CLASSIFICATION OF ILLEGAL ADVERTISEMENT
WORKING WITH IMBALANCED CLASS DISTRIBUTIONS USING MACHINE LEARNING

Hampus Adamsson
Abstract

CLASSIFICATION OF ILLEGAL ADVERTISEMENT

Hampus Adamsson

Interpreting human language entered a new era with the current prevalence of machine learning techniques. The field of natural language processing (NLP) concerns the interaction between human and computer languages. Machine learning is the scientific area involved with the design of algorithms that learn from past experience. These two areas within computer science – NLP and machine learning – enable complex ways of analyzing and working with written language.

The goal of this thesis is to implement a prototype that automatically find web-based advertisements related to illicit content. In such a scenario, the source material comes mostly in the form of unstructured-, unlabeled-, raw text.

In order to design a working algorithm in this context, a combination of machine learning and NLP techniques are used. Three machine learning algorithms were used in order to classify material by content: nearest neighbours algorithms, support vector machines, and multilayer perceptron.

Two dimensionality reduction techniques were used: principle component analysis and Latent Dirichlet allocation. It should be noted that the latter is more commonly used as a method of explaining documents in terms of topic mixtures, rather than reducing dimensionality in a problem.

NLP representation techniques such as the bag-of-word model and TF/IDF were used. These are essentially different variations of word embedding. The result is evaluated using metrics common in information retrieval: precision, recall, F1 measure, and confusion matrices.

We found that an important challenge in this context is class imbalance: content of interest is often overshadowed by data representing over-represented classes. Another important challenge is that there are multiple classes, and an accurate labelling of such ontology is often missing. In order to improve the accuracy, more annotated - historical - data is needed.

This thesis is published in collaboration with CGI, on behalf of the Swedish Financial coalition.
KLASSIFICERING AV ILLEGAL ANNONSERING

POPULÄRVETENSkapLIG SAMMANFATTNING


Tre olika modeller har utvärderats och testats för att realisera prototypen och klassificera material på Internet. Detta arbetar specifikt mot material som utgörs av annonser av olagligt material. Tanken var att kartlägga en specifik domän och implementera en modell som automatiskt identifierar vilken typ av produkt som utannonseras. Avslutningsvis så testas modellen på en annan del av Internet för att kartlägga hur väl modellen presterar under förändrade förhållanden. Detta anses vara ett mått på hur väl modellen kan generalisera.

När mer strukturerad data har skapats i tidigare processer så kan vi kartlägga att obalanserad data utgör hela vårt källmaterial. Detta leder naturligtvis till en speciell typ av modellering för att ta hänsyn till hur man bäst arbetar mot just obalanserade dataset. Således utvärderas olika typer av metoder för att motarbeta obalansen, och även olika typer av processer och verktyg för att tolka resultatet.
SPECIAL NOTES

This thesis is published in collaboration with CGI – who supervised, and directed the work – on behalf of the Swedish Financial Coalition. The original intent was to present a method to combat child abuse material on the Internet; emphasizing on transactions and payments methods. However, given the restrictions – laws, policies, and rules – and a limited supply of relevant content (data), we were forced to revise the scope in order to move forward. Some subcategories (e.g. child abuse material) are especially difficult to investigate due to security issues.

It is my opinion that the general approach of this thesis can be applied to other domains, hopefully with similar results. Incorporating child abuses material into the source material ought to, at least theoretically, produce result similar to that of some classes found in the result section. However, I leave speculations to whom this thesis may concern.
## CONTENTS

1 Background
   1.1 Goals and objectives ........................................... 1
   1.2 Threats to validity ........................................... 1
       1.2.1 Dataset content ......................................... 1
       1.2.2 Dataset labels ........................................... 2
       1.2.3 Model calibration ......................................... 2

2 Related work
   2.1 Imbalanced data ............................................. 3
   2.2 Natural language processing ................................. 3
   2.3 Domain ......................................................... 4

3 Theory
   3.1 Feature representation ....................................... 5
       3.1.1 Bag-of-words model .................................... 5
       3.1.2 Term frequency .......................................... 7
       3.1.3 Term frequency–inverse document frequency ... 7
   3.2 Dimensionality reduction .................................... 7
       3.2.1 Principal component analysis ........................................... 8
       3.2.2 Latent Dirichlet allocation ......................................... 9
   3.3 Classification models ......................................... 11
       3.3.1 Nearest neighbors algorithm ............................... 11
       3.3.2 Support vector machine .................................... 11
       3.3.3 Multilayer perceptron ...................................... 12
   3.4 Hyperparameter tuning ......................................... 14
       3.4.1 \(K\)-fold cross-validation ...................................... 14
       3.4.2 Parameter search ........................................... 15
       3.4.3 Sample weights ............................................ 15
   3.5 Evaluation ........................................................ 15
       3.5.1 Classification accuracy ...................................... 15
       3.5.2 Confusion matrix .......................................... 15
       3.5.3 Recall and precision ....................................... 16
       3.5.4 \(F_1\) measure ............................................. 16

4 Data
   4.1 Data sampling .................................................. 17
       4.1.1 Labelling samples .......................................... 17

5 Method
   5.1 Feature selection .............................................. 20
       5.1.1 Feature sampling .......................................... 20
       5.1.2 Feature sampling: other sources .......................... 23
       5.1.3 Feature representation ..................................... 23
   5.2 Dimensionality reduction impact on features ............ 23
       5.2.1 Latent Dirichlet allocation ................................. 23
       5.2.2 Principle component analysis ............................... 25
   5.3 Tuning classifiers .............................................. 26
5.3.1 Evaluation metric ........................................... 26
5.3.2 Nearest neighbors algorithm .............................. 27
5.3.3 Support vector machine ................................... 29
5.3.4 Multilayer perceptron ..................................... 31

6 Result ............................................................. 33
6.1 Feature representation ....................................... 33
6.2 Dimensionality reduction .................................... 33
6.2.1 PCA ....................................................... 33
6.2.2 LDA ....................................................... 33
6.3 Data imbalance ................................................ 33
6.4 Classification result .......................................... 33
6.5 Cross domain classification ................................. 37

7 Discussion ....................................................... 39
7.1 Result .......................................................... 39
7.2 Scalability ...................................................... 40
7.3 KNN ............................................................ 41
7.4 SVM ............................................................ 41
7.5 MLP ............................................................. 41

8 Future works ..................................................... 43
8.1 Incorporating images ......................................... 43
8.2 Advanced feature engineering .............................. 43
8.3 Extending the dataset ........................................ 43

9 Summary ......................................................... 43

References ......................................................... 45

10 Appendix ........................................................ 47
1 BACKGROUND

The Internet is a place where various people come together; sharing ideas, utter thoughts, and speak their minds. The freedom of speech that prevails on the Internet is perhaps only restricted by accountability. Intuitively to most people, we are responsible for our actions. However, when anonymity is valued more than accountability we find ourselves in a situation where people are not required to take responsibility for their actions. One place that is particularly influenced by this phenomena is Darknet. This area emphasis anonymity as a virtue; creating a haven for people seeking to avoid traceability. Unfortunately, but not entirely unexpectedly, this attracts much attention from people with illicit intent.

Darknet markets are notoriously known for advertising illicit goods. Although the content is primarily dominated by drug related material, all kinds of dubious objects and work are being advertised; from child abuse material to contract murder. Law enforcement agencies are being overwhelmed by the vast collection of illegal material that emerge on Darknet markets. Traditional law enforcement agencies attempt to combat the markets by reconnaissance work, or by following up on leads submitted by concerned individuals. Either method require much time consuming manual labour that could be spent elsewhere.

Another problem is that Darknet websites exhibit fluctuating uptime and seldom persist for longer than 18 months. On top of this, the majority of the websites disconnect for a prolonged duration on average every second day[1]. Subsequently it is extremely difficult to follow up on leads due to the limited uptime of arbitrary Darknet websites. Investigations are bound to come up empty handed given the conditions that surrounds the nature of Darknet markets. Shortening the time dilation between identification and action is instrumental in order to avoid that scenario. This could potentially be accomplished, and valuable time could be saved, by introducing an automated system that performs the preliminary filtering. Such a filter could potentially limit the amount of data that law enforcement agencies have to process. This background is the motivation and basis of this thesis.

1.1 Goals and objectives

The goal of this thesis is to evaluate appropriate models and feature selection algorithms in order to filter though unstructured text. The implementation should assign topics to arbitrary text by evaluating the content. The purpose of the implementation is to act as a filter and minimize the amount of data that has to be processed in order to investigate material associated with a specific topic. This is commonly referred to as topic modelling. Implementing and tuning a topic model is an introductory first steps in achieving this goal. This thesis aim to evaluate the problems that arise - both technical and domain specific - when targeting Darknet market content.

1) Create a prototype that classify Darknet advertisement of illicit goods.
2) Evaluate the result of classifying imbalanced category distributions (skewed classes).
3) Evaluate the performance of the prototype and investigate the potential of deploying the model in a more general domain.

1.2 Threats to validity

1.2.1 Dataset content

The data is gathered and labelled automatically. Roughly 10% has been manually validated to ensure that the content is both comprehensible and intuitively labelled. However, this is a collection of unstructured data, and not a predefined dataset with labels. The data is both washed and labelled as a part of the preprocessing. This ought to be kept in mind when evaluating the result.
1.2.2 Dataset labels

The dataset contains Darknet market listings (advertisement of merchandise). Listings are created by merchants and product categories are set manually. The dataset classes are then extracted from the websites by targeting the categories. Some categories are overlapping (Drug listings), while some categories are ambiguous (Other listings). The result is that the dataset itself is quite difficult to interpret. Arguments can be made that even people would have difficulties labelling the content.

The dataset is split into two smaller datasets: Alphabay and Poseidon. The category system differ between the two domains. The manually provided mapping should be treated as a rough estimate and a discrepancy.

1.2.3 Model calibration

Model calibration is paramount. Slight changes to hyperparameters can either enhance or diminish a models performance, and often by working in unison with other hyperparameters. Models with few hyperparameters are often regarded as easier to calibrate, while the effect is reversed for models with many hyperparameters. Some models have more hyperparameter and tend to require more calibration.
2 RELATED WORK

2.1 Imbalanced data

Imbalanced data (or skewed data) is a problem that requires special attention, and different methods depending on the domain. Yan et. al. discuss skewed data in multimedia data when working with neural networks[2]. Ghanavati et. al. work on a similar problem, albeit focusing more specifically on data imbalance in Big data[3]. It is interesting to observe how different angles of attack impact the reasoning. Furthermore, Yu and Ni discuss the effect of imbalanced in high dimensional biomedicine data[4]. This is particularly interesting seeing how text processing often lead to high dimensional data problems.

Zhang et. al discuss different ways of classifying imbalanced data using support vector machines (SVM). This is an in-depth evaluation of the SVM in a problem environment similar to that of the problem in this thesis. Different sampling techniques (emphasising under- and oversampling) are important source material that served as a basis for the method used in this thesis[5].

Shen et. al. investigate secondary sampling techniques as a mean to reduce dimensionality in high dimensional data. This work relates to the problems that arise when working with high dimensional data rather than imbalanced data. However, this is also subject to this thesis, and something that needs to be addressed[6].

2.2 Natural language processing

Incorporating entity correlation knowledge into topic modelling can be a difficult task depending on the source material. This thesis describe the importance of entity names as well as how to incorporate this into a topic model[7].

Clustering search engine suggests by integrating a topic model and word embeddings is an interesting take on how to utilize the feature representation in an efficient manner. Furthermore, this is also important when considering how to make use of the result found by the model; or how to interact with the implementation. Nie et. al. is important reading for those interested in pursuing the next step of this implementation[8].

Sentiment analysis is a branch of natural language processing where a subjective state is embedded in a phrase or wording. It is essentially trying to derive emotions from words, and incorporate this information during the classification phase. A comparison (Walaa Medhat et. al[9]) of current implementations shows that the field is still improving. This claim is backed by (Singh and Kumari[10]) where custom implementations exceeds the performance of state-of-the-art techniques, and that sentiment analysis can be used successfully on arbitrary text classification tasks. Sentiment analysis can be observed in real-world applications where sentiment analysis is used to predict stock market values (Bollen et. al.[11]) Thus backing the claim by Walaa Medhat et. al[9].

Part-of-speech (POS) tagging is the concept of grouping words or lexical items based on grammatical properties. Nouns, verbs, conjunctions, and pronouns are examples of POS in the English language. Some words play similar roles within the grammatical structure of sentences, and can thus be used more efficiently in tasks such as classification. Makazhanov and Yessenbayev show that character based feature extraction for POS-tagging is possible. Feature extraction, and feature engineering, is one of the most important processes that enables classification; new techniques to do so is bound to impact arbitrary classification tasks such as this one[12].

POS-tagging for agglutinative languages is something that is a synthetic, morphological language that uses agglutination in order to better understand the meaning of texts. Stemming and lemmatization are two important methods that breach the surface of morphology in arbitrary classification tasks such as this one.
Making these techniques more efficient align with the method of this thesis. This has been done in detail by Bölcü and Can[13].

2.3 Domain

Domain specific information- and knowledge is paramount when implementing a model to evaluate any domain. Lexical-, grammatical-, and morphology information is arguably enough to make any domain unique and differentiable. Topic models ought to consider the operational domain in which it operates. Owen and Savage have an intuitive take on the domain in which this model operates[1].

Coudriaau et. al. investigate the content using topological analysis. This is more of a technical perspective of the infrastructure, while also describing the anonymous nature of the domain. I think it is one of the reasons why people express themselves as they do; they are anonymous[14].
3 Theory

3.1 Feature representation

Topic modelling is an old field within NLP. The goal of any topic modelling algorithm is to provide a topic out of a predefined set of categories. This task might seem trivial to humans, but pose a difficult problem to machines. We need to tackle problems such as ambiguity, metaphors and negations. There are a vast number of seemingly trivial concepts that might alter the paradigm when considering NLP.

Written language is a way of representing spoken or gestural language. This make sense to humans since we map sound to meaning, and written language is simply an extension of this system. Thus text can be interpreted as a reference to sound, which is a reference to meaning. This goes back for generations, evolving without any apparent destination. The absence of logic and structure (in a purely mathematical sense) is arguably the reason why NLP pose such difficulties for computers. Computers do not process text as humans, but still need a way to distinguish between words. One way to map human understanding to computers is by carefully selecting features and assign indices to them. This is why features, and feature extraction is such an important part of NLP.

There are multiple ways of interpreting text in terms of features. This thesis only considers the one-hot vector representation, but some alternative methods can be found in the related works chapter. The one-hot vector representation is simply an array that contains all of the features in one representation. Vectorization of characters, words, sentences, or documents are the primary features that are transformed into the one-hot vector representation. It should be noted that more features can be derived from these (e.g. average word length or unique word frequency).

3.1.1 Bag-of-words model

The bag-of-words model is a way of representing documents in terms of tokens. Tokens can be thought of as words, albeit with the addition that any symbol (or set of symbols) can be conceived as tokens. Unique tokens in a set of documents are stored in a vocabulary. The model discards grammar and token order, but maintains token multiplicity. This makes the bag-of-words model apt for tasks such as topic classifications, but less useful in areas involving semantic classification[15].

Tokens can be obtained by separating documents into smaller chunks of text. This procedure is repeated for all documents, and the resulting list of tokens (or chunks) are stripped of all duplicates. Thus obtaining the bag-of-words. The default implementation often found in literature is referring to tokens as sub-strings between delimiters in the absence of a predefined vocabulary. It is the standard method denoted as \(b\) in regular expressions (eg. space, comma, dot)[16]. However, this process is customizable and tokens can represent characters, prefixes, suffixes, and/or sentences. The process is often regulated by regular expression, and regardless of how tokens are being extracted, the result is that the bag (in the bag-of-words) contains as many elements as there are unique tokens in all of the combined documents.

\[
\begin{align*}
\text{str1} &= ['You', 'are'] \\
\text{str2} &= ['Who', 'are', 'you'] \\
\text{bag} &= \text{set}(\text{str1}+\text{str2})
\end{align*}
\]

In [2]:

\{'You', 'are', 'Who'\}

An application of the Bag-of-words model is illustrated in table 1. The bag display token frequency where each column represent a unique token, and each row represent a document.

Constraints can be added to the sampling process in order to modify the bag-of-words model. A common practice is to discard stopwords - common words that has little lexical
information other than readability (e.g. is, a, the). This is a static method that relies on a predefined vocabulary that restrain tokens from being added to the bag-of-words. The stop-words vocabulary featured in this implementation can be found on the Scikit-learn website (www.scikit-learn.org)[17], along with a list of other constraints. A dynamic method that serves a similar purpose is that of looking at the occurrence of tokens across all documents. Tokens that occur in every document can not be used to distinguish between documents and will not impact any topic classification. Similarly, removing tokens that occur in many documents (e.g. 90%) can be removed in a similar fashion without significantly impacting the classification process. It is generally a barter between preserving information and limiting the feature space. The process is also used to discard rare tokens (e.g. entity names) by adding another prerequisite. The requirement is that tokens have a minimum document occurrence frequency (e.g. 1%) in order to appear in the bag-of-words. Removing tokens in this manner may reduce complexity and thus make the problem less susceptible to overfitting. It is especially true for rare tokens since they are likely to map exclusively to individual documents. This leads to a one-to-one relationship between documents and tokens, thus limiting the generalization of the model[18].

It is also possible to sample multiple tokens and treat it as a single token - \(N\)-grams. \(N\) equals the number of tokens that are treated as a conjunction. \(N\)-grams can potentially add some instance of semantics to the model since token ordering is taken into account. This method tend to result in more features than the default unigram (1-gram) sampling method devoid of conjunctions.

Character sampling is another method where tokens are treated in a similar, albeit completely different way. Rather than looking at tokens, or even conjunction of tokens, the character sampling method treats individual characters as features. This method tend to result in less features than tokens separated by delimiters. This method can be used in combination with \(n\)-gram.

The lemmatization constraint is a method to group inflected forms of a word into one unified representation (figure 1). Lemmatization works by considering the morphological analysis of words. The intention is to find the root of a particular word - the lemma. This is implemented by a huge dictionary that enables the algorithm to find the lemma. Consider a scenario where a topic classification model has been trained on millions of meeting protocols. Archives - such as meeting protocols - are generally written in past tense[19]. If the model is exposed to a new protocol in present continuous tense, the model would arguably perform poorly because tokens present in the document would not appear in the bag-of-words model. The lemmatization constraint can theoretically nullify the impact of grammar as explained in this example by simply reverting back to a unified representation. One advantage of lemmatization is the ability to limit the grammatical impact on feature representation[20][21]. The lemmatizer featured in this implementation is the Natural Language Toolkit’s WordNet[18].

Stemming is similar to lemmatizing in the sense that both methods aim to group inflected forms of a word into one unified representation. Where lemmatization rely on a dictionary, stemming rely on heuristics. It is not an exact science, and most implementations simply cut off the end of words. This is especially limiting in terms of ambiguity (noise). Hence the uncertainty[21]. The stemmer featured in this implementation is the Natural Language Toolkit’s Porter stemmer[20].

Fig. 1. Grouping of inflected words using lemmatisation
Table 1: Term frequency representation using the Bag-of-words model

<table>
<thead>
<tr>
<th></th>
<th>You are</th>
<th>are</th>
<th>Who</th>
</tr>
</thead>
<tbody>
<tr>
<td>You are</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Who are you</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

3.1.2 Term frequency

Document representation is achieved by using a vector space model. The vector space is itself based on the underlying bag-of-words model where it acts like a vocabulary. Vectors correspond to tokens, and document will be represented by the entire vector space. The model keeps track of how many times each token occurs in each document (table 1).

3.1.3 Term frequency–inverse document frequency

Term frequency–inverse document frequency (tf-idf) is also an extension of the bag-of-words model where token importance is incorporated in the representation. Tf-idf is a vector space model similar to the term frequency representation. Tokens are valued based on how often they occur in a document. This process is regulated by two separate statistical models - term frequency (tf) and inverse document frequency (idf). Rare tokens are considered more valuable than common tokens when compared to all documents within a corpus (idf). The process is then inverted, common tokens in a specific document are considered more valuable than rare tokens (tf). In this inverted stage the rare tokens receive a higher value, which limits the impact of common words in all of the combined documents[22]. The concept is that rare words that occur often in a specific document are more likely to be of importance than simply relying on either method alone. Two implementations of tf and idf can be found in equation 1 and 2[21][23].

\[
\begin{align*}
 tf(t, d) &= \frac{f_{t,d}}{\sum f_{t',d}} \\
 f_{t,d} &= \text{frequency of } t \text{ in } d \\
 d &= \text{document} \\
 t &= \text{token}
\end{align*}
\]

(1)

\[
\begin{align*}
 idf(t, D) &= \log \frac{N}{|d \in D : t \in d|} \\
 D &= \text{total number of document} \\
 N &= \text{all documents} \\
 d &= \text{document} \\
 t &= \text{tokens}
\end{align*}
\]

(2)

The combination of tf and idf is can be found in equation 3[23].

\[
\begin{align*}
 tf-idf(t, d) &= tf(t, d) \times idf(t, D)
\end{align*}
\]

(3)

Some slight modifications to the default implementation have been made in order to increase performance. Sublinear term frequency scaling is added to incorporate a logarithmic scale of the true term frequency. The result is that tokens observations are valued more than the repetition of tokens. Hence the name sublinear scaling. The implementation can be found in equation 4[21].

\[
\begin{align*}
 tf(t, d) &= \begin{cases} 
 1 + \log(tf_{t,d}) & \text{if } tf_{t,d} > 0 \\
 0 & \text{if } tf_{t,d} \leq 0
\end{cases}
\end{align*}
\]

(4)

3.2 Dimensionality reduction

Working with high-dimensional feature is common practise when opting for the bag-of-words model. It is a consequence of adding additional features as new unique tokens are introduced. These features can later be used by an estimator in order to evaluate similarities between documents. Training such an estimator in a high-dimensional feature space requires a considerable amount of training data in order to explore
a sufficient amount of input permutations. This is important in order to maximize generalization - to perform well on unseen data. The phenomena is known as Hughes phenomenon or the curse of dimensionality[24].

The obvious ramifications of working under Hughes phenomenon are increased demand on both performance and capacity. Simply maintaining a large enough vocabulary is likely to strain a system’s memory. Another implication is that overfitting is more likely to occur due to sparse input data. This can be found in equation 5. The figure features a bag-of-words model with tf-idf representation[24].

\[
\text{Number of input permutations} = x^n \\
\text{n} = \text{number of features} \\
x = \text{tf-idf}(t,d) \\
x \in [0, 1] \\
n \in R
\]

It is possible to put limits on the feature space by adding additional requirements to the sampling process (e.g. using stopwords, as seen in figure 13 and figure 14). However, features matching the requirements are not necessarily important, mainly due to correlations to other features. This is why dimensionality reduction techniques may prove useful even though static requirements are used in the sampling process. Colinear dependencies are the focus of one of the dimensionality reduction techniques applied in this thesis.

### 3.2.1 Principal component analysis

Principal component analysis (PCA) is an unsupervised linear transformation algorithm that can be used for dimensionality reduction. The algorithm projects high-dimensional data onto a new subspace of fewer or equal amount of dimensions. PCA is a suitable tool for identifying patterns in the input data, or simply visualizing data[25][23]. Furthermore, PCA can potentially increase performance of classification tasks[26].

Principal components are constructed from orthogonal vectors, which means that all vectors are pairwise perpendicular. Each principal component is chosen from a subset of candidates - pairwise perpendicular to one another - to maximize the variance on the projected component. This can be observed in figure 2.

![Fig. 2. The first principal component maximizes the variance of the data along one axis. The same concept applies to subsequent components, while also being orthogonal to all other components in the new subspace[27].](image)

Deciding upon an adequate subspace is another important factor. It is important to preserve information, while also reducing the size of the feature space. This can be achieved by observing the variance in each principal component in regards to the variance of the original feature space. How much information is lost can then be derived from comparing the original variance to the variance found in the new subspace. This is known as explained variance. The first principal component accounts for the highest amount of variance. This is equivalent to say that the first component accounts for more variability among the samples than any subsequent component. The impact of each component can be visualized in a scree plot. In an arbitrary example in figure 3 we observe that some data can be projected onto a new subspace while
preserving roughly 70% of the variance. The new subspace obtained by the PCA has roughly 98% less dimensions than the original data (100 features instead of 4465)[23].

The Elbow method can be used to determine how many principle component are warranted. The method is mainly used for unsupervised clustering algorithms to decide upon a suitable number of clusters, but it is also applicable to PCA when choosing a suitable number of components. The concept is simple - observe the graph and find the bending point (the elbow). Hence the name[28]. In figure 3 an elbow point occur after approximately 12 principal components.

The elbow point does not always occur. Another methods is to summarize the variance of each principle component and compare it to the variance found in the original data. It is possible to stack principle components until the cumulative explained variance reach sufficient coverage (e.g. 60% as seen after approximately 63 principal components in figure 3). The threshold is an arbitrary value, often set to a number where the cumulative explained variance levels out[29].

3.2.2 Latent Dirichlet allocation

Latent Dirichlet allocation (LDA) is an unsupervised generative statistical model where documents can be explained in terms of automatically generated topics. The assumption is that all documents exhibit mixtures of a finite set of topics. Documents can then be explained in terms of the underlying (latent) topic distribution. LDA is subsequently prone to consider latent semantics of words[23].

Topics are constructed based on the likelihood of term co-occurrence. Consequently topics are not derived from semantics, nor epistemology. This statement might seem counterintuitive since LDA do indeed cover semantics as explained by Sebastian Raschka[23]. Topics are not constructed by means of semantics, although the result may explain semantics found in other documents. Furthermore, words attributed to a specific topic are not necessarily correlated to human reasoning even though co-occurring terms may display some resemblance to - what humans would describe as - cohesive topics[30].

The entire model can be explained using plate notation (figure 4). Both $\alpha$ and $\beta$ are hyperparameters that modifies the distributions of topic-word and document-topic characteristics. Documents are denoted by $M$ and words are denoted by $N$. The topic distribution $\theta$ for a given document $M$ can be obtained by looking at each individual topic $z$. The prominence each topic $z$ is further obtained by looking at the individual words $\omega$ appearing in the document. Words attribute to topics to varying extent. This is foremost controlled by the distributions. The different distributions (topics, topic associated word attribute probabilities, words in topics, and topic mixtures) is a problem of Bayesian inference. This is imperative as it enables the
model to create topics based on word correlations[31].

\[ \alpha \text{ - Dirichlet prior on the per-document topic distributions} \]

\[ \beta \text{ - Dirichlet prior on the per-topic word distribution} \]

\[ \theta_m \text{ - topic distribution for document } M \]

\[ \varphi_k \text{ - word distribution for topic } K \]

\[ z_{mn} \text{ - topic for the } N^{th} \text{ word in document } M \]

\[ \omega_{mn} \text{ - } N^{th} \text{ word in document } M \]

The generative process can be explained by repeating the steps for a corpus \( D \) consisting of \( M \) documents of length \( N \) (in terms of words).

1) Choose \( \theta \sim Dir(\alpha) \)
2) Choose \( \varphi \sim Dir(\beta) \)
3) For each \( \omega_{ij} \mid j \in \{1, \ldots, N_i\}, \text{ and } i \in \{1, \ldots, M\} \)
   a) Choose a topic \( z_{ij} \sim Multinomial(\theta_i) \)
   b) Choose a topic \( \omega_{ij} \sim Multinomial(\varphi_{z_{ij}}) \)[31]

Dirichlet prior on the per-document topic distributions (\( Dir(\alpha) \)) explains how many topics, and to what extent, each topic contribute to the summation of the document. The Dirichlet distribution is a generalization of the beta distribution into multiple dimensions. This is typically regulated by the hyperparameter \( \alpha \). Higher \( \alpha \) means that documents are explained by many topics, while a lower \( \alpha \) means that documents are explained by fewer topics. This can be observed in figure 5.

Dirichlet prior on the per-topic word distribution (\( Dir(\beta) \)) explains the relationship between words and topic. This relationship works equivalently to that of the effect of \( \alpha \) on the relationship between documents and topics. Higher \( \beta \) means that topics are explained by many words, while a lower \( \beta \) means that topics are explained by fewer words[31].

Topics can be generated from arbitrary data and then be used to evaluate other unrelated data. One such example is Wikipedia - a general, unstructured dataset that can be used to create topics using the LDA model[32]. These topics can then be used for topic modelling over magazine content (as illustrated by Andrius Knispelis)[33].
3.3 Classification models

3.3.1 Nearest neighbors algorithm

The k-nearest neighbors algorithm (KNN) is a supervised classification algorithm that traditionally requires little tuning. KNN requires no explicit training, instead all computation is deferred until the classification step. The k nearest neighbors take an affinity vote during classification to establish the label of new samples. Whatever label receives a majority decides the affinity of the sample[23].

Choosing the k nearest neighbors is decided upon by the Minkowski distance metric (equation 6). This is simply a generalization of the Euclidean and Manhattan distance. The Manhattan distance is used if $p=1$, while the Euclidean distance is used if $p=2$[34].

$$d(x^i, x^i) = \sqrt[p]{\sum |x^i_k x^j_k|^p}$$

(6)

It is also possible to modify the voting system by adding weights to each vote (eg. equation 7). Weights are assigned based on the distance between the sample that is being predicted and its k nearest neighbors[34].

$$\text{Weight} = \frac{1}{d(x^i, x^i)}$$

(7)

The traditional KNN implementation requires that all of the training data is memorized. This is the effect of needing predefined samples when classifying new samples. KNN is thus susceptible to both Hughes phenomenon and scales poorly with an expanding datasets[23].

3.3.2 Support vector machine

Support vector machines (SVMs) are supervised models for both classification and regression. SVMs are originally intended as a binary classification algorithm, but it can been extended to solve multiclass classification tasks by performing multiple binary divisions[35].

SVMs attempt to separate classes by placing a delimiter between hyperspaces. The optimization objective is to maximize the margin between the decision boundary (the delimiter) and the samples adjacent on either side of the delimiter. The margin is illustrated in figure 6.

![Fig. 6. An SVM with highlighted area between the support vectors. The decision boundary (hyperplane) is illustrated as a solid line $x^T \beta + \beta_0$.](image)

The decision boundary has the characteristic equation of $x^T \beta + \beta_0 = 0$. Optimizing a linearly separable hard-margin problem is achieved by maximizing $\beta$ (See equation 8)[36].

$$\min_{||\beta||} = ||\beta|| \text{ subject to } y_i(x_i^T \beta + \beta_0) \geq M,$$

(8)

Hard-margin is seldom the only important factor when tuning an SVM. Allowing for errors in favour of regularization has resulted in the introduction of another optimization variable, namely the slack variable ($\xi$). The slack variable allows for both errors and in the event of infeasible problems (e.g. linearly inseparable problems). This can be observed in figure 6 and in the soft-margin equation 9. There is a trade
off between the size of the margin and the accumulated error sum on the training data. This is regulated by the hyperparameter C. A small C opts for a larger margin but more mistakes, while a large C opts for fewer mistakes but a smaller margin[37].

\[ \min_{||\beta||} = ||\beta|| \text{ subject to } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \forall i \\ x_i \geq 0, \sum x_i \leq C \end{cases} \]

(9)

The optimization problem (equation 9) can be represented using the Lagrange dual function (equation 10). This representation relies solely on the input features via the dot product[36].

\[ L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i\alpha_i' y_i y_i' \langle h(x_i), h(x_{i'}) \rangle \]

(10)

The kernel function can be derived from the Lagrange equation (10) and represented as a stand alone function - commonly referred to as the kernel. The Linear kernel can be seen in equation 11.

\[ K(x, x') = \langle h(x_i), h(x_{i'}) \rangle \]

(11)

The SVM implementation used in this thesis is based on Libsvm. The complexity of this particular implementation is quadratic in regards to the training data. Further complexity is added when opting for more complex kernels (e.g. RBF)[39].

3.3.3 Multilayer perceptron

Artificial neural networks (ANNs) are supervised nonlinear statistical models. ANNs are typically represented as a combination of multiple, interconnected neurons. These neurons are generally divided into multiple layers (one input layer, N hidden layers, and one output layer). The internal workings of such a model depends entirely on what type of ANN is being deployed, and there are multiple subclasses of ANNs (e.g. Recurrent neural networks, Convolutional neural networks). The multilayer perceptron (MLP) is a type of ANN consisting of at least three layers. Furthermore, the MLP is a type of feedforward ANN. This means that connections between neurons (called synapses) do not form a cycle[36].

The first layer in an MLP differs from other layers since it receives the input signal directly from the input features. Aside from the first layer, all subsequent layers receive an input signal from the previous layer. All neurons in the current layer are connected to all neurons in the previous layer. This is known as a fully connected layer. There are alternatives to this (e.g. convolution layers), and extensions that may be applied to the model (e.g. the dropout method); but this is considered out of scope of this project and will not be evaluated[40].

In figure 7 we can interpret the input layer \( x_n \) as the signal being sent to one particular perceptron in the current layer. All signals are typically weighted individually \( W_{nj} \) before an described as the inverse of the support vectors influence[38].

\[ K(x, x') = \exp(-\gamma ||x - x'||^2) \mid \gamma > 0 \]

(12)
accumulated signal is received by any given neuron in the current layer. The neuron will receive an additional input $\theta_j$ known as the Bias before fed into the activation function[40].

The model represented in figure 7 can be found in equation 13.

$$O_j = \varphi_j(\theta_j + \sum_{j=0}^{n} w_{kj} x_j) \quad (13)$$

The activation function is denoted as $\varphi$. This function regulates the output signal for each perceptron. Some popular functions are the logistic function (equation 14), the Rectified linear unit (equation 15), or the Gaussian function (equation 16).

$$\varphi(x) = \frac{1}{1 + e^{-x}} \quad (14)$$

$$\varphi(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad (15)$$

$$\varphi(x) = e^{-x^2} \quad (16)$$

By placing multiple perceptron in each layer and stacking multiple layers on top of each other we get the complete model. The number of hidden layers, and the number of neurons in each layer, are hyperparameter that require some consideration. More layers, and more neurons, will increase the complexity of the model. Subsequently it will require more training, and more easily result in overfitting.

However, it is important to ensure that the model reach sufficient complexity in order to solve the problem[40].

The MLP classifier optimizes a cost function using an algorithm such as stochastic gradient descent (SGD). There are alternatives to SGD (e.g. Broyden–Fletcher–Goldfarb–Shanno algorithm) that works differently. However, this is not an in depth evaluation of different algorithms; other optimization methods are discussed and evaluated in details in Deep Learning[40] (Ian Goodfellow et al.) or by Sebastian Ruder[41]. See other sources for further reading.

SGD is a generalization of Gradient decent. In the original algorithm the entire training data has to be evaluated before any weight modification takes place. Gradient decent works by updating parameter $\theta$ with emphasis on the cost function $Q(\theta)$ (equation 17). The learning rate is a constant (or shrinking variable) denoted as $\alpha$. The expectation (E) in equation 17 is approximated by evaluating the cost and gradient for the entire training set. The notation is a simplification of multiple steps that can be studied in depth in Stanford’s ”Optimization: Stochastic Gradient Descent”[42].

$$\theta = \theta - \alpha \nabla_{\theta} E[Q(\theta)] \mid \alpha > 0 \quad (17)$$

In SGD it is only required to traverse a fraction (batch) of the training data before updating the weights (equation 18). The entire training dataset is split into smaller batches. It is commonly referred to as updating between batches rather than epochs[42].

$$\theta = \theta - \alpha \nabla_{\theta} Q(\theta; x^{(i)}, y^{(i)}) \mid \alpha > 0 \quad (18)$$

There are multiple ways of measuring the error. This is known as the cost function. One such function is the mean squared error (MSE). It measures how much the true target deviates from the prediction by squaring the discrepancy. The accumulated error for all training
input, and output is then the error factor (equation 19). Amount of training data is denoted as \( n \), prediction as \( Y \), and true target as \( \hat{Y} \).

\[
Q(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2
\]  
(19)

MSE is more commonly used as a regression cost function (although it can be used in classification). Other methods that are better opted for classification are the Hinge loss function (equation 20) or the Logistic loss function (equation 21)[43].

\[
Q(y, \hat{y}) = \max(0, 1 - y \cdot \hat{y})
\]  
(20)

\[
Q(y, \hat{y}) = \frac{1}{\ln 2} \ln(1 + e^{-y \cdot \hat{y}})
\]  
(21)

The entire process can now be summarized by following a number of steps, henceforth referred to as backpropagation.

1) Calculate the output of an arbitrary input.
2) Calculate the cost function by evaluating the discrepancy between target and prediction.
3) Generate the error term for each neuron by propagating the output value backwards. It is an inverse feedforward process, starting from the output node.
4) Update the weights based on the backpropagation procedure in step 3.

### 3.4 Hyperparameter tuning

The dataset (see chapter 4 - Data) is separated into two sets - the training set, and the testing set. Classification models use the training set to adjust the internal workings of the model to fit the data. The testing data is then used to evaluate the performance of the model. The two sets are traditionally divided into 80% training and a 20% testing. It is worth mentioning that different sources opt for different ratios, but the discrepancy is customary in the vicinity of this ratio[23].

To optimize the parameters for a given models we inadvertently create a bias towards the data we evaluate. This is why the trained model should never be evaluated on the test set before a final model has been decided upon. Thus we split the training set into something called a validation set. The validation set can be interpreted as the test set of the training data, to which we attempt to tune the parameters. Classification models are by definition biased towards the training set, and by intuition biased towards the validation set. However, the model is unbiased towards the test set. This is the primary reason for dividing a dataset into three distinct sets[36].

#### 3.4.1 K-fold cross-validation

K-fold cross-validation is a method that allows the model to be both trained and validated on the same set. In other words, combining the training- and validation set. The combined set is divided into \( k \)-folds as observed in figure 8. \( K-1 \) folds are used for training, while the last fold is used for evaluation. Figure 8 is a visual representation of the 10-fold-cross-validation technique. We observe that \( \frac{9}{10} \) folds are used for training the model, while the last \( \frac{1}{10} \) fold is used for evaluating the result. This process is repeated \( k \) times, in which time the testing fold shifts one position to the right. The validation score is the mean score of \( k \) iterations[36].

![Fig. 8. K-fold cross-validation. K equals ten as there are ten folds.](image)
3.4.2 Parameter search

The parameter selection is implemented using the Gridsearch feature in Scikit learn. It is a brute force algorithm that selects the best performing model from a set of predefined parameters. The parameters are combined in an exhaustive search by evaluating all possible variations of the model (in regards to the predefined parameters). The most successful variation is then selected as the final model. Success is based on the test score of the K-fold cross validation, which is controlled by an evaluation metric (chapter 3.5).

3.4.3 Sample weights

Classification datasets in supervised learning are equipped with input- and target values. Models such as the multilayer perceptron use this information during training by systematically updating the inner workings of the model. This means that larger classes are likely to impact the training more than smaller classes.

This can be mitigated by introducing weights to combat the biased class distribution. The default sample weight is one. By increasing the weight, the sample can be seen as more important, and it will thus impact the training more than that of other samples. Similarly, by decreasing the weight it becomes less important. Amplifying the per-instance loss is a popular way of implementing the weight factor in a model[44].

3.5 Evaluation

3.5.1 Classification accuracy

Classification accuracy is the fraction of correctly predicted labels split with the total number of predictions.

\[
\text{Accuracy} = \frac{C}{T}
\]

\(C = \text{Correct predictions}\)

\(T = \text{Total predictions}\) (22)

The classification accuracy metric is arguably better suited for uniformly distributed datasets since it is a biased representation of the underlying class distribution. The implication of this phenomenon is that bigger classes impact the accuracy more than smaller classes. This is especially true in classification tasks emphasising the performance of minority classes[45].

In a hypothetical test dataset with two classes in a 1:10 ratio, where the smaller class has a classification accuracy score of 0%, and the bigger class has a classification accuracy score of 100%, the total classification accuracy score would be 90%. This metric is misleading if we are interested in predictions regarding minority classes or mean prediction accuracy of multilabel samples[21].

3.5.2 Confusion matrix

A confusion matrix is a table layout that allows visualization of performance. It is a square matrix where each row and column corresponds to a unique class in a predefined dataset. There are four types of predictions when measured in a confusion matrix: True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). In a binary classification problem this would be represented as a 2-by-2 real matrix (figure 9). The matrix dimensions in a multiclass classification problem is \(M^2\) where \(M\) equals the number of classes represented in the dataset[21][23].
Fig. 9. Confusion matrix layout. The y-axis represent the true class values, while the x-axis represent the predicted values. [23]

TP are correct predictions. In a 100% classification accuracy multilabel classification problem, the confusion matrix only contain values in the diagonal, where all other cells are filled with zeros.

FP are false positives where samples are misslabelled. In a confusion matrix this would express itself as multiple non-zero cells in one particular row. All non-zero cells in that row are FP, except for the cell positioned in the $i$th column of the $i$th row (diagonally placed cells). That is a TP cell.

TN are correct rejections. These are correctly discarded samples of the $i$th class, where $i$ is the index of the currently considered class.

FN are samples that are not accounted for when a specific class is matched to a dataset. If 10 samples belong to specific class, and only 9 are correctly labelled, the remaining sample that is unaccounted for is an FN[23].

3.5.3 Recall and precision

One particular strength of the confusion matrix is that it visualizes recall and precision. Both precision and recall are derived from TP, FP, TN and FN. Precision is the number of correctly predicted samples out of all the retrieved samples (equation 23)[21][23].

Recall is the fraction of unaccounted relevant retrievals, split over the true number of samples that belongs to a specific class. The recall metric is paramount when trying to find all samples that belong to a particular class. In extremely skewed dataset (often found in anomaly detection tasks) the emphasis is often to find all anomalies rather than correctly classifying all retrieved samples. Obtaining a better overall accuracy score might be less important than retrieving all of the relevant samples. One of the goals of this thesis is to create a prototype that remove irrelevant material for further analysis. Consequently, in this case it is arguably more important to retrieve relevant samples than it is to obtain high accuracy[46].

$$Recall = REC = \frac{TP}{FN \cap TP} \tag{24}$$

3.5.4 $F_1$ measure

$F_1$ measure is a test accuracy metric based on precision and recall. It is especially useful when evaluating the performance of imbalanced class problems[21][23][46].

$F_1$ measure can thus be derived from precision and recall. This is the traditional $F_1$ measure. However, this is a binary metric and not applicable to a multiclass problem. Some extensions are provided to make the metric applicable, but there are limitations to the multiclass $F_1$ measure. When used in a multiclass environment the mean result of each class in computed[21].

$$F_1 \text{measure} = 2 \frac{PRE \times REC}{PRE + REC} \tag{25}$$

The $F_1$ macro measure is an extension that does not take label imbalance into account. It is the simple average across all classes, and all samples. The average performance of individual classes ($S$) are thus equally important[?].
\[ F_1 \text{ macro accuracy} = \frac{\sum S 2 \frac{P_s R_s}{P_s + R_s}}{S} \] (26)

It is possible to balance the metric by adding weights to the macro accuracy. This is known as the weighted F\(_1\) measure. Each class is weighted by the number of true instances for each label (\(\omega\)). This can be interpreted as something between F\(_1\) macro measure, and classification accuracy[46].

\[ F_1 \text{ weighted accuracy} = \frac{\sum S 2 \omega_s \frac{P_s R_s}{P_s + R_s}}{S} \] (27)

4 DATA

The primary dataset is derived from an independent project that aimed to map Darknet. The goal of that project was to map major marketplaces and forums by downloading mirrors of each and every Web page. The dataset contain 1.6 TB of unstructured data - 89 marketplaces, and 37 related forums. The dataset belongs to Gwern Branwen - https://www.gwern.net[47].

Wikipedia was used as a secondary dataset when forming topics using the Latent Dirichlet allocation model. The dataset contains 5,455,917 English Wikipedia articles and can be found at the Wikipedia website - https://dumps.wikimedia.org/enwiki/[48].

4.1 Data sampling

The dataset used for classification is based on a subset of Branwen’s Darknet dataset. Two marketplaces - Alphabay and Poseidon - where chosen based on the layout. The layout was paramount since product categories could be found on each web page in both Alphabay and Poseidon. Thus it was possible to accurately extract categories to obtain labels (classes) for the classification models. This is what made supervised learning an option.

Each unique web page found on the two domains are stored as individual files in date-and-domain separated folders. Uniqueness is assured by omitting overlapping files that appear in multiple crawls. Removing duplicates is important in order to both preserve class distributions, and not to create an involuntary bias towards specific classes. Duplicates can arguably be considered weighted samples since the effect is somewhat similar - samples that appear more than once, or receive weights, have greater impact during training. Furthermore, all files devoid of advertisement where removed from the dataset. The resulting documents - commonly referred to as listings - can be interpreted as web pages where arbitrary items are advertised.

Alphabay contains 2’024’819 files without preprocessing. The first constraint result in 520’403 files that are considered listings. These files can be found in a folder called listings. This is predefined in the dataset. The remaining files are named based on an ID (e.g. 500023) which can be used to remove duplicates. All listings have multiple tabs that all refer to the same page, but contains different information (Description, Bids, Feedback, Refund policy, etc.). Description is the only information that is directly connected to the advertisement. Removing all tabs but description result in 76’110 files. Some of these files appear in multiple crawls but under different IDs. Comparing files using a hashing algorithm (e.g. MD5) does not work since some time dependent values differ between crawls (e.g. timestamps, currency, stock). It is possible to circumvent this by having regular expressions removing numbers from the content. Using the MD5 algorithm to compare files without numbers result in 15’222 files. The same procedure result in 3’688 files when targeting the Poseidon dataset.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabay</td>
<td>15’222</td>
</tr>
<tr>
<td>Poseidon</td>
<td>3’688</td>
</tr>
</tbody>
</table>

The data is processed as a list of web pages without considering when the data was obtained. First we iterate through all sub-
directories in all crawls and concatenate the files to two lists - one for each domain. The web pages are stripped from scripts and styles, leaving only readable text that is presented to people browsing the web pages. This step is implemented using a Python library called Beautiful Soup.\(^1\) Scripts and styles can arguably be used as features but it would also introduce multiple static elements. The focus of this thesis is to retrieve and classify documents based on natural language. This is the second reason why scripts and styles are not considered valid features.

### 4.1.1 Labelling samples

Vendors specify a set of attributes before listing objects for sale. These attributes may contain information regarding price, origin and shipment, among others - see Fig10. We are primarily interested in the product category since that will serve as labels for the classifiers. The product category is chosen by the vendor from a fixed set of topics specified by the marketplace. This information is obtained using regular expressions, and made possible by the static layout.

The product specification is additional information provided by the vendor. This information is chosen by the vendor to market the product, and it is vital since it enables mapping between vendor specification and the product category. Thus it is possible to evaluate a classifier’s performance by comparing predictions (product categories predicted by the classifier) with the true values (product categories obtained by regular expressions).

The category systems for Alphabay and Poseidon are different. Alphabay has a hierarchical category system with 12 parent categories and 60 subcategories (Appendix - figure 32). Poseidon has a flat category system with 36 categories (Appendix - figure 33).

A mutual category system was used in order to evaluate listing distributions and prediction performance between the different markets. Furthermore, a mutual metric is needed in order to train and test classifiers on different markets. The product category is arbitrarily chosen by the vendor, and there is no obvious transformation between Alphabay’s- and Poseidon’s category system. I resorted to manual mapping, using the Alphabay category system as a baseline. The exact mapping can be found in the Appendix - see table10. The resulting category distribution can be seen in figure11. The content distribution align with prior studies made by Gareth Owen and Nick Savage. Both Alphabay and Poseidon are dominated by drug related material, followed by fraud, and then a variety of other material[1]. The biggest discrepancy is “other listings” in Poseidon. This is considered a consequence of the transformation process. It is also a potential threat to validity and should be kept in mind when observing the result.

1. https://www.crummy.com/software/BeautifulSoup/
Fig. 11. Product category distribution among Alphabay and Poseidon.
5 Method

The process can be explained by the different methods that together build the system (figure 12). It is a sequential process that starts by obtaining data in the feature selection phase. The data is processed and transformed before continuing to the dimensionality reduction phase. In this step the data undergo a second transformation. This is followed by the classification phase, where different classification models are used to classify the data based on content. The result of this step is then evaluated in the evaluation step.

To investigate the scalability of the prototype we separate two different Darknet markets and evaluate the result of training on data from one market, and testing on data from another market. This was decided because of the availability of reliable source material.

5.1 Feature selection

In this chapter we evaluate the effect of selecting features; either by introducing sampling constraints, opting for a predefined vocabulary, and/or looking at different sampling domains.

Data pre-processing and deciding upon suitable features are two important factors when constructing a model. Dimensionality reduction techniques project the original feature space onto a smaller feature space. Thus the result depends as much on the original features as the dimensionality reduction itself. This fact ought to be considered when observing the result.

The primary method to map tokens to vectors in the traditional bag-of-words model is by introducing sampling constraints. The constraints, along with some documents, essentially result in a vocabulary. Tokens that occur in both the vocabulary and the target document will cause a match in the representation (e.g. token frequency). This vocabulary can either be specified in advance, or automatically by adding constraints during the sampling process.

5.1.1 Feature sampling

Sampling constraints impacts dimensionality differently depending on both domain and method. In this chapter we observe the result of feature sampling on the target domain: Alphabay and Poseidon. This can be interpreted as looking at a data subset rather than the entire dataset. Alphabay and Poseidon are derived from a larger dataset (which makes it a subset).

Alphabay is featured in figure 13 showing the effect of different methods. The same methods are used on Poseidon in figure 14, but with different results due to the different domain. Neither domain appears affected by maxDF. This constraint causes words that appear in a fixed amount (or quota) of documents to be omitted from the bag-of-words. The default tokenization procedure is identical to that of removing words that occur in 50% of the documents for Alphabay, and 70% of the documents for Poseidon. This means that words recurring in the majority of the documents are extremely rare. Another similar observation is that stopword removal is not especially important. There are simply too many unique features, and too few stopwords. Only 22 tokens were removed from the Poseidon dataset, while 302 were removed from the Alphabay dataset, by introducing the stopwords constraint. Stopwords account for roughly 0.37% of the features in the Alphabay dataset, and roughly 0.94% of the Poseidon dataset.

The lemma constraint display an interesting trait. In the Alphabay dataset the resulting bag-of-words contains roughly 54% more features
than compared to the default tokenization process. The selected behaviour of the tokenization model (sciKit-learn’s Countvectorizer) is to ignore decoding errors when faced with an unrecognized encoding. The lemma constraint replaces this behaviour and manage to extract an incomprehensible string. This is why the resulting number of tokens are more than that of the default tokenization process. Furthermore, adding the lemma also makes the entire process much slower.

A discrepancy between Alphabay and Poseidon is the number of features in relation to the number of documents. The Alphabay document-to-feature ratio is 18.7%, while Poseidon document-to-feature ratio is 82.6%. Alphabay also exhibit linear feature growth in all tokenization requirements except for minDF; which is to be expected since only features that appear in more than 1% of all documents are added to the bag-of-words. However, the feature space in the Poseidon dataset appears to level out after approximately 750 documents. This indicates two things: there is a more uniformly distributed vocabulary in Poseidon, and/or there is more nonsense incorporated in the Alphabay vocabulary. Intuitively, this seem to concur with the most common words in each domain (figure 15). The top words in Alphabay are numbers, while the top words in Poseidon exhibit some domain specific information.

Evaluating the constraints is difficult. I used popular word frequency (figure 13 and 14), and box plots (figure 16) to evaluate the discrepancy between the two domains. Finding a feature sampling technique that result in similar outputs on both Alphabay and Poseidon is important to ensure that the classifiers are not

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Poseidon Tokens</th>
<th>Alphabay Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>4463</td>
<td>81187</td>
</tr>
<tr>
<td>Stop words</td>
<td>4241</td>
<td>80885</td>
</tr>
<tr>
<td>Lemma</td>
<td>4474</td>
<td>124768</td>
</tr>
<tr>
<td>(11 &gt; \text{word length} \geq 2)</td>
<td>4300</td>
<td>76233</td>
</tr>
<tr>
<td>Alphabet letters</td>
<td>307</td>
<td>950</td>
</tr>
<tr>
<td>(\text{MinDF}=0.01)</td>
<td>496</td>
<td>3484</td>
</tr>
<tr>
<td>(\text{MaxDF}=0.7)</td>
<td>4399</td>
<td>80813</td>
</tr>
</tbody>
</table>

TABLE 2

Vocabulary size associated with sampling constraints.
restricted to either domain alone. Obtaining a good generalization score require similar representation regardless of input. The box plots illustrate domain characteristics in regards to token lengths of different feature sampling techniques. The box plots shows token length in vocabularies associated with the constraints in table 2. Minimizing the amount of outliers (box plot circles) is done to avoid overfitting. The token length distribution in the Poseidon dataset is quite coherent with some outliers. Alphabay is extremely incoherent with multiple tokens that are more than 100 character long. These are somewhat mitigated by adding the MinDF- and MaxDF constraint.

The idea of limiting the vocabulary to Latin characters and of a certain length is based on the idea that words carry more information than symbols and numbers. Words that are shorter than three character, and longer than twelve characters, are omitted based on the intuition that most words of any importance probably resides within that length. Furthermore, including longer (or shorter) tokens will doubtlessly introduce more noise to the dataset. This argument is based on the fact that longer tokens (e.g. 500 characters long) occur less frequently than shorter tokens; which makes it less useful in terms of generalization. It should be noted that this is the opinion of the authors.

Table 3 illustrates the effect of combined constraints. The output of this served as the input for both the dimensionality reduction, and the classification process. The tokenization process is regulated by the following regular expression $u'[a-zA-Z]{3,12}$. The resulting tokens are Unicode strings, constructed by Latin characters of length 3 to 12. Alphabay- and Poseidon tokens are the amount of resulting tokens for each sampling constraint.

Linear feature growth is alarming since classifiers rely on previously observed data to find patterns for future predictions. However, previously unseen features can not be used since classifiers are not trained to utilize them.
5.1.2 Feature sampling: other sources

Instead of deriving the vocabulary from a subset (Alphabay and Poseidon), we can extract a vocabulary from other sources.

The entire dataset (which also contains both Alphabay and Poseidon) can be used to derive a vocabulary. The feature space is kept to a minimum by only sampling tokens from the Latin alphabet of length 3 to 12. Tokens that occur in less than five documents are also omitted along with stop words. This can be observed in table 4.

Wikipedia is another dataset that can be used to extract a vocabulary. Only the top 100,000 tokens are allowed into the bag-of-words due to the sheer size of the dataset. This can also be observed in table 4.

5.1.3 Feature representation

There are only two types of feature representations presented in this thesis: tf and tf-idf. Multiple variations are considered throughout the thesis, primarily by tuning the sampling constraints. One distinction to this is that tf-idf works differently than the default tf since the representation requires knowledge of all samples in order to perform the inverse document frequency. It can be interpreted as an extension of the tf representation, where the inverse document frequency is computed following the default term frequency representation. However, the inverse document frequency is not necessary computed based on the input data. It is possible to use any data as a baseline for the inverse document frequency computation. Granted - the common approach might be to use the input data as a baseline, but it should be noted that there are alternatives. All experiments in this thesis are performed using the input data to compute the inverse document frequency, unless other information is explicitly stated.

5.2 Dimensionality reduction impact on features

5.2.1 Latent Dirichlet allocation

The idea of LDA is to represent documents (websites in this case) as a mixture of topics, and that each word is attributable to one of those topics. Topics are created by iterating over a large number of documents in order to find word correlations. The source material used to construct the topics are not necessarily related to the input data - the Alphabay- and Poseidon dataset. In this thesis we consider two source materials: the entire Darknet dataset as presented by Gwern Branwen, and all articles found on the English Wikipedia.

There are a number of constraints that limits the usability of LDA under these circumstances. Firstly, the algorithm requires tuning to operate as a dimensionality reduction technique. The problem is that there is no obvious way of evaluating the result once the algorithm has been tuned. The only reliable benchmarking tool is to tune classifiers and evaluate the result of the actual classification process. Secondly, the tuning is very time consuming. Both Wikipedia and Darknet are huge collections of data.

I resorted to use default values for the LDA process in order to proceed without too much consideration on tuning the LDA. It is beyond the
scope of this thesis. Hyperparameters were set in accordance with the suggestions from Scikit learn and Gensim[49][32].

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topics</td>
<td>100</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>0.01</td>
</tr>
<tr>
<td>( \beta )</td>
<td>0.01</td>
</tr>
<tr>
<td>( \theta )</td>
<td>10</td>
</tr>
<tr>
<td>( \varphi )</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Any document can be fed into the LDA after it has been tuned. Documents are then represented as a mixture of 100 topics. This process can be thought of as a dimensionality reduction technique where we transform an arbitrary number of features into 100 features (topics). Ideally, the mixture should differ between samples associated with different categories, but correlate with samples associated with the same category as the sample in question. This can be observed in figure 17 and 18. All series represent the mean topic mixture of the specified category. A rolling mean is utilized to make the graphs coherent. The rolling window contains 15 values, which is why all of the series start at 15 rather than at zero. It is not as accurate as the true series, but it is easier to observe trends in the topic mixture.

![Graph](image1)

**Fig. 17.** The average mixture of topics for all samples associated with each category on Alphabay. The source material is either Darknet (top) or Wikipedia (bottom).

![Graph](image2)

**Fig. 18.** The average mixture of topics for all samples associated with each category on Poseidon. The source material is either Darknet (top) or Wikipedia (bottom).

All of the resulting mixtures, regardless of do-
<table>
<thead>
<tr>
<th>Domain</th>
<th>Domain</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabay</td>
<td>Darknet</td>
<td>0.0040</td>
</tr>
<tr>
<td>Alphabay</td>
<td>Wikipedia</td>
<td>0.0022</td>
</tr>
<tr>
<td>Poseidon</td>
<td>Darknet</td>
<td>0.0007</td>
</tr>
<tr>
<td>Poseidon</td>
<td>Wikipedia</td>
<td>0.0031</td>
</tr>
</tbody>
</table>

**TABLE 5**

Variance of the topic mixture series in figure 17 and 18.

...main and source material, appear correlated. The visual representations found in figure 17 and 18 show no obvious distinguishing between any categories except for saturation in individual topics; the overall trends are similar. The variance between the series can be found in table 5. Higher variance might indicate more separation between categories. It is a crude metric that remains inconclusive, but without variance there can be no separation between categories.

### 5.2.2 Principle component analysis

The features we obtained in the previous chapter serve as input to the PCA. The result of the PCA can be observed in figure 19 and figure 20. The impact of each component is visualized in a scree plot, featuring cumulative explained variance.

Using the elbow method we settle for 50 components. This include the elbow of all variations of either dataset. The plots in figure 19 and 20 are somewhat misleading since they feature cumulative explained variance rather than normalized explained variance. The representation is better suited for finding a threshold for total explained variance rather than finding an elbow point. We settle for this representation since we are doing both the elbow method and the total explained variance method.

The total explained variance method for term frequency representation (figure 19) can be set at 200 components for Poseidon since nearly 100% of the explained variance is obtained at that point. Alphabay is more difficult as it levels out at around 90% explained variance. Thus 200 components where selected for Alphabay as well with the argument that further components will not improve the model significantly.

Tf-idf representations (figure 20) are not linearly correlated to the same extent as that of term frequency representations. This is not too prominent in the Poseidon dataset and roughly 200 components reach 100% explained variance. However, looking at Alphabay we find that cumulative explained variance grows almost linearly as opposed to logarithmically as in figure 19. This makes it difficult to decide upon a suitable subspace.
5.3 Tuning classifiers

The dataset is split into a training set (80%) and a testing set (20%). The training set utilizes k-fold cross validation. Thus removing the need for an explicitly selected validation set. Five folds ($k=5$) where decided upon as suggested as a rule of thumb by Trevor Hastie et. al in The Elements of Statistical Learning[36].

Alphabay and Poseidon are treated separately as two distinct datasets. These two datasets have four different vocabularies, resulting in four unique feature representations each. Furthermore, each feature representation is subject to the dimensionality reduction technique described in the previous chapter. This means that there are an additional four variations of each dataset for every dimensionality selected by the PCA.

Resampling techniques such as over- and undersampling are ways of counteracting data imbalance. Smaller classes are traditionally oversampled in order to be more representative. Larger classes can likewise be undersampled. This means that some samples are removed from the dataset. I have decided against using resampling since it would impact the already sparse input space.

This thesis considers three different evaluation methods. Choosing an appropriate evaluation metric had great impact on the result. Optimizing to maximize classification accuracy was seldom aligned with the accuracy of individual classes. The dataset is extremely imbalanced, and in some cases, simply targeting the biggest class proved more accurate (in terms of classification accuracy) than attempting to predict the actual class. By deploying a naive static classifier that only predicts a single class - the largest class - we can see the outcome of such behaviour. The result of this can be observed in figure 21. By feeding the test data to the model, we get the following result: classification accuracy (45%), $F_1$ macro accuracy (5%), and $F_1$ weighted accuracy (30%). It is not difficult to imagine that a solution is likely to be biased if the final model is solely optimizing the classification accuracy.

5.3.1 Evaluation metric
Fig. 21. The confusion matrix of a naive static classifier that always predicts the largest class. The classification accuracy is 45%, the F\textsubscript{1} macro accuracy is 5%, and the F\textsubscript{1} weighted accuracy is 30%.

The F\textsubscript{1} weighted score was chosen as the primary evaluation metric in order to avoid any bias towards either recall or precision. All models where thus tuned to maximize the F\textsubscript{1} accuracy score. This decision was based on a desire to get a balanced result and avoid the pitfalls of narrow metrics (e.g. classification accuracy). Some early models proved better than the final models in regards to either classification accuracy or F\textsubscript{1} macro score - but never both at the same time.

The F\textsubscript{1} macro score calculate metrics for each label, and find their unweighted mean. This can be used to identify bias. It was difficult to obtain a high F\textsubscript{1} macro score due to the characteristics of the Alphabay dataset; it is too imbalanced. Tuning the models in accord with the F\textsubscript{1} macro metric led to diverse predictions with emphasis on recall for smaller classes. Consequently, it also led to poor performance in term of precision for smaller classes, and less recall for larger classes.

The classification accuracy metric is a less than optimal choice when applied to an imbalanced, multiclass dataset (as observed in figure 21). The reverse effect can be stated regarding the F\textsubscript{1} macro score. However, the classification metric can be used in combination with the F\textsubscript{1} macro score. If we consider the F\textsubscript{1} macro score the lower bound, and the classification accuracy the upper bound, we can try to balance the two scores. Increasing the score of both metrics is ideally the best outcome, but a trade-off is likely to occur when the accuracy score levels out.

One way to view the evaluation step is that the F\textsubscript{1} score directs the tuning phase while the confusion matrix evaluate the result of the final model. The F\textsubscript{1} score is important in order to sort through multiple models in order to find the final version. Once that model is obtained it can be evaluated by feeding the predictions through a confusion matrix. This might give valuable insight to precision and recall for individual classes, and also identify any bias.

5.3.2 Nearest neighbors algorithm

All variations of the input data (domain, feature representations, dimensionality reductions) where subject to a parameter sweep of 1-50 neighbours. One weighting method was used aside from the standard uniform method (equation 7). Both the Manhattan distance and the euclidean distance where used.

**Distance metric**: The exhaustive parameter search generated similar result across both domains and features. The inverse distance weighting system proved more successful than the default uniform weighting system. The weighted metric proved more efficient in all parameter sweeps, regardless of either domain or feature representation.

**Distance calculation**: The distance calculation metric was equally dominated by the Manhattan distance. All parameter sweeps featured the Manhattan distance as the most successful distance measurement metric.

**Neighbours**: The most efficient number of neighbors varied between 1 and 14. The Al-
phabay dataset exhibited less variance in terms of preferred number of neighbors (between 6 and 12) while the Poseidon dataset was less coherent and sometimes favoured 1 neighbour. This is quite alarming since "it uses only the training point closest to the query point, the bias of the 1-nearest neighbour estimate is often low, but the variance is high"[36]. The conclusion of this - as stated by Trevor Hastie et. al in The Elements of Statistical Learning - is that "the presence of bias indicates something basically wrong with the model, whereas variance is also bad, but a model with high variance could at least predict well on average"[36]. Recurrent tests showed that 6 neighbours had balanced performance regardless of domain and metric.

<table>
<thead>
<tr>
<th>Poseidon Feature representation</th>
<th>Test score (Val. score)</th>
<th>Test score (Val. score)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>(0.96) 0.97</td>
<td>(0.96) 0.97</td>
</tr>
<tr>
<td>u'[a-zA-Z][3,12]</td>
<td>(0.96) 0.97</td>
<td>(0.96) 0.97</td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.96) 0.96</td>
<td>(0.96) 0.96</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.96) 0.96</td>
<td>(0.95) 0.96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alphabay Feature representation</th>
<th>Test score (Val. score)</th>
<th>Test score (Val. score)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>(0.45) 0.49</td>
<td>(0.52) 0.60</td>
</tr>
<tr>
<td>u'[a-zA-Z][3,12]</td>
<td>(0.58) 0.59</td>
<td>(0.66) 0.66</td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.55) 0.55</td>
<td>(0.49) 0.49</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.55) 0.55</td>
<td>(0.49) 0.49</td>
</tr>
</tbody>
</table>

| PCA => 200 |
|-----------------|-----------------|
| Poseidon Feature representation | Test score (Val. score) Tf | Test score (Val. score) Tf-idf |
| Default | (0.96) 0.97 | (0.96) 0.97 |
| u'[a-zA-Z][3,12] | (0.96) 0.97 | (0.96) 0.96 |
| MinDF=5 | | |
| maxDF=0.75 | | |
| Stop words | | |
| Entire dataset | (0.96) 0.97 | (0.96) 0.96 |
| Wikipedia | (0.96) 0.97 | (0.96) 0.97 |

| PCA => 200 |
|-----------------|-----------------|
| Alphabay Feature representation | Test score (Val. score) Tf | Test score (Val. score) Tf-idf |
| Default | (0.47) 0.52 | (0.56) 0.64 |
| u'[a-zA-Z][3,12] | (0.66) 0.72 | (0.71) 0.76 |
| MinDF=5 | | |
| maxDF=0.75 | | |
| Stop words | | |
| Entire dataset | (0.60) 0.65 | (0.64) 0.66 |
| Wikipedia | (0.59) 0.64 | (0.60) 0.66 |

| PCA => 50 |
|-----------------|-----------------|
| Poseidon Feature representation | Test score (Val. score) Tf | Test score (Val. score) Tf-idf |
| Default | (0.97) 0.97 | (0.97) 0.98 |
| u'[a-zA-Z][3,12] | (0.96) 0.97 | (0.97) 0.96 |
| MinDF=5 | | |
| maxDF=0.75 | | |
| Stop words | | |
| Entire dataset | (0.96) 0.97 | (0.96) 0.96 |
| Wikipedia | (0.96) 0.97 | (0.96) 0.97 |

| PCA => 50 |
|-----------------|-----------------|
| Alphabay Feature representation | Test score (Val. score) Tf | Test score (Val. score) Tf-idf |
| Default | (0.48) 0.60 | (0.51) 0.62 |
| u'[a-zA-Z][3,12] | (0.63) 0.70 | (0.68) 0.75 |
| MinDF=5 | | |
| maxDF=0.75 | | |
| Stop words | | |
| Entire dataset | (0.61) 0.69 | (0.66) 0.75 |
| Wikipedia | (0.59) 0.69 | (0.67) 0.74 |
Around ten dimensions are required to reach a point where improvements appear to stagnate. This can be observed in figure 22. The graph features the effect of neighbors in terms of the weighted F₁ score. There is clearly a relation between explained variance and classification accuracy. The elbow point (50 dimensions) and the total explained variance-method (200 dimensions) reach roughly 70% accuracy. More importantly, higher dimensionality do not result in higher accuracy score. This is visible in the graph where 400 dimensions are - on average - slightly worse than

5.3.3 Support vector machine

C: The default linear kernel associated with SVMs is strictly regulated by the soft margin constant (C). The hyperparameters used during the exhaustive search can be found in equation 28. The best result was found when C=1, while further fine tuning the around 1 led to the overall result in C=0.25.

$$C = \{2^x \mid x \in [-15, 15] \mid x \in \mathbb{Z}\}$$  \hspace{1cm} (28)

Weights: The best validation score was found when samples where balanced as suggested in equation 29. Samples are weighted inversely proportional to class size (i).

$$Weight_i = \frac{S}{C \times S_i}$$

$$S = \text{Number of samples}$$

$$C = \text{Number of classes}$$  \hspace{1cm} (29)

The RBF kernel is subject to both the C and γ hyperparameter. Optimizing SVMs with RBF kernels is typically done by pairwise hyperparameter tuning. Tampering with C will impact the effect of γ, and vice versa. The different values of γ can be found in equation 30, and the values of C can be found in equation 28. The same values of C where used for both kernels.
\[ \gamma = \{ \, 3^x \mid x \in [-6, 6] \mid x \in \mathbb{Z} \, \} \]  

Tuning an additional hyperparameter, along with the fact that the RBF kernel is more complex than the linear kernel, makes both the tuning and the training exponentially more time consuming. 2,160 different variations were evaluated for the RBF kernel; 30 \( C \) values, 12 \( \gamma \) values, and 6 weighting methods. The linear kernel had 180 different combinations of hyperparameters; 30 \( C \) values, and 6 weighting methods. Both of these evaluations where repeated 48 times each: once for every domain, dimensionality, and feature set. The linear kernel still proved superior in all cases.

The best solution was had the linear kernel, \( C=0.25 \), and weights according to equation 29. This produced the result presented in the tables in this section.

<table>
<thead>
<tr>
<th>Poseidon Feature representation</th>
<th>PCA ( \rightarrow ) 200 Test score (Val. score)</th>
<th>Tf</th>
<th>Test score (Val. score) Tf-idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>(0.96)0.96</td>
<td>(0.96)0.96</td>
<td></td>
</tr>
<tr>
<td>( \text{u'[a-zA-Z]{3,12} )</td>
<td>(0.96)0.96</td>
<td>(0.96)0.96</td>
<td></td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.96)0.94</td>
<td>(0.95)0.95</td>
<td></td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.94)0.94</td>
<td>(0.95)0.95</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alphabay Feature representation</th>
<th>PCA ( \rightarrow ) 200 Test score (Val. score)</th>
<th>Tf</th>
<th>Test score (Val. score) Tf-idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>(0.75)0.75</td>
<td>(0.75)0.75</td>
<td></td>
</tr>
<tr>
<td>( \text{u'[a-zA-Z]{3,12} )</td>
<td>(0.74)0.74</td>
<td>(0.77)0.79</td>
<td></td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.71)0.73</td>
<td>(0.76)0.77</td>
<td></td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.70)0.69</td>
<td>(0.76)0.76</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Poseidon Feature representation</th>
<th>PCA ( \rightarrow ) 50 Test score (Val. score)</th>
<th>Tf</th>
<th>Test score (Val. score) Tf-idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>(0.96)0.97</td>
<td>(0.95)0.95</td>
<td></td>
</tr>
<tr>
<td>( \text{u'[a-zA-Z]{3,12} )</td>
<td>(0.96)0.96</td>
<td>(0.96)0.96</td>
<td></td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.96)0.96</td>
<td>(0.95)0.95</td>
<td></td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.96)0.96</td>
<td>(0.95)0.95</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alphabay Feature representation</th>
<th>PCA ( \rightarrow ) 50 Test score (Val. score)</th>
<th>Tf</th>
<th>Test score (Val. score) Tf-idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>(0.75)0.75</td>
<td>(0.75)0.75</td>
<td></td>
</tr>
<tr>
<td>( \text{u'[a-zA-Z]{3,12} )</td>
<td>(0.74)0.74</td>
<td>(0.77)0.79</td>
<td></td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.71)0.73</td>
<td>(0.76)0.77</td>
<td></td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.70)0.69</td>
<td>(0.76)0.76</td>
<td></td>
</tr>
<tr>
<td>Poseidon Feature representation</td>
<td>LDA Test score (Val. score)</td>
<td>Test score (Val. score)</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------</td>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td>(0.60)0.61</td>
<td>(0.87)0.88</td>
<td></td>
</tr>
<tr>
<td>u’[a-zA-Z]{3,12}</td>
<td>(0.59)0.60</td>
<td>(0.86)0.87</td>
<td></td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.60)0.60</td>
<td>(0.87)0.87</td>
<td></td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.60)0.60</td>
<td>(0.87)0.87</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alphabay Feature representation</th>
<th>LDA Test score (Val. score)</th>
<th>Test score (Val. score)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>(0.40)0.45</td>
<td>(0.54)0.58</td>
</tr>
<tr>
<td>u’[a-zA-Z]{3,12}</td>
<td>(0.38)0.40</td>
<td>(0.52)0.52</td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.39)0.41</td>
<td>(0.53)0.52</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.39)0.42</td>
<td>(0.52)0.52</td>
</tr>
</tbody>
</table>

### 5.3.4 Multilayer perceptron

There is a vast collection of different hyperparameters in MLPs. The following chapter is a summary of the parameter grid and the result. Some decisions were made to limit the scope of the exhaustive search. Learning rate was kept at a constant 0.001 with momentum at 0.9. These are all default values in the Scikit learn MLP library. Weights were initialized uniformly.

**Hidden layers:** The input layer is controlled by the dimensionality of the input space. The number of hidden layers (one or two layers) was decided upon to limit the scope and complexity of the model. It was also kept small in order to avoid further overfitting caused by the curse of dimensionality. The first iteration evaluated 100, 150, 200, 250, and 300 neurons in the first hidden layer, and the second iteration added an additional layer of 50, 100, and 150 neurons to the first hidden layer. Twenty different layer structures were evaluated: 5 containing one hidden layer, and 15 containing two hidden layers.

**Activation functions:** Two activation functions were considered: the rectifier function and the logistic function. Activation functions are deployed layer wise in this experiment. Different layers might have different activation functions, but all neurons in one given layer have the same activation function.

**Optimization methods:** Two optimizers were considered. Default stochastic gradient descent was the first, while Adams optimizer was the second. It is worth mentioning that tuning the Adam optimizer itself is warranted, but kept to a minimum to avoid unnecessary complexity.

**Loss functions:** Two loss functions were considered: mean squared error, and categorical cross entropy.

**Learning method:** Two learning methods were used: online training and batch oriented training. Online training is traditionally a quick method, but training time was not really an issue since most models managed to converge quickly given the sparse dataset. Batch training means that a set of predictions are done before updating the weights. The entire batch is then evaluated in order to updated the weights accordingly. I variety of batches were tested: sets of 5, 25, 50, 100 and the entire epoch.

<table>
<thead>
<tr>
<th>Poseidon Feature representation</th>
<th>Test score (Val. score) Tf</th>
<th>Test score (Val. score) Tf-idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>(0.96)0.97</td>
<td>(0.96)0.97</td>
</tr>
<tr>
<td>u’[a-zA-Z]{3,12}</td>
<td>**(0.97)**0.96</td>
<td>**(0.96)**0.97</td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.96)0.96</td>
<td>(0.96)0.97</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.96)0.96</td>
<td>(0.96)0.97</td>
</tr>
<tr>
<td>Alphabay Feature representation</td>
<td>Test score (Val. score)</td>
<td>Test score (Val. score)</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>Tf</td>
<td>Tf-idf</td>
</tr>
<tr>
<td>Default</td>
<td>(0.93)0.78</td>
<td>(0.95)0.81</td>
</tr>
<tr>
<td>u’a-zA-Z]{3,12}</td>
<td>(0.96)0.80</td>
<td>(0.97)0.83</td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.96)0.79</td>
<td>(0.96)0.78</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.93)0.75</td>
<td>(0.95)0.76</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Poseidon Feature representation</th>
<th>PCA -&gt; 200 Test score (Val. score)</th>
<th>Test score (Val. score)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tf</td>
<td>Tf-idf</td>
</tr>
<tr>
<td>Default</td>
<td>(0.95)0.97</td>
<td>(0.93)0.96</td>
</tr>
<tr>
<td>u’a-zA-Z]{3,12}</td>
<td>(0.95)0.96</td>
<td>(0.91)0.95</td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.90)0.96</td>
<td>(0.91)0.95</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.86)0.96</td>
<td>(0.90)0.95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alphabay Feature representation</th>
<th>PCA -&gt; 200 Test score (Val. score)</th>
<th>Test score (Val. score)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tf</td>
<td>Tf-idf</td>
</tr>
<tr>
<td>Default</td>
<td>(0.85)0.87</td>
<td>(0.82)0.80</td>
</tr>
<tr>
<td>u’a-zA-Z]{3,12}</td>
<td>(0.88)0.81</td>
<td>(0.89)0.82</td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.88)0.75</td>
<td>(0.82)0.78</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.82)0.74</td>
<td>(0.81)0.77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Poseidon Feature representation</th>
<th>LDA Test score (Val. score)</th>
<th>Test score (Val. score)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wikipedia</td>
<td>Darknet</td>
</tr>
<tr>
<td>Default</td>
<td>(0.60)0.61</td>
<td>(0.91)0.92</td>
</tr>
<tr>
<td>u’a-zA-Z]{3,12}</td>
<td>(0.59)0.60</td>
<td>(0.90)0.91</td>
</tr>
<tr>
<td>MinDF=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxDF=0.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entire dataset</td>
<td>(0.59)0.60</td>
<td>(0.90)0.91</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>(0.59)0.60</td>
<td>(0.90)0.91</td>
</tr>
</tbody>
</table>

The best performance was achieved by an MLP with two hidden layers. The first hidden layer had 250 neurons and a rectified activation function. The second hidden layer had 50 neurons and a rectified activation function. The output layer had 12 neurons and featured the logistic function. The output is a vector of continuous values between 0 and 1. To obtain a class prediction from this I looked at the cell with the highest value and set the rest of the values to 0. The Adam optimizer achieved better score than stochastic gradient descent. The loss function had little impact and both mean squared error and categorical class entropy had the approx-
imately same result. Categorical class entropy was used in the final model on the basis that weight changes does not get smaller as it levels out[18]. The best result was achieved by using full batch learning, where one batch equals an entire epoch.

6 RESULT

6.1 Feature representation

Tf-idf proved slightly better than tf on average. The discrepancy is small, albeit significant. It is difficult to derive too much meaning from the Poseidon dataset since the result remain static around 90% - 95% regardless of modifications. The Alphabay dataset respond better to minor tuning.

Sampling constraints is another important subject. More features does not always equal better performance. Opting for Latin characters between 3 and 12 characters proved far superior to any other explored constraint. Further feature engineering could potentially yield better results.

6.2 Dimensionality reduction

Utilizing features fed through dimensionality reduction algorithms did not surpass the performance of the original input space. KNN was the only model that managed to obtain a higher score when using a PCA transformed subspace. However, KNN could not compete with either SVM or MLP in terms of either precision or recall.

6.2.1 PCA

The upside of using PCA is that it is possible to trade accuracy for complexity and training time. All of the models (KNN, SVM, and MLP) executed much faster when operating on a smaller subspace. The explained variance graph is a powerful tool to use when deciding upon a suitable subspace. Explained variance appear to coincide with the performance of the model. As long as the explained variance remain high enough, there is but a slightly penalized result compared to the original input. The result tend to decrease once explained variance drops off. It is not difficult to make a favourable trade-off if decreasing the training time is valued more than classification accuracy. Hyperparameter tuning can be sped up significantly under these circumstances.

6.2.2 LDA

LDA generated data resulted in the worst performance regardless of metric and domain. It is also the most complex model and difficult to evaluate since it require more tuning than PCA. Stacking classification models on top of the LDA made the evaluation almost impossible. Some meaningful topics where constructed by the LDAs operating on both Wikipedia and Darknet, but evidently without effect.

There was no obvious correlation between performance and variance (figure 17 and 18) as theorized earlier. LDAs trained on the Darknet data performed slightly better than LDAs trained on the Wikipedia data.

6.3 Data imbalance

Sample weights resulted in slightly better performance for all classifiers, on all domains, given any features. Adding weights always led to a trade-off between classification accuracy and recall. Optimizing based on F\textsubscript{1} weighted score meant that smaller classes got better recall and more TP, but not necessarily better precision.

6.4 Classification result

The best performing models are presented in the following chapter. The models where picked based on the validation score obtained in the previous chapter.
The performance of the models operating on the Poseidon dataset have nearly flawless performance regardless of classification model, features, hyperparameter tuning, or dimensionality reduction (table 7 and figure 23). The only apparent flaws are centred around ambiguous topics: digital products, other listings, and guides and tutorials. All of these categories are difficult to separate based on content alone. Consider an E-book called "How to earn money in the stock market"; should it be categorized as a digital product or as guides and tutorials? It is quite likely that some people would disagree in the categorization of certain listings. Subsequently, it is unlikely that any algorithm can reach a perfect score.

The following listing appear on Poseidon.

| Book Name: How to hide Anything |
| By: Michael Connor |
| Press: Paladin Press |
| Pages: 111 |
| Language: English |
| Format: PDF |

"With little effort and expense, you can hide cash, armaments and even family from the menacing eyes of burglars, terrorists or anyone. Learn how to construct dozens of hiding places right in your house and yard. Here are small hiding places for concealing money and jewelry and large places for securing survival supplies or persons. More than 100 drawings show how to turn ordinary items into extraordinary hiding places."

The MLP featured in figure 25 classify the listing as "Digital products" while the actual category is "Guides and tutorials". The model confidence is 77% in favour of Digital products. Guides and tutorials is the second highest category prediction with 22% confidence. The merchant is clearly advertising an E-book, which is digital by definition. However, the content of the E-book is apparently a guide on "how to hide anything". This can arguably justify categorizing the listing as guides and tutorials.

The Alphabay dataset is less coherent if we are to believe the result of the classification. The average $F_1$ weighted score is 81%, and the best score is 83%. Observing miss-classified samples (MLP) reveal that the vast majority appear to suffer from ambiguity. These samples constitute 29% of the entire test dataset, and 92% of these are correctly classified by the second highest predictive confidence. This leads me to conclude that there is nothing fundamentally wrong with the performance, by rather with the way the categories are issued. It is the same problem as with the Poseidon dataset, but more prominent.

The following listing appear on Alphabay.

"Proven Secrets to Gain The Extra Edge and Make Money Fast to Generate $20 to $100 In A Matter of Minutes. Then The Best Part is All It Can Be Done Online and Offline and This Can Be Scaled As Much As You Want and You Will Have The Power to Run The Whole Show. Here You Will Have Bit Coin Foundation Fast Start with The Secrets of How BitCoin Works So You Can Gain The Extra Edge and Make Money Fast. Then You Will Also Learn How to Find Out How A BitCoin is Made And What You Need to In Order To Get a Ton Of Them Easily and Quickly. You Will Also Learn Why The $20 Bill in Your Pocket Right Now Is Worthless and What You Should Be Focused On Instead."

The MLP featured in figure 28 classify the listing as "Guides and tutorials" (55%) while the actual category is "Counterfeit items" (30%). The listing contains information on how to earn money using Bitcoin. The text itself express no information other than the fact that money can be made. There is no mentioning of counterfeit or falsification. This does not rule out the possibility of counterfeiting being part of the solution, but that is simply guesswork. Instead I would argue that this listing is categorized inaccurately based on the information at hand.
The Alphabay dataset is apparently more com-
plex than the Poseidon dataset, and ambiguity is arguably an even bigger issue compared to Poseidon. This can be observed in both SVM and KNN as they exhibit similar traits (table 26 and figure 27). Digital product and guides and tutorials are mixed up in almost all categories, resulting in many FP. Fraud is also accountable for a large amount of FP. A visual inspection of both models show that an total of four categories are difficult to separate from other categories (digital products, drugs and chemicals, fraud, and guides and tutorials). Drugs and chemicals deviate from the other categories as it can be considered more of a stand alone category than the other three. Drugs and chemicals contain material resolving around the topic itself, while the other three categories are topics that require some additional topic. As an example - guides and tutorials can hardly be “guides of weapons”. It is more likely “guides of weapons”, which means that the content would also revolve around weapons. This is why ambiguity is expected in some categories, but not others. Consequently we ask ourself “why all the FP?” in drugs and chemicals. The answer to that question is that there is a bias towards the biggest class in the dataset (drugs and chemicals). Adding weights and tuning hyperparameters to optimize recall is apparently not enough to eliminate the residue of skewed classes.

The most balanced model is doubtlessly the MLP. There is no evident bias towards either category, and both recall and precision appear uniformly distributed among all categories. The $F_1$ macro score is 0.65 compared to the KNNs 0.52, and the SVMs 0.57. This is quite impressive when considering that the MLP has the same $F_1$ weighted score as the SVM (0.83) and the overall accuracy is only 0.01 less than that of the SVM. This means that the recall score is much higher compared to either KNN or SVM, with no apparent trade off in accuracy.
6.5 Cross domain classification

Cross domain classification can be thought of as an additional test set. The idea behind the cross domain classification procedure is to evaluate each model's ability to operate outside the original dataset. Successfully deploying a model that where trained on one dataset, and then tested on another dataset, is a way of measuring how well a model performs in a more generalized environment. To test the effect of this I deployed the best models from either domain on the opposite dataset (models trained on the Alphabay dataset where tested on the Poseidon dataset, and vice versa). The models in question are those associated with table 6 and 7. The result of this test can be found in figure 29.

The best cross domain performance model was the MLP trained on Alphabay. The associated confusion matrix can be found in figure 30. The test score for that model when operating on the Poseidon dataset was: $F_1$ weighted score 49%, classification accuracy: 51%, and $F_1$ macro score: 31%. It is also worth mentioning that the MLP performed better on all metrics compared to both KNN and SVM. To put these numbers into perspective we can observe the naive static classifier (figure 21). The naive static classifier score was: $F_1$ weighted score 30%, classification accuracy: 45%, and $F_1$ macro score: 5%.
Alphabay to Poseidon & Test score & F<sub>1</sub> weighted & F<sub>1</sub> macro & Accuracy \\
KNN & 0.41 & 0.13 & 0.43 \\
SVM & 0.47 & 0.29 & 0.51 \\
MLP & 0.49 & 0.31 & 0.51 \\

TABLE 8 Cross domain testing scores associated with best performing models.

Poseidon to Alphabay & Test score & F<sub>1</sub> weighted & F<sub>1</sub> macro & Accuracy \\
KNN & 0.41 & 0.12 & 0.41 \\
SVM & 0.48 & 0.13 & 0.50 \\
MLP & 0.47 & 0.12 & 0.53 \\

TABLE 9 Cross domain testing scores associated with best performing models.

The result is rather underwhelming when looking at the different scores; F<sub>1</sub> weighted-, F<sub>1</sub> macro-, and accuracy score are essentially halved. However, the confusion matrix show that the model still manage to identify the majority of the categories from the other domain. Some classes are apparently less susceptible to ambiguity, which might explain why they appear unaffected by the crossover (e.g. drugs and chemicals, and weapons).

It is possible to modify the model to account for ambiguity by allowing for more than one prediction for every sample. This would give a better understanding of ambiguity as it implies that although the model did not predict the correct category, it was stuck in an conflict of interests, and could not decide between multiple candidates. Subsequently the second (or potentially third) predicted category in terms of confidence score would be the correct prediction. The idea of allowing for more than one prediction per sample might come across as cheating. However, in certain scenarios where recall is more important than accuracy, the concept of having multiple predictions per sample has benefits (e.g. it might be more important to find all anomalies rather than correctly classifying each candidate). To illustrate the effect of this I gave the algorithms three attempts to get the classification correct (figure 31). The test score for the same model as in figure 30 with additional predictions was: F<sub>1</sub> weighted score 75%, classification accuracy: 86%, and F<sub>1</sub> macro score: 45%. Note that this score only considers the correct prediction out of three possible attempts. That means that there are multiple ways to interpret the score. The metric featured in this example does not consider false predictions leading up to a correct prediction to be classified as errors if the correct prediction was found amongst the top three candidates. The point is that better coverage can be achieved by allowing for this behaviour. The confusion matrix show that three categories where completely ignored even with additional predictions, whilst the remaining eight categories achieved nearly full coverage.

![MLP Alphabay](image-url)
The idea is to provide a filter by classifying material in order to minimize manual labour. Classification accuracy is subsequently not as important as recall. Classification accuracy, or precision to be specific, is important to keep recall to a minimum. However, retrieving a suitable amount of relevant samples is accomplished by optimizing for precision and recall alike. This is the reasoning behind opting for the $F_1$ weighted score rather than recall or precision alone. Opting for a good metric, or multiple good candidates, is probably the most important decision when tuning models associated with imbalanced data. It is also important to note that neither metric is better than the other. Different metrics are good at different things. One such example is that the best performing SVM surpassed the MLP in some cases, both in terms of the $F_1$ score, and in terms of classification accuracy. This is when the confusion matrix proved invaluable as it became clear that the SVM had a greater bias towards larger classes, whilst the MLP was more balanced. This is not easily interpreted by observing a single value (e.g. $F_1$ score), especially if there are multiple models of similar performance. Some would argue that both recall and the $F_1$ macro score would have identified this behaviour. I argue that there are limitations to what any individual metric can identified given enough eventualities. This is why all of the metrics are considered throughout the thesis.

Optimizing models in regards to the $F_1$ weighted score is done by utilizing weights during the training phase. It is also accomplished by tuning hyperparameters to maximize the $F_1$ weighted score rather than classification accuracy. The final evaluation is done by observing a confusion matrix featuring the classification result of the test data. A final suggestion to make the model better suited for the task at hand is presented in the result. The idea is to make models support multiple predictions per sample.

### 7 Discussion

The first research question can be considered answered based on the result - it is possible to classify Darknet advertisement based on content. Granted, there are multiple ways of interpreting how classification should be evaluated. Considering the novelty of the domain, I settled for proving that the concept actually works (classifying Darknet advertisement).

The second research question regarding the skewed datasets can also be considered answered; imbalanced class distributions have immense impact on classification performance. Modifying models, evaluation metrics, and weights, are powerful methods to expand upon during construction. With this being said, I argue that knowledge of the goal is more important than the actual performance. Anyone with some knowledge in machine learning knows how to optimize a model in order to improve any arbitrary metric. The true optimization is to ensure that the model is tuned and directed toward the correct optimization goal. Consider

![Confusion Matrix](image)

Fig. 31. The confusion matrix of the best performing MLP trained on the Alphabay dataset, and tested on the Poseidon dataset. The model has three predictions for each sample.
the concept of recall and precision; optimizing either will impact the other. Blindly optimizing without knowledge of what is important is perhaps more hazardous than not optimizing at all (e.g. default values - non-optimized solutions - tend to linger within default values, whilst optimized models most likely made a trade-off which makes the model specialized to that particular problem).

The third research question should be considered inconclusive based on the scope of what it means to generalize. It is said that all optimizations methods perform equally well given an infinite number of problems. Similarly, classification generalizing can evaluated on an infinite number of problem domains. Subsequently, it is extremely difficult to evaluate generalization performance of an arbitrary classification model. I showed that models can perform near flawlessly on unknown material from the same marketplace. I also showed that it models can be considered better that random guessing on similar, albeit different marketplaces. It is possible to deploy models of similar architecture on different domains. However, one should not expect result similar to test-or-validation performance from the same domain.

Another important aspect of generalization - and especially concerning the third research question - is that this is a field of dynamic data; it is bound to change over time. This pose an interesting question that is somewhat out-of-scope of this project, but still deserve some recognition. Language tend to change over time. New words appear, new phrasing emerge, and the way people generally converse tend to change even more rapidly on the Internet. It is important to consider management and administration in order to adapt to these change. Simply implementing and tuning an algorithm will not work. It is an ongoing process. This is arguably a problem relating to model management. However, theoretically speaking, if we were to account for long term exposure to changes in phrasing, language et. al. we could rely on anomaly detection techniques. This would be an interesting addition to the current idea of classification where statistical abnormalities are taken into account. Since this technique is somewhat separated from the actual language, we could potentially implement a more robust model.

The result of this thesis is that a filter should not be implemented by a traditional classifier under these circumstances. It is too narrow. The need for more accurate data can not be stressed enough. This thesis conclude that it is definitely possible to achieve the goal, but the success depends on the source material. The biggest problem when trying to operate on previously uncharted territory is that there is no real material to start with. I was forced to synthesis a dataset by deriving information from scraped sources. Subsequently, the data quality was poor. Data driven development is prone to underwhelm under these circumstances.

The prototype implemented in this thesis can classify advertisement based on content in a satisfactory manner. The cross domain testing did not result in a high accuracy, but I think that it should not be interpreted as a failure. Evaluating the false predictions led me to believe that ambiguity had a great role to play in the underwhelming result. It is my opinion that some samples where better explained by the predictions performed by the model, rather than by the actual category. This is not that unexpected since there is no guideline on “how to categorize material” on Darknet markets. I think that a more generalized topic model should consider more than just one class when evaluating material. Creating a filter on a binary basis (it either belongs to a class, or it does not) is perhaps not the best way to implement a filter for this reason. However, any other type of classifier or topic model would require even more data quality.

I can summarize the entire result in one sentence: more accurate data is required.

### 7.2 Scalability

Working with NLP and the bag-of-words model require lots of memory. Creating matrices of hundreds of thousand columns (fea-
tures), and an equal amount of rows (documents) is the primary bottleneck in this type of implementations. Insightful readers might argue that a well maintained dataset is equally important and an even greater threat to the success of a model. Although this is true, it is not limiting in terms of performance. The KNN algorithm is a good example of this. The default implementation of KNN require all samples to be remembered by the model in order to predict future samples. There is a limit to how many samples can be stored in such systems. Intuitively, there is a performance limit to what can be done under such circumstances; it will not scale to infinity. Furthermore, the outcome is independent of how well tended the dataset might be. It is therefore important to remember that the algorithm itself might limit the outcome of any result when attempting to extend the existing solution.

7.3 KNN

KNN is susceptible to the curse of dimensionality. Large feature spaces that characterize most bag-of-words models result in large neighbourhoods. The sparse feature space (also a result of the curse of dimensionality) means that samples have to look far and wide for neighbours. The exhaustive hyperparameter grid search featured in the result section is an excellent example where this problem emerged. Something is fundamentally wrong when one neighbour is the preferred number of neighbours. It can be considered overfitting.

7.4 SVM

Quadratic complexity in regards to the size of the training data is doubtlessly a shortcoming. Questioning the need for a big dataset is probably an initial thought for most people faced with this dilemma, but NLP do require a big dataset. The curse of dimensionality is the primary reason for that. The bag-of-words model result in a large feature space, which subsequently result in the curse of dimensionality, which in turn result in worse generalization.

Limiting the size of the training data is therefore not the solution.

A better way to manage this problem is to keep it simple. The kernel trick is often referred to as a way to boost performance to the SVM, but it also introduce more complexity. On top of that, RBF is considered one of the most complex (default) kernels and introduce even more complexity. The kernel trick allows the model to operate in a high-dimensional space rather than the initial input space. However, the input space is already huge given the nature of NLP, especially in combination with the bag-of-words model. Increasing it even further to find latent variables is bordering towards overdoing it. Staying with the basic linear kernel might outweigh the performance gain of the RBF kernel. As with many other methods there is a trade-off between result and complexity.

7.5 MLP

MLPs are known to scale with additional data. MLPs are also known to perform well on NLP problems. Some interesting alternatives to traditional classifiers are emerging in state-of-the-art research. I think MLPs will be incorporated in other solutions and methods to improve already saturated fields. Google’s AlphaGO project is a project where methods such as Monte Carlo simulations are used in combinations with ANNs in order to solve complex problems[50]. Similar projects are emerging every day. MLPs are simply stepping stones in the field of ANNs. Other powerful alternatives are probably better suited for NLP as well; convolutional neural networks and recurrent neural networks are techniques that are known to scale well in this domain. However, it is important to match problem with complexity and not create advanced solutions to trivial problems. If this project is to be continued I would argue that other representations than the bag-of-words model are needed. The scaling problem (whether it is KNN, SVM or MLP) is primarily throttled by the feature representation. It does not matter if the classification model scales when the representation does not.
We are currently solving the problem by using cloud services, out-of-core learning, and other methods to manage huge sets of data that is implemented to run in the memory.
8 Future works

8.1 Incorporating images

Classifying advertisement based on content is not necessarily restricted to text alone. Incorporating image classification on top of the current implementation is an interesting next step in improving the overall accuracy. Most Darknet markets have both an object description and an image relating to the listings. Combining the methods and comparing the results is a natural extension of the thesis. Furthermore, it would create an interesting dynamic - similar to ensemble learning - where a voting mechanism is required.

8.2 Advanced feature engineering

Adding advanced features such as part-of-speech tagging or lexical ordering would be interesting.

8.3 Extending the dataset

It is a large dataset, but the amount of labelled samples is quite small. The biggest improvement in terms of overall accuracy is probably connected to both the quality and the size of the dataset.

The most direct approach in order to extend the dataset, aside from manually validating samples, is probably to sample listings based on keywords. All classes are characterised by keywords. One example is that some prescription drugs such as Vicodin, Diasophan, or Demerol, appear frequently in the Drugs and medical supplies-category. These can arguably be used to distinguish other similar listings as they are unlikely to be found in any other listing. Similarly, weapons can be found using known weapons name (e.g. AK5, AK47, or Glock). The downside is that there is no way of telling if material found in this fashion is related to advertisement. It is quite possible that some arbitrary web page in the dataset is simply mentioning the keyword while discussing something else. The problem is thus to validate the new samples and make it both quantifiable and accurate.

Extending the dataset by using semi-supervised learning would be an interesting attempt to increase the size of the dataset (but probably at the expense of data quality). This would create more data, but the immediate drawback is that it would be difficult to evaluate the result. Any mistake would cause false positives to propagate through the system and cause further errors.

Another method to extend the dataset is to synthesis new samples using current samples. Randomly splitting and combining multiple samples from the same class might cover more of the feature space, and thus combat the curse of dimensionality. Adding some noise (e.g. random words) to some samples is bound to show some interesting results. However, it is also likely to result in overfitting.

9 Summary

This thesis presents different methods to classify unstructured- and imbalanced data. The process leading up to the result starts with preprocessing in terms of synthesising a coherent dataset using scraping- and sampling techniques. Labels are extracted using regular expressions in unison with domain knowledge. The resulting data can be considered an imbalanced multiclass dataset. The dataset is then broken down into different features. The transformed data is evaluated based on size, coherence and intuition. PCA and LDA transform the data into yet a new feature space that act as input for different classification algorithms - KNN, SVM and MLP are the three supervised machine learning algorithms featured in this thesis. Models are tuned using a k-fold cross validation. The result is evaluated using appropriate metrics such as the F1 measure and confusion matrices. An additional test set is applied to the best performing model in order to investigate the possibility of extending the classification algorithms to other domains of
similar content. An interesting approach to improve performance can be applied by allowing for multiple predictions on the same sample. The extension might increase performance if recall is considered more important than precision\(^2\). Further studies should be conducted in order to assign more appropriate labels to samples; mainly by incorporating multiple labels to every sample.

2. Require a certain way of calculating recall


10 Appendix

Fig. 32. Alphabay product subcategory distribution.

Fig. 33. Poseidon product subcategory distribution.
<table>
<thead>
<tr>
<th><strong>Alphabay</strong></th>
<th>Electronics</th>
<th>Money</th>
<th>Fraud</th>
<th>ExploitKits</th>
<th>Other</th>
<th>Hosting</th>
<th>Other</th>
<th>Other</th>
<th>Legit Software</th>
<th>Silver</th>
<th>Social Engineering</th>
<th>Fraud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appliances</td>
<td>FakeIDs</td>
<td>Personal Information</td>
<td>Exploits</td>
<td>VPN</td>
<td>Hand Weapons</td>
<td>Steroids</td>
<td>Other</td>
<td>Other</td>
<td>Other</td>
<td>GameKeys</td>
<td>Gold</td>
<td>Carding</td>
</tr>
<tr>
<td>Digital</td>
<td>Jewelry</td>
<td>Other</td>
<td>Security Software</td>
<td>Socks</td>
<td>Ammunition</td>
<td>Benzos</td>
<td>GameKeys</td>
<td>Gold</td>
<td>Carding</td>
<td>Other</td>
<td>Social Engineering</td>
<td>Fraud</td>
</tr>
<tr>
<td>Clothing</td>
<td>Electronics</td>
<td>Accounts</td>
<td>Fraud</td>
<td>Botnets</td>
<td>Malware</td>
<td>Explosives</td>
<td>Psychedelics</td>
<td>Other</td>
<td>Other</td>
<td>Other</td>
<td>Social Engineering</td>
<td>Fraud</td>
</tr>
<tr>
<td>Other</td>
<td>Other</td>
<td>CVV Cards</td>
<td>Explosives</td>
<td>Long-Range Guns</td>
<td>Opioids</td>
<td>Cannabis</td>
<td>Hashish</td>
<td>Tobacco</td>
<td>Dissociatives</td>
<td>Paraphernalia</td>
<td>Ecstasy</td>
<td>Other</td>
</tr>
<tr>
<td>Clothing</td>
<td>Dumps</td>
<td>Fraud Software</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Poseidon</td>
<td>Carding</td>
<td>Money</td>
<td>Fraud</td>
<td>Software</td>
<td>Erotica</td>
<td>Security Software</td>
<td>Non-lethal Weapons</td>
<td>Hash</td>
<td>Digital Products</td>
<td>IT</td>
<td>Hacking</td>
<td>Making Money</td>
</tr>
<tr>
<td>Electronics</td>
<td>Accounts</td>
<td>NB</td>
<td>Security Anonymity</td>
<td>Pistols</td>
<td>Drugs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accounts Banks</td>
<td>Other</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 10**
Category mappings between Alphabay and Poseidon. Parent categories are displayed in light blue cells and sub categories are displayed in transparent cells. Columns list the subcategories belonging to each.