documentation”, http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[34] Statsoft.com,. “Support Vector ”s (SVM)”, http://www.statsoft.com/Textbook/Support-Vector-"s

[35] Fon.hum.uva.nl,. “Knn Classifiers 1. What Is A Knn Classifier?” http://www.fon.hum.uva.nl/praat/manual/kNN_classifiers_1__What_is_a_kNN_classifier_.html

[36] Numpy.org,. “Numpy — Numpy”, http://www.numpy.org/

[37] Application of K-Nearest Neighbor (KNN) Approach forPredicting Economic Events: Theoretical Background, *Sadegh Bafandeh Imandoust And Mohammad Bolandraftar*

[38] Pillreports.net,. "Pill Reports - Ecstasy Test Results Database By Enlighten". `http://www.pillreports.net/index.php?page=display_pill&id=33416`

[39] Ravi Parikh, Matin Movassate *Sentiment Analysis of User-Generated Twitter Updates using Various Classification Techniques*, `http://nlp.stanford.edu/courses/cs224n/2009/fp/19.pdf`

[40] Rare-technologies.com,. "Doc2vec Tutorial » Rare Technologies", `http://rare-technologies.com/doc2vec-tutorial/`