Master Thesis:
Deep Anomaly Detection for Text Documents

Philipp Schoneville

Cognitive Systems: Language, Learning and Reasoning

First reviewer Prof. Dr. David Schlangen
Second reviewer Dr. Lutz Goldmann

July 2021

Abstract

Anomaly detection describes the process of finding data points that are unlike the majority of data points in a given dataset. This can be an important first step to clean up a new dataset, as to ensure that such data points will not have a negative effect on the training performance, or to remove outliers automatically in a machine learning pipeline, either as the main task or as a pre-selection step. Anomaly detection using deep neural networks is a relatively recent but promising approach with a variety of different ideas which are introduced in this thesis. Additionally, applying outlier detection techniques has rarely been tested on document data, which can combine both textual and visual information.

The goal of this project is to give an overview of the state of document feature extraction and outlier detection techniques. Then these approaches are evaluated to solve both unsupervised and supervised outlier detection problems for document datasets. This reveals and confirms a number of important ideas such as using dimensionality reduction techniques like UMAP improving feature embeddings in unsupervised settings, the use of deep one-class networks and the effectiveness of combining textual and visual features. Lastly, an improved performance when using deep learning techniques over traditional algorithms could be established within the evaluation settings.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Modern data processing pipelines require an ever-growing amount of data to function at their best, as deep learning model accuracy typically scales with dataset size [Hes+17]. The accumulation of a lot of new data points is often in conflict with the requirement for datasets to be clean since data mining is highly automated and the resulting datasets are too big to be cleaned manually. Among other factors, such as missing data, inconsistencies, wrong labels or duplicates, a dataset may be unclean due to **anomalies**. Such anomalies (also abnormalities, deviants or outliers) are data points that deviate from the majority of the data sufficiently and are significantly fewer in numbers than the normal data [Dom+18]. This is illustrated in figure 1 where N_1 and N_2 are regions where the bulk of the data lies and the data points O_1, O_2, as well as the smaller region of O_3 might be considered as outliers [CC19].

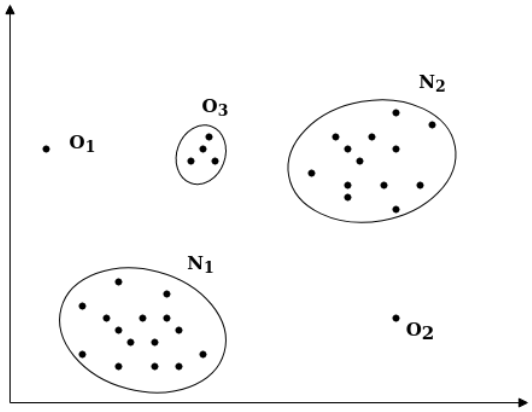


Figure 1: Illustration of anomalies in 2D dataset. O being outlier points or clusters and N being inlier clusters.

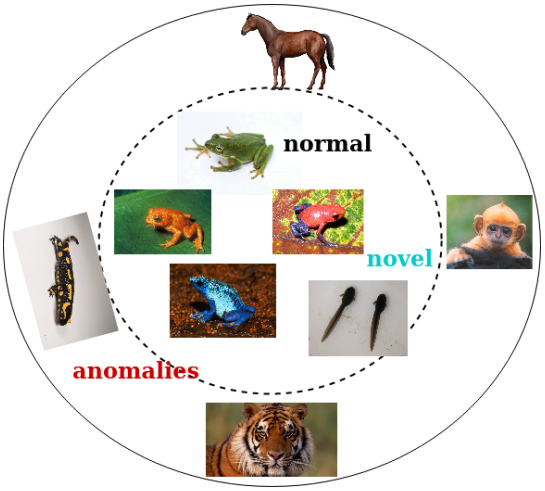


Figure 2: Anomalies and novelties in an image dataset for frog image classification.

As mentioned before, anomalies might be the reason for an unclean dataset that is undesirable to work with in downstream tasks. E.g. when training a classifier with labeled data, anomalous data might confuse the classifier about the exact features of a certain class. Similarly, when analyzing a new dataset, it might be desirable to

not use a class with a proportionally small amount of data points because it is not seen as important enough and/or to avoid creating heavily class imbalanced datasets which are hard to work with. However not all anomalous data is undesirable and instead new data patterns might be actively searched for. This new data could then be included into the dataset as its own new class or added to existing classes (see "novelty" in figure 2).

Related to this is the problem of finding anomalous data when doing **inference** of trained models on new data points. This might be the main task of a system, such as a fraud detection system that discovers anomalous patterns in data to return a probability that a data point is "normal" or deviates from the norm and is under suspicion for fraud. It might also be an extra feature of a system to avoid a wrong output: An image classification model that was trained to differentiate between n classes of dog breeds should not assign any of these n classes to an image of a dolphin but instead reject it as "not a dog". A classifier that knows what it does and does not know might be considered an open world classifier or open classifier. It does not work under the closed-world assumption that the classes in the test data must have also been in the training data [SXL17]. In either case, a bound around *normal* data needs to be established during training so that new data points that don't fit within this bound can be rejected as outliers during inference. A classic way to tackle this problem is with one-class classification, where training is done primarily on the objects of one class [Ruf+21] [Ruf+18] [Sch+99].

Anomaly detection is relevant for a number of use cases (see figure 4). One common case is *fraud detection*, where the inliers are often regular contracts between two partners such as banks or insurance with clients. Anomalies are then a combination of features in a data point (or alternatively a series of such data points) that does not occur frequently, either when compared to the distribution of data points for a single client (e.g. a bank customer suddenly showing a very different spending behavior for his credit card) or a single use-case (e.g. an insurance detecting a much higher price for a routine repair). Other applications which produce high amounts of regular observations, where detection of outliers is however crucial, are malware detection, the medical field or monitoring industrial machinery. Lastly anomaly detection has been popular for detecting fake news and trends in social networks [CC19].

Depending on the type of data and problem outlier detection can be a trivial problem or a very complex one. When working with a single variable and a unimodal distribution, such as the word count in a text or a monetary value in fraud detection, outlier detection can come down to finding a single number that can describe a good

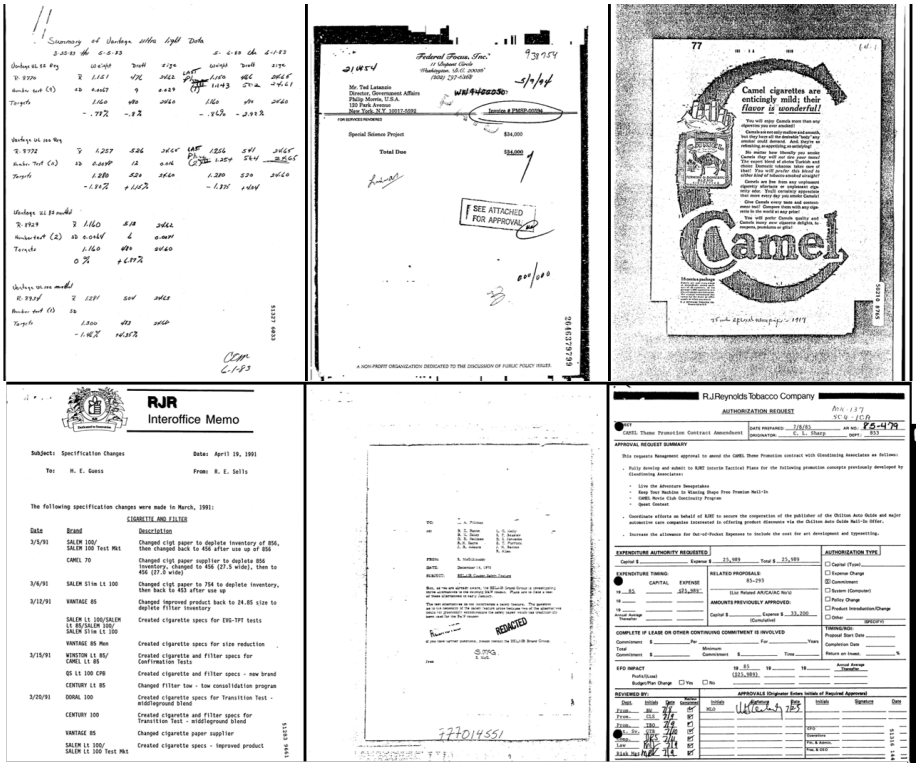
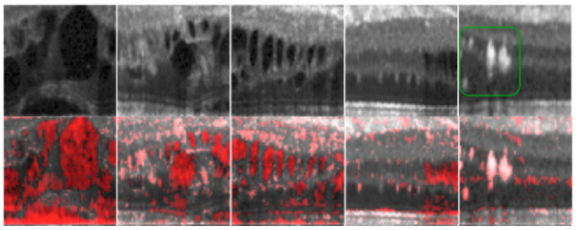


Figure 3: Sample documents from used RVL-CDIP dataset.

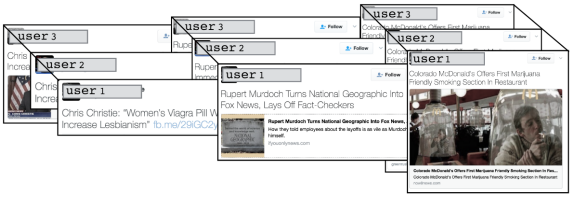
threshold to separate the majority of the data from outliers. Many real world problems however use multimodal distributions of high-dimensional data. Such complex data requires more complex algorithms to detect patterns, create a concept of distance between data points and finally decide which data points are not part of local clusters of *normal* data. Such techniques are for example the One-Class Support Vector Machine (OC-SVM) developed in 2001, [Sch+99] up to very recent advances in deep learning architecture, which have shown promise of significant improvement of accuracy [CC19] [Zha20].

One such complex problem is document classification and anomaly detection for documents as it typically requires a certain level of language understanding from a model and in many cases additional information on location of information on a document and/or visual structure. Some sample documents from the RVL-CDIP dataset can be seen in figure 3: Many of the documents have information arranged in table format, where (relative) location of tokens and overall visual structure are major cues for the document type. Data points within a class can additionally

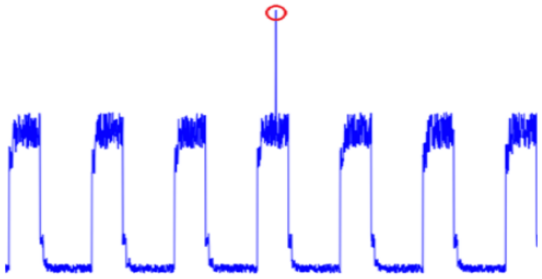
(a) Illegal Traffic Flow detection.



(b) Detecting Retinal Damage.



(c) Fake News detection on Twitter.



(d) Internet of Things (IoT) big-data anomaly detection.

Figure 4: (a) Video Surveillance, Image Analysis: Illegal Traffic detection [Xie+17] (b) Disease detection and treatment monitoring [Sch+17] (c) Fake News detection on Twitter [RSL17] (d) Sensor Networks: Internet of Things (IoT) big-data anomaly detection [Moh+17]

be quite diverse which makes setting clear boundaries between inliers and outliers hard (see figure 3 for sample documents). And lastly the text data first has to be embedded into a vector to be processable by anomaly detection algorithms which is a complex problem on its own. This can be approached with classic techniques such as a simple vectorization of individual words [Mik+13a], the extensions of word2vec to whole documents Doc2Vec [LM14] or modern deep learning techniqures such as text representations extracted from Transformers [BPC20] or the state of the art approach of combining text representation, token location and visual information into one represention [Xu+20b].

1.2 Goals

The first goal of this thesis is to introduce the theoretical background as well as solutions found in literature for both *document representation* - that is how the document data can be best represented in vector form for further processing - and *outlier detection* - the process of finding data points that are unlike the majority of the dataset.

The second goal is to evaluate the performance of some of the ideas found in literature empirically on two different tasks for outlier detection for text documents:

- **Unsupervised Outlier Detection**: The starting point for this task is a new unlabeled dataset and the goal is to find anomalous data that should be labeled as such and possibly removed from the dataset. This is one of the necessary steps during data analysis and pre-processing, before a new dataset can be used for further tasks.
- **Supervised Outlier Detection**: For supervised outlier detection a labeled dataset is necessary, as well as a definition which labels are inliers or outliers. This can for example be a dataset with class labels and with the definition that certain classes are inliers. The goal is then to train a model that learns a concept of how the inlier data points look like to distinguish them from new and unseen outlier data points after training.

The experiments for the supervised task are previously unseen in literature, as the combination of documents and outlier detection has not been done before. Therefore results can't be directly compared to previous work. Nevertheless this thesis will

introduce results of an extensive parameter grid search and show the effectiveness of several ideas in literature applied to this task.

1.3 Thesis Structure

First, the **chapter 2** provides an overview of different techniques to obtain a feature vector for document data. This chapter is split into sections for derivation of such vectors from *visual* data, from *text* data, as well as a combination of the two.

Following this are two chapters that introduce concrete problems and their solutions. The **chapter 3** introduces the problem of finding anomalous data points in new and *unlabeled* datasets, while **chapter 4** is about training a model on a *labelled data* which can generalize to new and unseen outlier classes. Both of these chapters provide *background* on a variety of definitions and related algorithms to solve this problem found in literature. Next, a problem and the *methodology* to solve it are described: This includes the used datasets, the pre-processing steps, the proposed solution and pipeline, as well as the metrics by which the performance is measured. Following this, the results of various experiments related to this problem and different possible solutions and parameter settings are presented and discussed, along with ideas for possible improvements.

Lastly results are summed up with an outlook for future developments regarding outlier detection for text documents in **chapter 5**.

2 Document Description

Document data enters a pipeline in the form of an image file - such as a photo, a scan or a digital document - or a text file. However to allow further processing and to classify a document (either as one of multiple classes or deciding between inliers and outliers), the document first has to be transformed into vector form. Such a vector holds information in the form of individual features that might describe a measurable characteristic of the document. This feature vector or vector representation of the document can then be further leveraged by algorithms to make decisions.

Traditionally such vectors had to be manually created by using knowledge of the problem and hand pick features that should be given to the algorithm within the

feature vector. This however is problematic because it requires extensive domain knowledge and can severely restrict what a model is able to learn. It has been shown that in many cases it is better for the model to learn a useful vector representation from data that is as raw as possible (such as pixel data for images) in end-to-end deep learning models. These models can however be computationally expensive, which is why a popular approach recently has been the use of pre-trained models to avoid the expensive training process and still leverage the power of learned features.

The following sections within this chapter will further look at different approaches to document description through textual and visual features, and how traditional methods compare to modern deep learning approaches. The efficiency of these models for the problem of outlier detection for text documents will be shown in the result sections of chapter 3 and 4.

2.1 Textual

Textual representation relies solely on the digital character sequence in text files. The text could be extracted from already documents such as e-mails, native PDFs or other files that were created digitally. In many cases however the text has to be extracted from the image of a text document through the means of *optical character recognition* (OCR) algorithms that can identify and extract characters and words from pixel based images (see chapter 4.1 for how this is done in this project).

2.1.1 Traditional Methods

The first set of methods fall into the category of feature engineering and only take into account word frequency for the whole document. Nevertheless they have empirically been shown to be a good baseline to represent text [LM14]. The simplest method to represent a text is through the use of **Bag of Words** (BoW) and works as such: Each text document is assigned a vector of the length u, where u is the number of unique words in the whole text corpus. At each position there is one value that corresponds to exactly one unique word in the corpus. The value itself represents the absolute frequency in the given document (see figure 5). This idea can easily be extended from looking at the count of only one unique word per value in the vector to the counts of unique short sequences - so called n-grams, where n stands for the sequence length.

1. They like to play chess and they like to watch football.
2. We like to play chess. We do not like to play basketball.

	They	We	like	do	not	to	watch	play	chess	and	football	basketball
1.	2	0	2	0	0	2	1	1	1	1	1	0
2.	0	2	2	1	1	2	0	2	1	0	0	1

Figure 5: The Bag of Words algorithm.

Additionally, to reduce the weight of words that hold low information because they appear very frequently in every text such as articles ("the", "an"), prepositions ("in", "for") or other words specific to the text corpus, and to instead value those words higher that occur rarely and make the different documents distinguishable, a weighting scheme can be used. The most popular approach is term frequency–inverse document frequency (**TF-IDF**) which is a product of the term frequency (in the simplest case the count of words in a document) and the inverse document frequency which describes the information value of a word and is found by taking the number of documents in the corpus divided by the number of documents that contain the word and scaling it logarithmically [MRS08].

Both BoW and TF-IDF rely solely on the assumption that only documents that have a similar number of exactly matching words and n-grams are similar to each other. This however fails to capture the relatedness of documents based on their semantic information, which can be represented in language through a different set of words or n-grams. As an example, the words "baby" and "child" are closely related but would be treated as entirely separate cases by BoW or TF-IDF.

2.1.2 Word embeddings

To extend text representation to capture semantic meaning, a popular approach and basis of many models is to encode each word with a vector of fixed length that contains semantic features which are learned by the respective algorithm. Word embeddings such as **word2vec**, invented in 2013 by Mikolov et al. [Mik+13b] can be efficiently obtained using a small feedforward neural network and a large text corpus to train on. They are based on the idea that similar words appear in the same context, meaning they are surrounded by the same words. Such statistics for co-occurrence

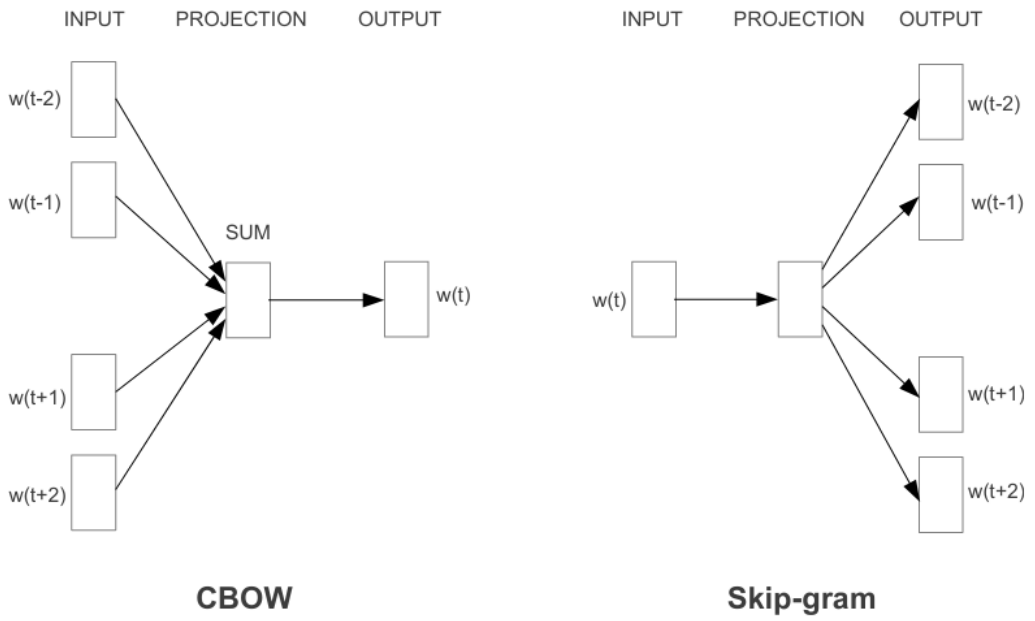
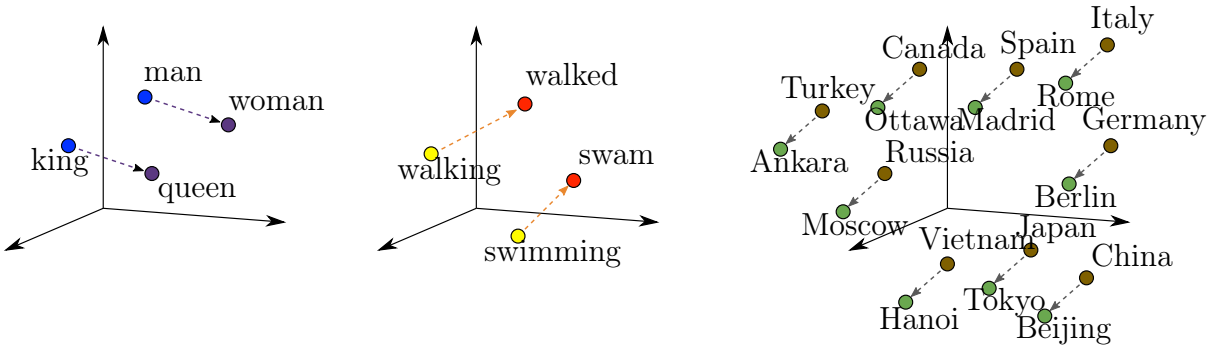


Figure 6: Word2vec training using either CBOW or skip-gram [MLS13].

can be encoded into a vector during training using two different methods: *CBOW* (Continuous Bag of Words) and *skip-gram*. In CBOW the objective is predicting a word depending on its *context*, the n words surrounding it (called the *context window*). That means the the context words serve as the input of the neural network and the output will be the predicted word vector. Conversely for skip-gram a single word is the input and the objective is predicting its context (see figure 6).

A similar popular word representation are *GloVe* (Global Vectors for Word Representation) which, in contrast to the purely local approach of word2vec, take into account both local and global word distributions. A word co-occurrence matrix, that describes which words occur together throughout the whole corpus, is built before training and serves as input to the training model [PSM14]. While the algorithms and intuition behind word2vec and GloVe differ, performance has been shown to be very similar [NHB17]. Lastly, there is *fastText* which uses the same basic training algorithm as word2vec but does not work on words but on character n-grams (also called subwords). As an example the word "likes" in the form of a character 3-gram is split into <li, lik, ike, kes, es>. The intuition behind fastText ist therefore that it can better handle morphologically rich languages such as Czech or German and understand the information encoded in their affixes. Another big advantage is the fact

that unlike word2vec it can handle out-of-vocabulary words. While word2vec learns vector representations for a fixed vocabulary of words and is unable to represent any new words, fastText learns representations for each subword and can therefore represent a new word by simply summing up over its subword vectors [Boj+17].



Male-Female Verb Tense Country-Capital

Figure 7: Analaogies in the vector space spanned by word embeddings. Dimensions reduced to 3. [1]

Word embeddings have been shown to be highly interpretable by simply measuring the cosine (or euclidean) distance between each other. Thus it can be shown that words that are close to each other semantically are also close to each other in the vector space. E.g. all colors are close to each other, the names of the months or countries. Furthermore it can be shown that reasoning with the vectors is possible by using the vector offset and solve questions such as "a is to b as c is to d" with d being unknown (see figure 7:

$$v_b - v_a + v_c = v_d$$

The resulting vector v_d is then close in cosine distance to a word that fits the analogy, such as "King - Man + Woman" resulting in the word "Queen" [MYZ13].

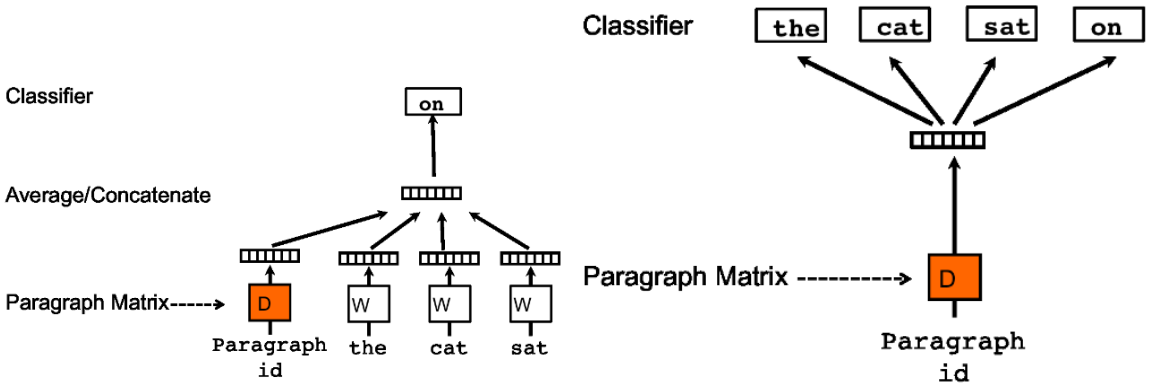
2.1.3 Document embeddings

Creating embeddings for documents that can capture similar semantic relationships between each other by measuring their cosine distance is desirable, as it is a prerequisite for many downstream algorithms to efficiently classify the documents (or in

[1] developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space in February 2021

the case of this thesis divide them into inliers and outliers). Extending the training process for word2vec embeddings (shown in figure 6) to instead learn vector representations for texts is relatively simple and called **PV-DM** (Paragraph Vector - Distributed Memory) or **Doc2Vec** (see figure 8a). The only addition is a one hot vector that represents a unique document ID for each document in the training set and the parameter matrix D to obtain a fixed length embedding from the document ID. A concatenation or average of this vector together with the vectors of n words of context (multiplied by W as before) is used to attempt to predict a missing word of the context. However because documents are thought to be unique unlike words which belong to a fixed size vocabulary and can each be assigned a specific one hot encoding and simply be multiplied with the learned parameter matrix W to obtain their fixed length word embedding, PV-DM works differently than CBOW for word2vec during inference: When inferring the document vector for a new document, the learned weight matrix for the words W, as well as the softmax weights U, b are kept fixed and the new document is assigned a randomly initialized vector. Forward propagation is then run as before by choosing a context of n words and another word missing from the context as the word to predict. Then stochastic gradient descent is used on D for a fixed number of steps. The resulting document ID and weights D can then be used to calculate a fixed length document vector [LM14]. Just like word2vec models, Doc2Vec models can be trained on a large corpus and then later used to infer embeddings for documents from new documents without fine tuning [LB16]. A notable variation of PV-DM is Doc2VecC [Che17] which takes an average over word embeddings during learning and also implicitly implements regularization to diminish the effect of non-discriminative words by randomly dropping words from the documents during training.

Word embeddings can serve as an input for more complex deep learning architectures that are computationally expensive and require thousands or millions of parameters to obtain document embeddings. It has been shown however that relatively simple and parameter-free pooling operations can serve as a strong baseline or even outperform such models on document similarity tasks [She+18] [Jou+16], especially when it comes to out-of-domain data [Wie+16]. The simplest pooling operation is average pooling which, as the name suggests, takes an element-wise average over the word vectors computed for a given sequence. This can be often be further improved by using a weighted average that takes into account word frequency and that common words don't contribute a lot to determine text similarity. Such a weighing scheme can be the previously discussed *TF-IDF* or SIF (smooth inverse frequency) [ALM17]. The intuition that only a few key words have an impact on the final pre-



(a) (b)

Figure 8: (a) Learning a paragraph vector (PV-DM, Paragraph Vector Distributed Memory): This framework works similar to word2vec, except for the additional paragraph token. When trying to predict the fourth word from three input words, this paragraph vector can serve as a memory of the topic and provide missing information to the context. (b) Distributed bag of words (PV-DBOW), trying to predict words from a sampled text window, given the paragraph vector and a single sampled word from said text window [LM14].

diction also led to the proposal of **max pooling** [She+18]. For every dimension in the word vector, only the maximum value among all vectors is kept, eliminating "average" values and keeping only the most salient ones. Stacking (concatenating) these two types of pooled embeddings has also been shown to be successful (similarly to how stacking different kinds of word embeddings can be a good strategy [Akb+19a]). These two methods can also be combined so that the average is taken over n-grams of the word vectors and then max pooling is used on the output globally, to preserve word-order an spatial information and increasing performance (*hierarchical pooling* [She+18]). Lastly the **Word Mover Distance** (WMD) [Kus+15] has been proposed to measure similarity between documents and successfully adapted to create document embeddings that can outperform all the previously described document embeddings on text classification tasks [Wu+18]. However this method can be prohibitively slow for long documents. WMD works by first removing all frequent words (stop words) from the documents and then taking the minimum cumulative Earth Mover Distance (Wasserstein distance) between all remaining word vectors.

2.1.4 Deep Learning Based Approaches

Stepping away from traditional approaches, in recent years **deep learning** models have increasingly dominated the field of NLP and led to new state of the art performances in all tasks including text classification [Min+21]. These models by their nature create intermediate representations as hidden layer activations that they are passing on between layers which can be extracted and used as interpretable embeddings of their input. Both CNNs and RNNs/LSTMs have been used successfully for the task of text classification [Min+21], and a few attempts have been made to do feature extraction from **CNNs** [LZV18] and **LSTMs** [Pal+16]. However most recently new state of the art performances have come from large **Transformer** models that are able to handle text sequences similarly well to their LSTM counterparts but are not limited by sequentiality during training, being based solely on the attention mechanism [Vas+17], and can therefore handle training with the biggest datasets available (this difference in performance might not exist when training on a small corpus [Eze20]) and become increasingly bigger. This increase in depth and parameter count has so far culminated in the 175 billion parameter language model behemoth GPT-3 by OpenAI [Bro+20]. As training these models from scratch has become prohibitively expensive and even fine-tuning is becoming problematic, other approaches to extract information from the pre-trained models have become more popular and feasible. The popular Deep Bidirectional Transformer (BERT) [Dev+19] has shown that simply extracting the features from the pre-trained model's hidden layer and feeding them into a simple two-layer BiLSTM is competitive for Named Entity Recognition (NER) with their fine-tuned model (especially when taking the last four layers and concatenating them). These findings are consistent with later studies for various Transformer based models and a variety of NLP tasks such as NER, semantic role labeling, paraphrase detection, syntactic chunking etc. [Tos+20] [PRS19] [Liu+19] (and similarly for pre-trained BiLSTM and Gated CNN [Con+18]). Previously it has not been possible to feed entire documents into Transformer architectures, due to self-attention scaling quadratically with the sequence length. Thus other approaches had to be taken such as chunking the data into sequences below the 512 token limit. However recently there have been a variety of new approaches to tackle this problem, such as the **Longformer** [BPC20], which is a Transformer with an attention mechanism that scales linearly with sequence length, allowing a new token limit of 4096 tokens and thus the processing of whole documents at once. The Longformer has been shown to have good performance when used on its own or in conjunction with image embeddings to classify the RVL-CDIP dataset [PMP20].

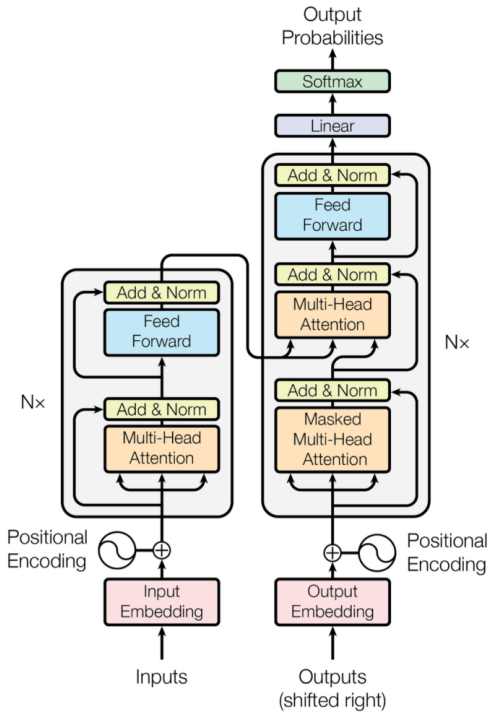


Figure 9: Overview of the Transformer architecture [Vas+17].

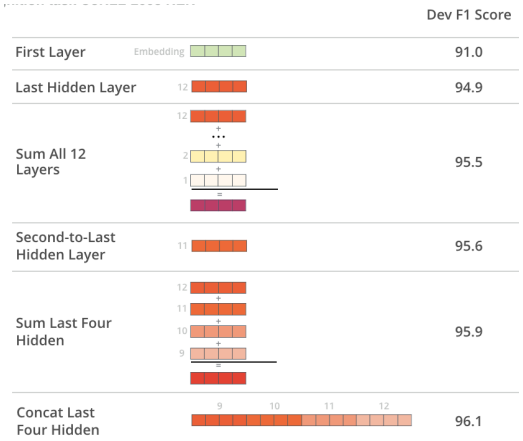


Figure 10: BERT NER (CoNLL-2003) with feature extraction results using different (combinations of) layers from BERT [Ala] Fine-tuning achieves an F1 score of **96.4** [Dev+19].

2.2 Visual

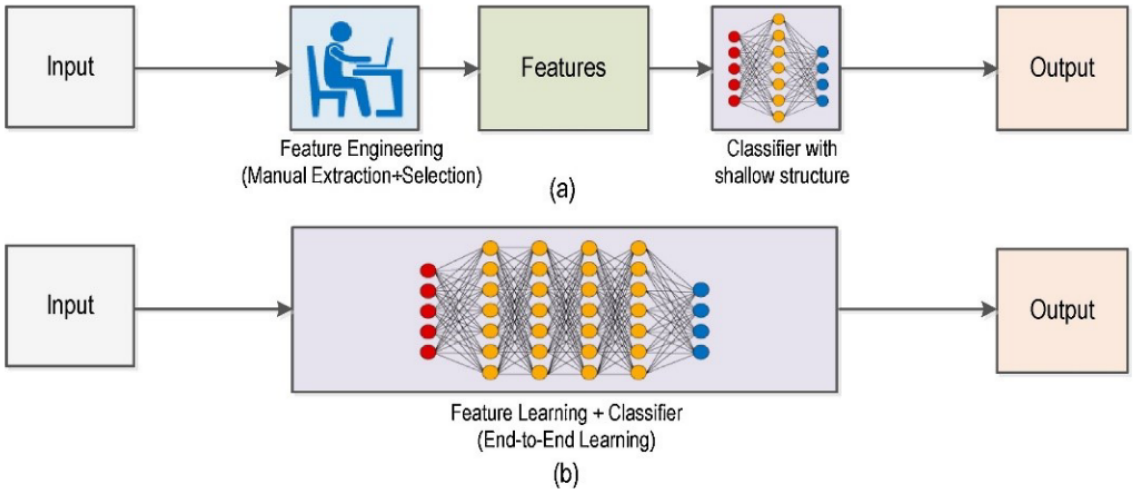


Figure 11: Comparison of traditional workflow (a) where a human has to decide on which features to use as input for the classifier and deep learning (b) where the features are learned during training [Mah+20].

In addition to their textual content, documents can contain a variety of visual information. This includes non-textual content such as images, general design patterns such as standardized document structures or tables and visual variations of the text (see figure 3. Training models that rely purely on visual input has therefore proven successful to classify such documents.

Just like it is the case for the language based classification, visual classification have largely moved away from handcrafted features to learned features within deep learning models (see figure 11). Previously the majority of Computer Vision (CV) tasks relied on a manually defined feature extraction step to capture information about color and structure using methods such as color histograms or Haar-like features (see figure 12 for handcrafted features compared to learned ones in figure 13)[Mah+20].

The biggest breakthrough for deep learning in CV happened in 2012, when AlexNet was able to achieve an unprecedented score in classification on the ImageNet dataset[2], which contains over 14 million images and 20.000 categories, in 2012 [KSH12], using solely an end-to-end deep CNN architecture. CV has since then

[2]http://image-net.org/

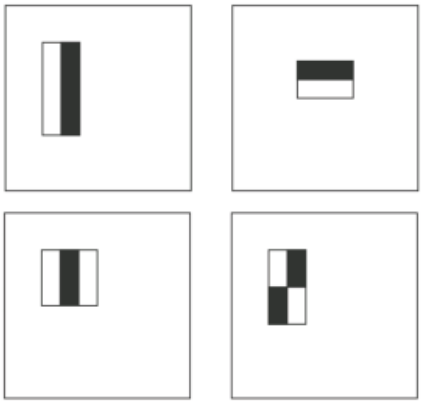


Figure 12: Standard set of Haar features used for edge detection or changes in texture [VJ01].

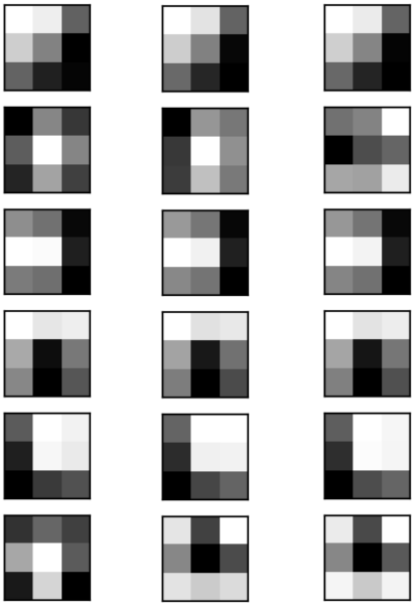


Figure 13: Visualization of the first six VGG16 3x3 filter weights for all three channels [Bro19].

largely shifted to using deep learning models pre-trained on large amounts of data (such as the ImageNet dataset) as the basis for their models and only fine-tune them on their own smaller and more domain-specific datasets, as these transfer learning capabilities have proven to be highly effective [Raz+14]. Despite the fact that there are large domain difference between documents and natural images, because the latter contain objects in variety of 3D poses that can appear anywhere on the image while the former are 2D entities that occupy the whole image [TM17], CNNs initialized with the weights learned on ImageNet have been proven to learn faster and achieve higher classification scores on document data. However pre-training on document datasets will still significantly outperform training on ImageNet [Afz+17] [Stu+19].

VGG-16 , which was used in this project, is an extension of AlexNet. Notably it is deeper (using 16 layers or alternatively 19), alternating between two or three convolutional layers and one max pooling layer, finishing with two fully connected layers and a softmax [SZ15]. VGG-16 has shown better performance on the RVL-CDIP dataset when used for classification [Afz+17]. Additionally, VGG-16 can gain a significant performance boost when using a VGG-16 pre-trained on ImageNet is

used as a starting point to fine-tune on document data, and then using the resulting model to initialize partial models, that attempt to classify the document based only on a part of it (such as the header, footer, left and right side) and lastly concatenating the resulting embeddings [Das+18].

2.3 Multimodal

As previously discussed, document data can contain both visual and textual information that allows to distinguish between them. Combining both visual and textual features into one cohesive classification model can therefore produce better results for classification or outlier detection tasks. This has been shown by Xu. et al. in 2019 [Xu+20b] to set a new state of the art for three classification datasets of documents, including the RVL-CDIP dataset used in this thesis (and with some tweaks improve on it again in their second version [Xu+20a]). LayoutLM takes the popular BERT architecture [Dev+19] with three types of input features (see figure 14): Word embeddings, image embeddings and 2D position embeddings. To do that they first run OCR (Optical Character Recognition) on the whole document and extract each token's text content and position. The text is being embedded with pre-trained word embeddings and fed normally to BERT, together with the position. The token's position provides valuable information to the model about the relative position of a token, such as in the case that a key like "Passport ID" would expect its value to be to the right or bottom of it, but not to the left or top. Additionally the word's location data from the OCR step is used to cut out the relevant part of the image. A pre-trained ResNet [He+15] is then used to generate features for this image of the word, as well as one image of the whole document being run through ResNet to use in conjunction with the CLS token's final representation state[3]. In comparison to the *early fusion* approach taken by LayoutLM (combining the different features before putting them into the singular BERT model), Bakkali et al. [Bak+20] took a late fusion approach by training both an image model and a Transformer as the text model and only joining their output afterwards to calculate a final prediction using fully connected layers on top of the combined embeddings. This architecture was able to beat LayoutLM v2's accuracy score on RVL-CDIP classification of 95,6% with a new score of 97%. Similar improvements through multimodality (using late fusion) over using only text or visual embeddings could be shown for the use case of

[3]Within the sequence-to-sequence Transformer model, the CLS token is the only one not associated with any single word but instead it represents the sequence as a whole and serves as a sort of memory.

detecting fake news [NLW19].

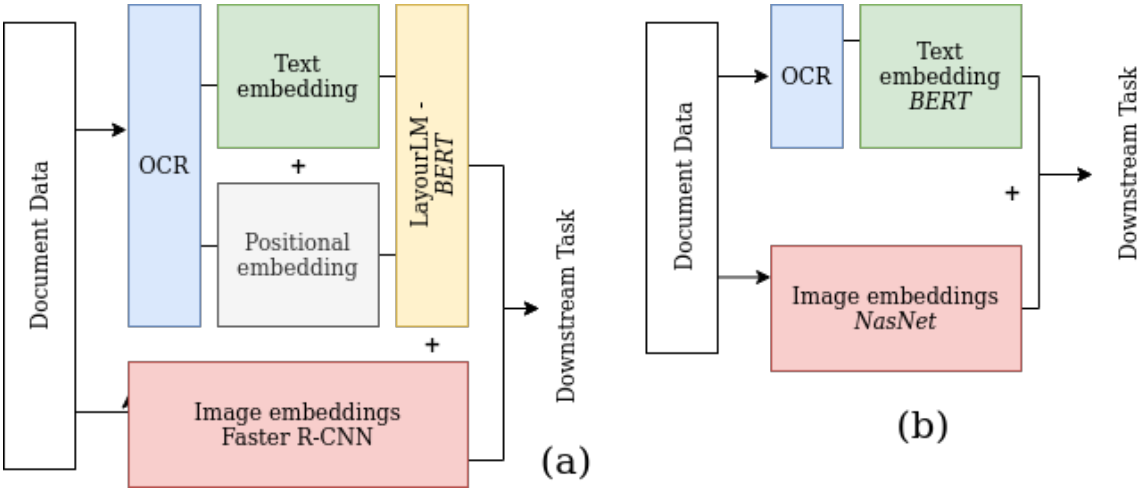


Figure 14: LayoutLM architecture overview (a) and comaprison to Bakkali 2020 (b). LayoutLM uses the additional positional embeddings during fine-tuning of the Transformer while (b) merely concatantes the output features of the Transformer for text and NasNet for images.

3 Unsupervised Anomaly Detection

Unsupervised learning solely concerns itself with extracting information from datasets by learning from data that has *no labels*. Specifically for anomaly detection this will mean that data features have to be learned in an unsupervised manner and then a classifier has to learn patterns in these features to distinguish *normal* (inlier) data from outliers - or, in the case of end-to-end deep learning architectures, the feature learning and classification task is being combined into one model. The classifier (or outlier detector) therefore cannot differentiate between different classes within a dataset and has to rely solely on the assumption that the majority of data belongs to the *normal* data distribution with only a small percentage of it being different enough to be considered outliers. Most algorithms therefore rely on a parameter that gives them a good idea about how big the contamination of the dataset might be (meaning what percentage of the dataset can be considered as an outlier).

Unsupervised anomaly detection is mainly a tool to automate steps to detect corrupted data, data from different data distributions that doesn't belong to the

dataset or data from classes with very few samples (also known as novelty detection) during the data analysis and data cleaning steps when working with new datasets. In the absence of any labels the true contamination of the dataset can only be guessed, a clear definition of *inliers* is hard and novelties within a class are commonly labelled as outliers, when they are in fact useful to keep during downstream tasks. Therefore it is problematic to incorporate models that learned in an unsupervised manner in automatic production pipelines.

3.1 Background

3.1.1 Proximity Based Models

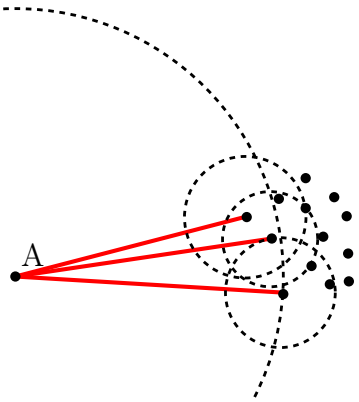


Figure 15: Local Outlier Factor: The idea is to compare a points density (distance to k neighbours) to the density of it's neighbours. Here point A has a considerably lower density than its neighbours.

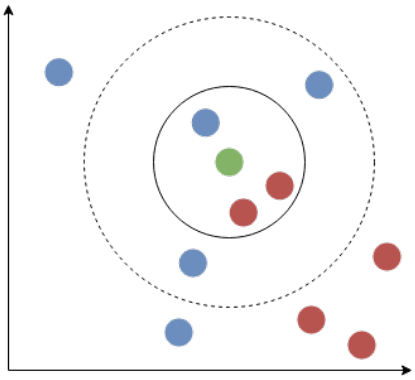


Figure 16: K-Nearest Neighbours: When classifying the green point, it will either belong to the red class if k=3 or to the blue class if k=5.

Proximity based methods rely on the assumption that in the given vector space of features that represent the data points, feature vectors of similar data points are close to each other (in euclidean or cosine distance) while dissimilar data points are far away. Because the data representation models discussed in chapter 2 explicitly or implicitly (if the training task is classification) use this as their training objective,

proximity based methods should work with features extracted from any of these models.

The simplest proximity based outlier detection algorithm is **kNN** (k-th Nearest Neighbour) [RRS00]. This algorithm simply measure the distance of every point to the k nearest neighbours it has (see figure 16), sum up all the distances and then rank these sums of every point. The top n points in this ranking then make up the outliers. This algorithm is simple but relies on a good choice for k to work well and should therefore be run for multiple values of k to give adequate results. Additionally it can only capture global outlierness and fails to capture local outliers such as are common in multimodal distributions.

LOF (Local Outlier Factor) is another simple algorithm that can overcome the weaknesses of kNN by accounting for local density. LOF first defines the k-distance as the a point's distance to its k-th neighbour. Then the *local reachability distance* of points A and B is either their true distance or at least the k-distance of point B:

$$reach - distance_k(A, B) = max\{k - distance(B), d(A, B)\} \tag{1}$$

The *local reachability density* (lrd) is then the average of all of point A's k-neighbours' (short $N_k(A)$ reachability distances inverted. This means a point A's local density is being judged by the distance that its neighbours perceive it to be (see figure 15):

$$lrd_k(A) = 1/(\frac{\sum_{B \in N_k(A)} reach - distance_k(A, B)}{|N_k(A)|}) \tag{2}$$

Comparing a point A's lrd to that of its neighbours then gives an interpretable value of density which is around 1 when the density of its neighbours is similar and higher if the point lies in an area of low density [Bre+00].

Isolation forest is a tree ensemble alternative to purely measuring distance or density, relying on the assumption that outlier points are easier to seperate from the other points in the dataset than inliers. The algorithm thus starts with a subsample of the data and recursively selects one of the features and a normally distributed split value that divides the data. This is done until all data points are isolated (or have the same value) and the points that took the least amount of splits to isolate can then be considered anomalous. Formally this process can be described in a decision

tree, called the iTree. After building many such iTrees, every point in the dataset can be run through all of them to determine an average path length until it is isolated (arrives at an external node) and calculate an anomaly score [LTZ12].

Lastly **HBOS** (Histogram-based Outlier Score) is a very simple yet surprisingly well-performing outlier detection algorithm [GU16] that works especially well for large datasets, as its assumption of independence of the features [GD12] makes for fast computation speeds. The algorithm will first build a histogram for every dimension in the feature vector with only the amount of bins k as a parameter. After normalizing all histograms to a maximum height of 1, each bin in a histogram represents a density estimation. The HBOS of a point A is thus the multiplication of the inverses of every dimension's histogram value (or in the actual implementation the sum of the log values):

$$HBOS(A) = \sum_{i=0}^{d} log(\frac{1}{hist_i(A)} \tag{3}$$

3.1.2 Transformative Models

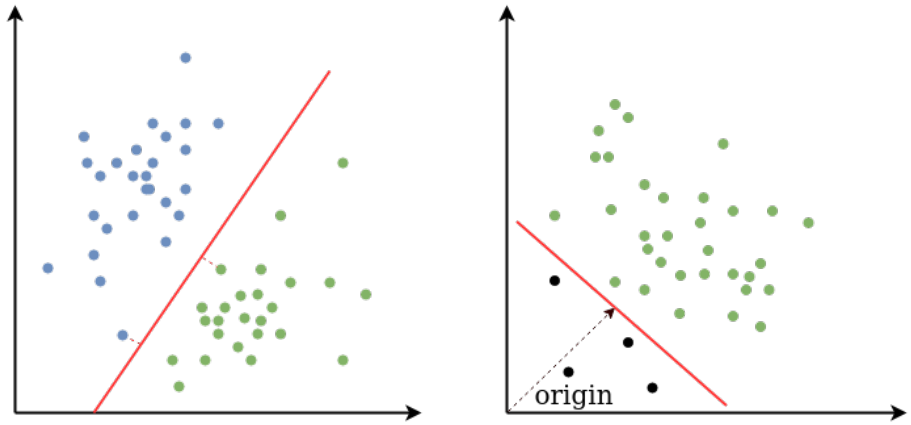


Figure 17: SVM is able to seperate two classes of data with a decision boundary. The closest points to it are called the *support vectors*. The OCSVM on the other hand seperates the data from the origin.

Transformative models (implicitly) transform the features using linear or non-linear operations to find an appropriate boundary that contains only inlier data.

OCSVMs (One Class Support Vector Machines) are a variation on SVMs [Sch+01] [Sch+99]. The learning objective of the classic SVM is finding a hyperplane that seperates two classes of data in n-dimensional space using a labelled dataset. The hyperplane should have the maximum distance to the data points with minimum distance to the hyperplane. These vectors are then called *support vectors* (see figure 17). In an easy case this is possible to do in k-dimensional space where k has the same dimensionality as the input features. However for many problems this does not work and the SVM will instead use linear or non-linear transformation of the features to find an n-dimensional space in which the two classes can be linearly separated with a hyperplane. To avoid a big computational cost by explicitly calculating coordinates in higher and higher dimensions, the trick behind SVMs is to only implicitly look at the n-dimensional space by instead calculating dot products between vectors, called the *kernel trick*. Between the invention of the kernel trick for SVMs in 1992 [BGV92] and the rise of deep learning networks, SVMs have been the most popular choice for binary classification tasks. OCSVMs are an unsupervised variation on SVMs. In the absence of labels, the training objective for OCSVMs is to separate the data points from the origin. This can be seen as a normal SVM where all the data points belong to one class while the origin is the only data point belonging to the second class. The parameter ν in an OCSVM gives an upper bound for the contamination of the dataset.

PCA (Principal Component Analysis), in contrast to an SVM, does not expand the dimensions of the features but reduces them. Typically PCA is used to decrease the effect of variables that are linearly dependent on one another and re-codifying the information of the original features in new uncorrelated linear combinations of the features, thereby reducing the feature vectors' size to k dimensions. PCA has been successfully used for outlier detection [Pas+10][Shy+05] but can - as any dimensionality reduction algorithm - also be used in conjunction with another outlier detection algorithm.

3.1.3 Deep Learning

Autoencoders have been successfully used for outlier detection [Sch+18]. The usual idea behind autoencoders is to obtain an information-rich vector representation with (much) fewer dimensions than the original input vector, using an unsupervised learning setting [Kra91][GBC16]. This is achieved by creating a fully connected neural network with n input dimensions, n output dimensions and a bottleneck hidden state with h dimensions (see figure 18. The learning objective is then to reconstruct the

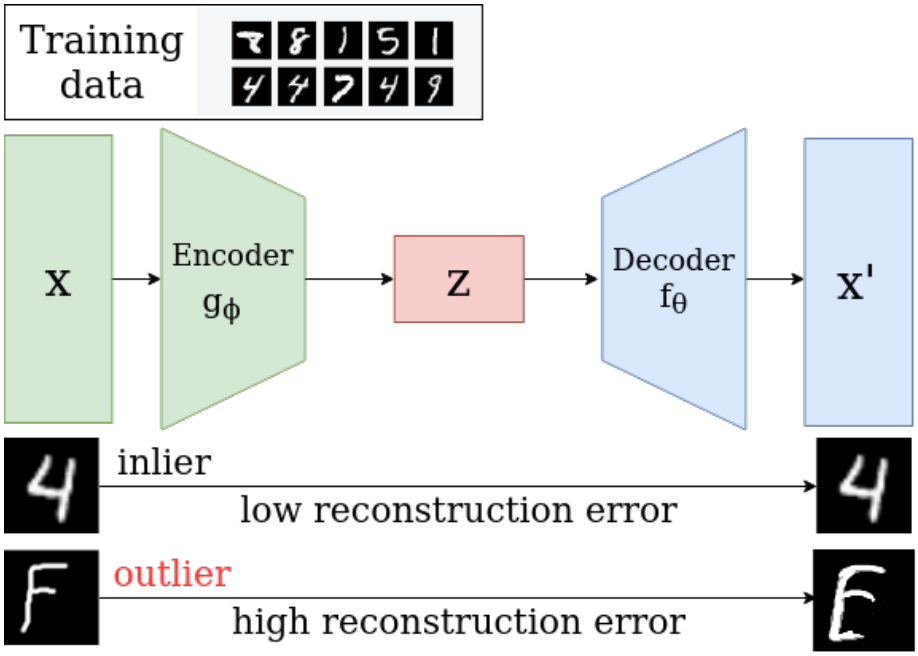


Figure 18: Autoencoder trained on MNIST data: After training the autoencoder's reconstruction loss can be used to distinguish between inlier data (that is similar to the training data) and outlier data (e.g. not numbers but letters).

input vector, resulting in a hidden vector that has enough information to achieve this task. Choosing a high h can result in overfitting, where the network is easily able to store information for every training sample in the hidden vector. Conversely choosing a smaller h prevents this case and forces the network to generalize. A small enough h then forces the network to use the capacity in the hidden vector to accurately reconstruct only the majority of the data points that looks similar (the inliers), while being unable to generalize to data points that are different (the outliers). In this way the *reconstruction error* becomes an outlier score, with a small reconstruction error showing that a data point is similar to the majority of data in the training dataset and vice versa. The hidden vector of an autoencoder can additionally be used as a more compact and semantically rich representation of the data and serve as input to other simple outlier detection algorithms like the aforementioned density based approaches [CNM16].

Similar results using the reconstruction loss as an anomality score have been achieved using *Variational Autoencoders* (VAEs) [KW19], although these two architectures differ fundamentally insofar that autoencoders learn a compressed represen-

tation of the input data while Variational Autoencoders learn the parameters for a probability distribution representing the data and are therefore *generative models*. The other popular generative model architecture - Generate Adversial Networks (GANs) - have also been used to predict anomalies through their loss [Di +19].

Most other deep learning technique require some labelled data for their learning signal. This can either be labels for only the inliers (mostly meaning a clean dataset of only inliers) for the deep one-class case or semi-supervised (weakly-supervised) cases. These are therefore being discussed in chapter 4.1.

3.1.4 Clustering

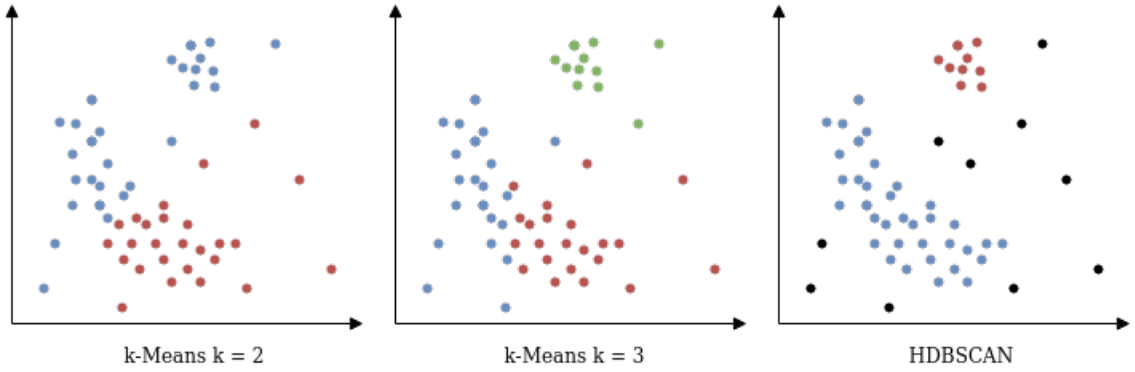


Figure 19: k-Means with k=2 fails to find the outlier cluster in the top right. k-Means with k=3 correctly finds the top right cluster but splits the bigger cluster. HDBSCAN correctly identifies both clusters and can also classify all low density points as not being in any cluster (outlier points).

Unsupervised anomaly detection as a step in data analysis is closely related to *cluster analysis*. Distance based algorithms like the k-Means clustering algorithm [Llo82] work under the *prototype assumption* that a finite amount of data points can serve as prototypes to represent the data well [Ruf+21]. K-means works by randomly initializing k points to different clusters and then assigning every other point to the nearest of these k points. Then the mean point in the vector space for every cluster is taken as the new point of reference and all points are assigned to the mean they are closed to, which happens again iteratively until data point assignment doesn't change anymore. Due to it's random nature and the fact that the number of clusters k is usually unknown in advance, k-means has to be run for

different initializations and k values to take a useful average. Naively using k = 2 to obtain an inlier and outlier cluster will fail in most cases (see figure 19). However, importantly, a successful clustering with a number of clusters higher than 2 can in many cases be a preferable way to analyze data, as it allows handpicking the clusters of data that a domain-expert deems inlier or outlier data.

Density-based clustering algorithms, like the density based outlier detection algorithms described in chapter 3.1.1, are able to infer clusters in the data based on local density. Commonly used algorithms are for example DBSCAN [EKX96] or the hierarchical extension HDBSCAN [CMS13] which can handle clusters of varying densities. Density based clustering has two main advantages for the outlier detection task: (1) They have an inherent concept of outliers, as data points in areas of low density are not assigned to any cluster and (2) prior estimations about a good number of clusters to split the data into does not have to be done and instead the algorithm will find a number of clusters by itself, again based on how many areas of high density it can identify.

Lastly, cluster labels can be obtained in an end-to-end deep learning framework. As an example, Zhou et al., 2019 [ZCZ19] showed how to do this for document data without labels: The training is done in a contrastive manner using a siamese network that returns a normalized vector representing probability for cluster membership for a data point as output. Pseudo labels are obtained by sampling sentences from the same paragraph as positive pairs and sentences from different paragraphs as negative pairs.

3.1.5 Dimensionality Reduction

Dimensionality reduction techniques can be an important part of an outlier detection pipeline, as they can make feature vectors gained during feature extraction easier to process for outlier detection algorithms that struggle with high dimensionality. This can for example be the case when the feature vectors are highly dimensional and the *curse of dimensionality* makes proximity-based methods fail to distinguish between the distances between data points or computational cost for highly dimensional data becomes too high. As previously mentioned, an algorithm like *PCA* is able to remove correlation from features and make each dimension more relevant and higher in variance among the data. *Autoencoders* on the other hand are able to find nonlinear combinations of the input features to build a new compressed hidden vector that should be more semantically rich.

Another popular dimensionality reduction algorithm is *Uniform Manifold Approximation and Projection* **UMAP** [MHM18] which, as the name suggests, learns a a manifold and seeks to accurately represent local structure as well as preserving global structure. This technique's theoretical foundation is Riemann geometry and works with three assumptions: The data is uniformly distributed on a Riemann manifold, the Riemann metric is locally constant and and manifold is locally connected. From this it is possible to compute a weighted k-neighbours graph and from this graph compute a low dimensional projection of the data by searching for a fuzzy topological structure that is as close as possible to the original's structure.

Empirically it has been shown that using that UMAP can improve the results of outlier detection [4] as well as drastically improve clustering results for different kind of algorithms, such as the aforementioned *K-Means* and *HDBSCAN*, when used alone [AKC20] or if used in a pipeline after reducing the dimensionality first with an Autoencoder [McC+19]. This is interesting for outlier detection as these algorithms rely on the same concept of proximity and density as common outlier detection algorithms and can therefore expect the same kind of performance improvement from UMAP.

When compared to the popular **t-SNE** (t-distributed Stochastic Neighbor Embedding) [MH08], which is commonly used for visualization, UMAP has advantages in computation speed for larger datasets and while both are good at preserving local structure, it arguably is able to preserve more global structure. Lastly, the **ivis** algorithm [Szu+19] was used in this project. It again claims good performance on large datasets and to preserve both local as well as global structure. Ivis created a lower dimensional representation of the data by training in a contrastive manner with a siamese network and the triplet loss, optimizing to reduce the distance between a point and its nearest neighbours while maximizing distances to points that are not nearest neighbours. This can be done in an unsupervised, semi-supervised or fully supervised manner.

3.2 Methodology

This empirical section will focus on testing unsupervised outlier detection for purely textual documents by focusing on telling apart two similar datasets, one of which occurs only with a small fraction in the training data. The code for all experiments

[4] Qualitative analysis https://umap-learn.readthedocs.io/en/latest/outliers.html.

can be found on GitHub: github.com/pschonev/deepanomaly4docs.

3.2.1 Pipeline

The used pipeline for the experiments (as shown in figure 20) is a classic combination of feature extraction and outlier detectors. The text data - after the pre-processing step - is first fed to one of the possible feature extractors. This can either be a Doc2Vec model (pre-trained or self-trained [LB16], loaded with gensim [ŘS10]) or the word2vec based RNN or Pooling models or a pre-trained Transformer model from HuggingFace [Wol+20], both loaded with flair [Akb+19b]. Transformer models are always used in their pre-trained form as a feature extractor with no amount of fine-tuning taking place, due to the computational costs involved. The feature vectors can then optionally be reduced in size by a dimensionality reduction algorithm, in the hopes of making the task for the final outlier detection algorithm easier. The majority of outlier detection algorithms will then assign each data point a binary result based on it being predicted as an inlier or outlier.

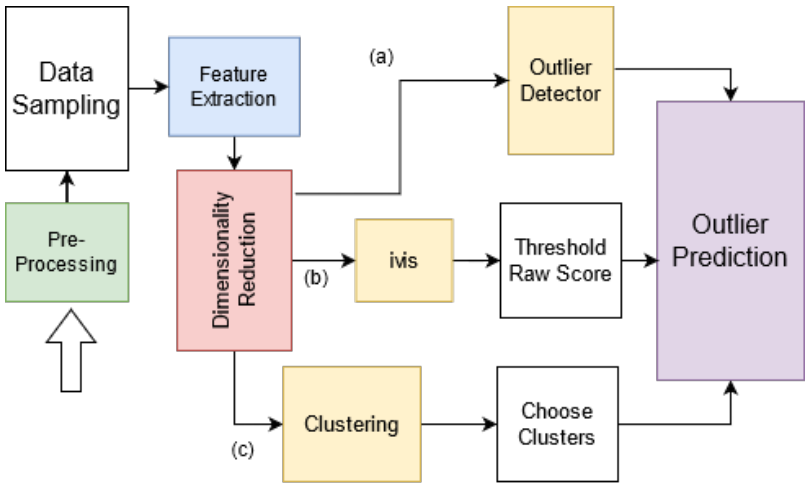


Figure 20: Unsupervised pipeline. After extracting embeddings from a text model and reducing them, there are three alternative ways to get outlier predictions: (a) are traditional outlier detection algorithms or Autoencoders, (b) using ivis to reduce dimensionality to 1 and declaring the values outside a certain range as outliers and (c) clustering and handpicking the clusters with outlier data.

When reducing the dimensions to 1 using a dimensionality reduction algorithm (or rather the combination of multiple) like UMAP, autoencoders or ivis, the resulting

score is not normalized to be between 0 and 1 automatically. The assumption is made that outliers lie at either end of the value spectrum in one dimension. Therefore $\frac{c}{2}$ of the data are labelled as outliers on both the high end and low end of all the obtained integer values across the dataset, with c denoting the fraction of estimated contamination of the dataset (this is a necessary hyperparameter for many outlier algorithms to work).

3.2.2 Data

Name	n	Format	a len	med len	Description
20 Newsgroup[5]	20k	Text	503	1242	news articles divided into 20 different categories
IMDB Reviews[6]	50k	Text	970	1309	user submitted IMDB reviews
Amazon Reviews[7]	233,000k	Text	367	414	user submitted Amazon product reviews
Wikipedia[8]	6̃,000k	Text	741	2174	text body of all English Wikipedia articles as of 2020
All the News[9]	2,700k	Text	2746	3958	news articles from 27 American publications between 2013 and 2020
RVL-CDIP[10]	400k	Image	751	1080	greyscale images divided into 16 classes of documents

Table 1: Overview over all used datasets (in unsupervised and supervised).
n is the number of documents in the dataset, *alen* and *med len* are the average and median document length.

[5]http://qwone.com/∼jason/20Newsgroups/
[6]http://ai.stanford.edu/∼amaas/data/sentiment/
[7]https://nijianmo.github.io/amazon/index.html
[8]https://dumps.wikimedia.org/
[9]https://www.kaggle.com/snapcrack/all-the-news
[10]https://www.cs.cmu.edu/∼aharley/rvl-cdip/

The experiments use two datasets of medium length documents that are in purely textual form for testing: Stanford's *Large Movie Review Dataset* [Maa+11] which contains 50.000 user-submitted IMDB movie reviews and contains binary labels for sentiment (which will not be used), as well as the *20 Newsgroups* dataset with 20.0000 short news articles divided into 20 categories. A dataset of all English *Wikipedia* articles as well as a another large news dataset (*All The News*) which contains another 2.7 million news articles between 2013 and 2020 are (partially) used to train custom Doc2Vec models.

A subset of the data is randomly sampled for training and test data for each new time a model is trained, averaging the results over all runs (also known as repeated random subsampling or Monte Carlo cross-validation). The same subsets of the data are used across all models and parameter combinations, to allow for comparability of the results.

3.2.3 Pre-Processing

For the Doc2Vec model, the text is first converted to all lower case, as the pre-trained models and all models trained during this project were trained with lower case training data. Secondly, tokens are defined as maximal contiguous alphabetic characters (without digits) and out of these tokens, only the ones with a length between 2 and 15 are kept (as implemented in gensim [ŘS10]). Stop words are not removed, however infrequent words are implicitly ignored due to the minimum word occurrence hyperparameter during Doc2Vec model training (set at 20).

For the other models implemented in the flair library (Transformer, Word Embedding RNN and Pooling), the text is optionally converted to lower case, according to the model's specifications. Then the default tokenization is used, splitting the text into word and symbol tokens.

3.2.4 Metrics

The outlier detection algorithms return binary labels 0 and 1. That means during evaluation there are four possibilities for any given data point: It can be correctly labelled as a *True Positive* or a *True Negative* or there can be a mismatch between label and prediction, which is either a *False Positive* or a *False Negative* - see table

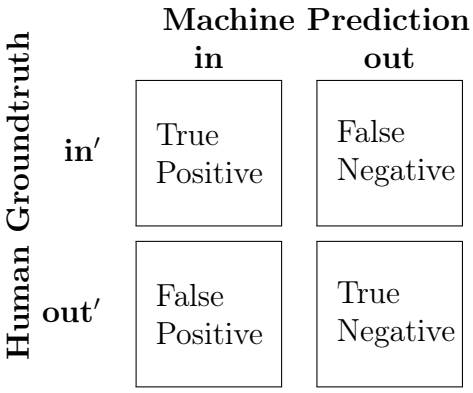


Figure 21: Confusion matrix for inliers as the positive case.

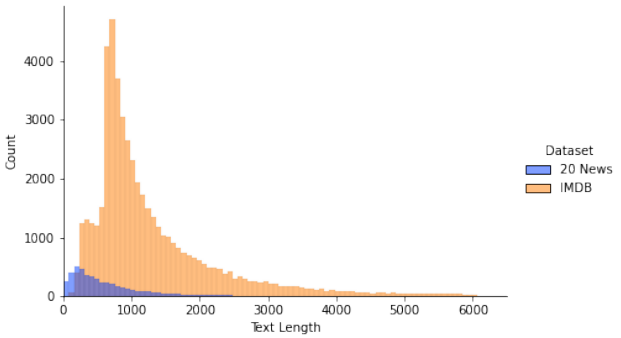


Figure 22: Document length by characters for the two used datasets IMDB and 20 Newsgroup.

21. The success is then measured by calculating

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

which is higher, the more positives got predicted as positives, and

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

which is higher, the more data points predicted as positive are actually positives. The *F1-score* is then the harmonic mean between the two and indicates overall prediction performance on a scale from 0 to 1:

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

The F1-score is preferred over a simple measure of successful classifications divided by all samples (accuracy) due to it consisting of the interpretable parts Recall and Precision and due to the harmonic mean penalizing either one being very low to obtain a more realistic measure of performance.

Because the dataset in outlier detection is highly imbalanced with outliers making up only a small amount of the data, it is important to consider the F1-score for the inliers or outliers separately and consider the *F1-macro*, which is the arithmetic mean of their F1-scores.

3.3 Results

This section contains all the results on different text embeddings and outlier detection techniques when working without any labels in an unsupervised setting, separating two datasets with a high imbalance to model inliers and outliers.

The experiments show the performance when using different models for **document representation** as input to a multitude of previously discusses **outlier detectors**. Additionally the effect of **dimensionality reduction** in the pipeline was tested. These three variables and the respective algorithms' parameters were tested extensively to find the combinations with the highest performance. Lastly, a comparison of the binary unsupervised classification and **clustering** was made.

3.3.1 Document Representations

Feature vectors extracted from different models and through different algorithms had a significant impact on the outlier detection performance. As can be seen in figure 23, using RNNs with different word embeddings as input failed to achieve good results that were significantly above the baseline of 0.5. Conversely a simple average pooling over GloVe vectors proved to be a much stronger baseline algorithm, achieving many results above 0.7. The Longformer features (as the best representative for pre-trained Transformer models) proved to only provide average results. However Transformers do have the most theoretical space for improvement given more computational power to use both a larger Longformer architecture and fine-tuning it on the training data and should therefore not be so easily dismissed. When it comes to UMAP reduction of these vectors (see figure 53), both Longformer and Glove Pool could perform better when they were reduced to lower dimensions. Remarkably, Longformers had the worst average performance across all combinations when reduced to only 300 dimensions while Glove Pool performed well without UMAP.

Doc2Vec models were able to achieve the best performances, however the performance was heavily dependant on which training data was used. This specific task of seperating IMDB reviews and 20 News data was best achieved when pre-training on the AP News or All News datasets. In fact for most outlier detection algorithms pre-training on all of Wikipedia performed very badly, with the big exception of OCSVMs, which handled both news data and Wikipedia for training Doc2Vec almost equally well (see figure 27).

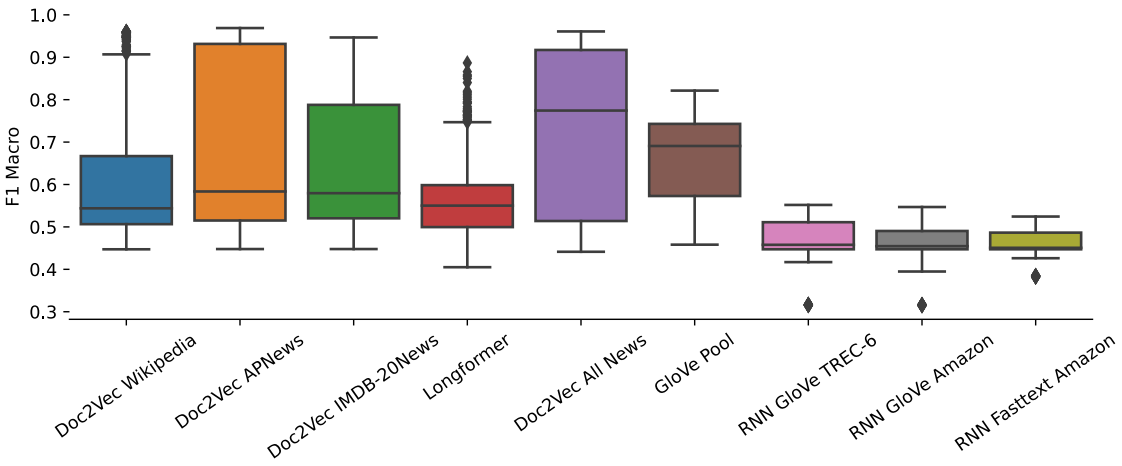


Figure 23: Vectorization models for unsupervised detection.

Figure 24 shows the performance of training Doc2Vec models on different datasets, data amounts, epoch counts and minimum word occurences m for a word to be considered during training and compared their performance to the pre-trained Wikipedia and AP News models. Several observations could be made: Decreasing m increases training time severely but actually decreases performance for outlier detection. Training for more epochs is generally good, when the amount of data is enough, can however fail completely on datasets that are too small (compare training only on the IMDB and 20 News data for 10 or 100 epochs). More data is only useful if it is the right data (presumably similar to the data in the outlier detection task). The Doc2Vec model pre-trained on all of English Wikipedia actually performs quite badly on its own, and so does training on only IMDB and 20 News, which apparently is not enough data to generalize well. Combining the IMDB and 20 News data (which is specific to the problem) and Wikipedia data (to generalize better) however gives good performance. Figure 25 shows that performance during training of a Doc2Vec model using half of the All News dataset becomes stable after a couple of epochs.

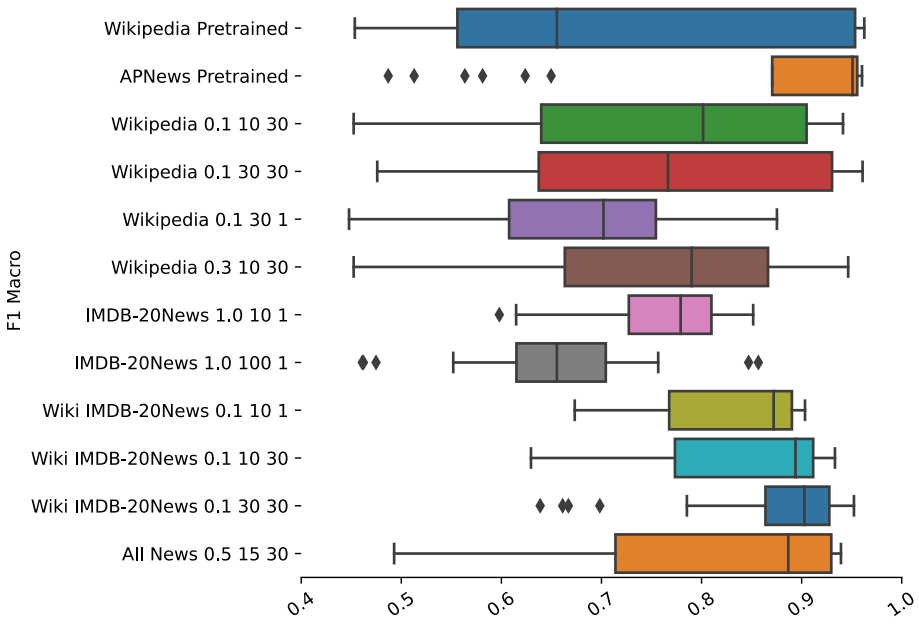


Figure 24: Unsupervised results for self-trained Doc2Vec models. Results with HBOS and OCSVM. Numbers describe the fraction of the dataset used, training iterations and minimum word count parameter setting.

3.3.2 Outlier Detectors

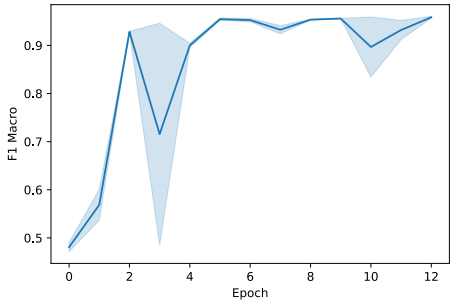


Figure 25: Performance progress on unsupervised task for each epoch of training Doc2Vec model. Stabilizes after a few epochs.

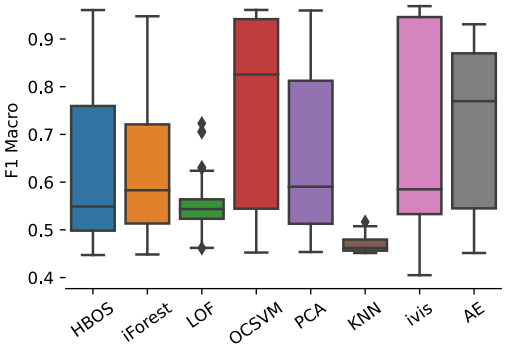


Figure 26: Outlier detectors for unsupervised detection.

The outlier detection algorithms were tested across multiple different settings for possible text embeddings and n-dimensional input vectors, with a reduction to n dimensions using UMAP. An average over all test settings can be seen in figure 26: It is obvious that the simplest algorithm - k Nearest Neighbour - is not usable, as it consistently performs under the baseline of 0.5 for the F1 macro score. Local Outlier Factor (LOF) has similarly bad performance, however can outperform the baseline given the right embeddings. Interestingly both of these algorithms work best when using GloVe embeddings as input (see figure 27) and no prior dimensionality reduction (see figure 52), both of which is not true for any other outlier detection algorithm.

The classic algorithms HBOS, iForest and PCA were all able to achieve decent performances with occasional F1 macro scores at the top for specific combinations. They are however outperformed by OCSVM. OCSVM was not only able to achieve some of the best F1 macro scores but was also most consistent among different Doc2Vec models (figure 27 as well as dimensionality of input vectors (figure 52).

Lastly the two deep neural network based architectures Autoencoder and ivis were able to achieve performances on par with OCSVM. Autoencoders were generally more stable across different settings with both having consistently very good performance when given a Doc2Vec model pre-trained on the most fitting data for to separate the IMDB and 20 News datasets. The combination of UMAP and ivis was able to

achieve the majority of the absolute best F1 macro scores across all combinations (see table 2). Notably, ivis never achieved good results using the GloVe pooling feature vectors or the Transformer features (Longformer).

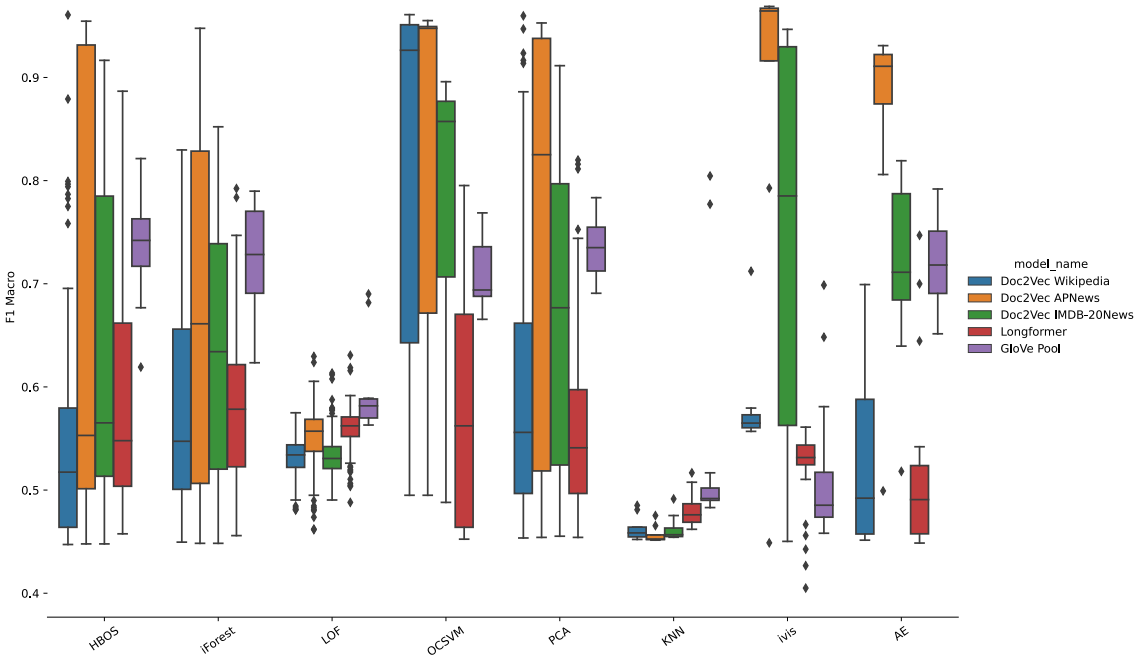


Figure 27: Unsupervised outlier detector and model combinations.

3.3.3 Dimensionality Reduction

With the two exceptions of LOF and KNN, all outlier detection algorithms worked better if the feature vectors were not fed raw but UMAP dimensionality reduction was used on them first (see figure 52). In general reducing to lower dimensions proved the most useful, as most algorithms had a better performance when doing so. However the best overall results were achieved when using UMAP only to restructure the feature vectors slighly (keeping 300 dimensions or reducing only to 200) and then using ivis (which is a dimensionality reduction algorithm itself but uses non-linear neural networks).

Using t-SNE for dimensionality reduction gives bad results, as can be expected from an algorithm that is only built to handle reduction to 2 or 3 dimension for visualization purpose (see figure 28). PCA is a little more successful but could not match the performance of the more recent algorithms UMAP and ivis. Lastly, using a combination of UMAP and ivis was able to achieve the highest results (given the right Doc2Vec model).

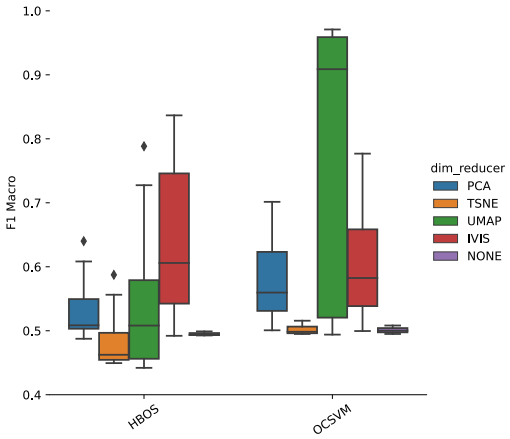


Figure 28: Results for dimension reduction algorithms when used in conjunction with HBOS and OCSVM.

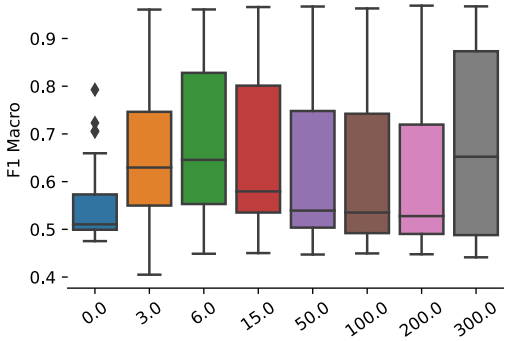
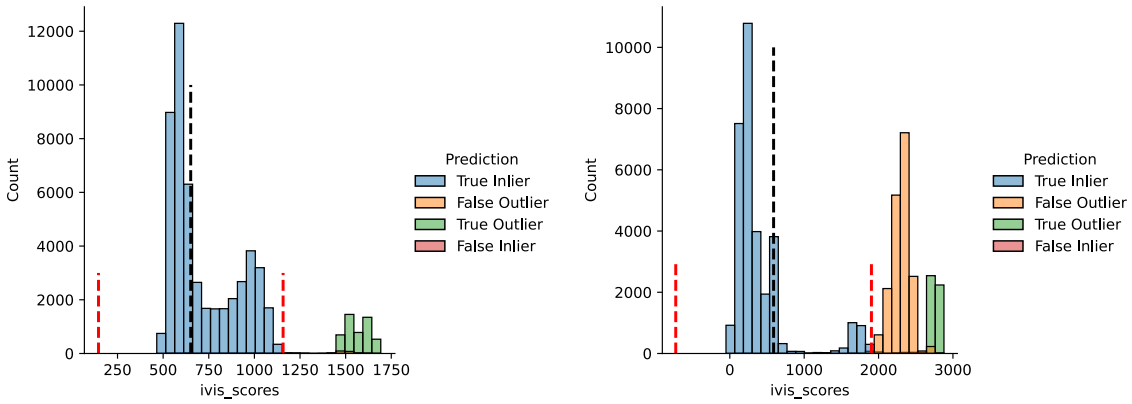


Figure 29: Unsupervised results for UMAP Reduction to n dimensions. n = 0.0 means no reduction was applied.

Rank	Doc2Vec Model Training Data	Outlier Detector	n_{UMAP}	$f1_{macro}$
1	AP News	ivis	200	0.968
7	All News	ivis	200	0.949
9	IMDB-20News	ivis	50	0.946
12	All News	OCSVM	6	0.934
14	AP News	AE	15	0.930
23	All News	iForest	6	0.919
24	All News	PCA	6	0.914
35	All News	HBOS	6	0.867

Table 2: Best unsupervised combinations of the different Doc2Vec models, outlier detectors and UMAP reduction to n dimensions. Tested with a contamination of 0.1 on the IMDB - 20 News split for 3 sampling runs. Only the first combination of a model and outlier detector combination are shown, ignoring duplicates of different n_{UMAP}. In total 300 combinations are possible.



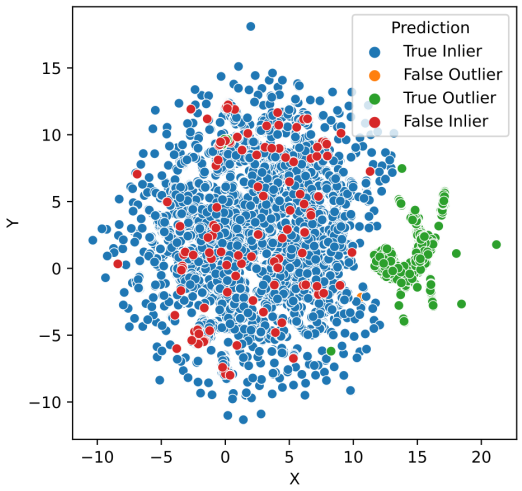
(a) (b)

Figure 30: Raw scores after applying UMAP and ivis to reduce Doc2Vec vectors to one dimension. (a) is using Doc2Vec trained on APNews data (b) and on Wikipedia data.

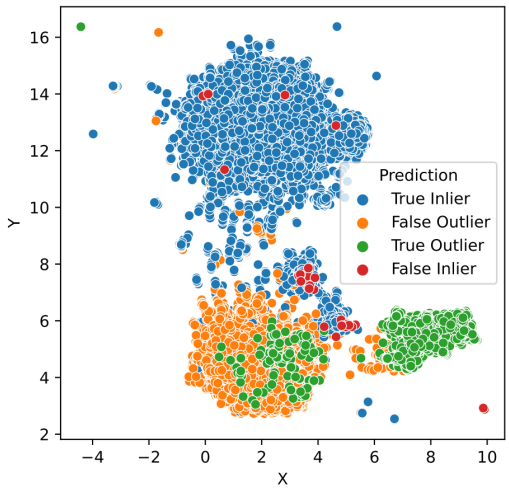
3.3.4 Clustering

As mentioned before, clustering can be a more flexible alternative to analyze data, which does however require manually looking at the resulting clusters and assigning them to in or outliers. When comparing the resulting scores in figure 30a of the pipeline using Doc2Vec, UMAP and finally ivis to reduce extract a single outlier score, it's evident that for both the Doc2Vec model trained on AP News as well as on Wikipedia, the outlier scores are successfully pushed to one end of the scale. Nonetheless, when taking a range (such as the interdecile range) to detect outliers in these scores, the AP News based model performs much better. That is because the Wikipedia model forms a multimodal distribution over the inliers, which makes it so that the median is not clearly defined at one point and neither is the interquartile range. Additionally to define an a fitting range, one has to make assumptions about the contamination of the dataset.

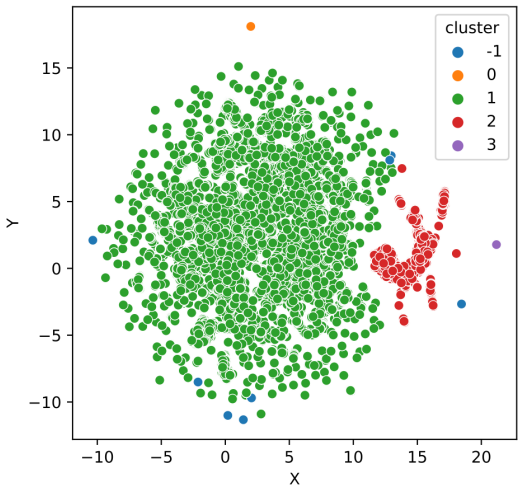
Figures 31a and 31b show how how the Doc2Vec vectors extracted from the model trained on AP News and Wikipedia differ when processed with UMAP. The Wikipedia model seems to actually have found multiple clusters in the inliers while the AP News model only differentiated between the inlier and outlier documents. When using a clustering algorithm such as HDBSCAN on UMAP reduced vectors (to 256 dimensions) however (figures 31c and 31d), the outlier cluster emerges and can



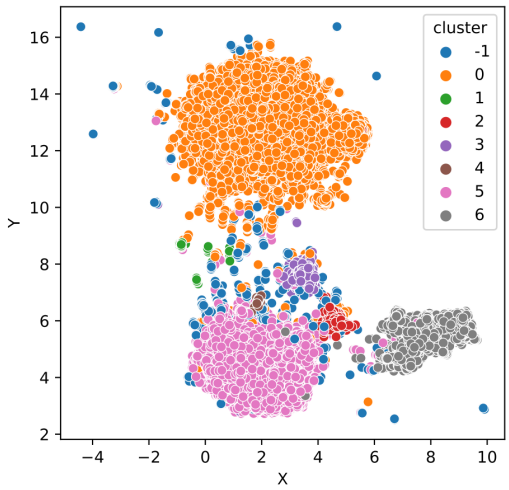
(a) AP News - ivis: $F1_{macro}$ 0.9766



(b) Wikipedia - ivis: $F1_{macro}$ 0.564



(c) AP News - HDBSCAN: $F1_{macro}$ 0.9762

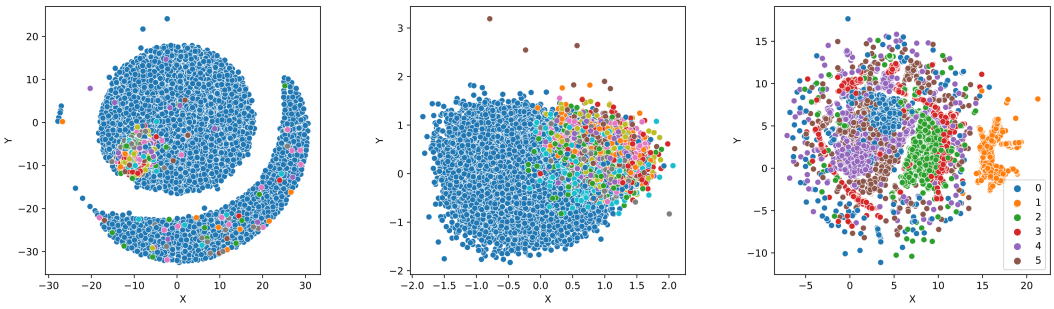


(d) Wikipedia - HDBSCAN: $F1_{macro}$ 0.9732

Figure 31: 2D visualization of the IMDB - 20 News dataset using either vectors from Doc2Vec pre-trained on Wikipedia or AP News dataset, applying UMAP and then using either ivis to obtain an outlier score or HDBSCAN to obtain clusters. Visualization in 2D with UMAP. F1 scores for HDBSCAN clusters are calculated by picking the proper clusters by hand.

easily be picked out, to get a result competitive with the best scores in the previous section. Using the more simple (and faster) k-Means algorithm does achieve the same result here (figure 32c).

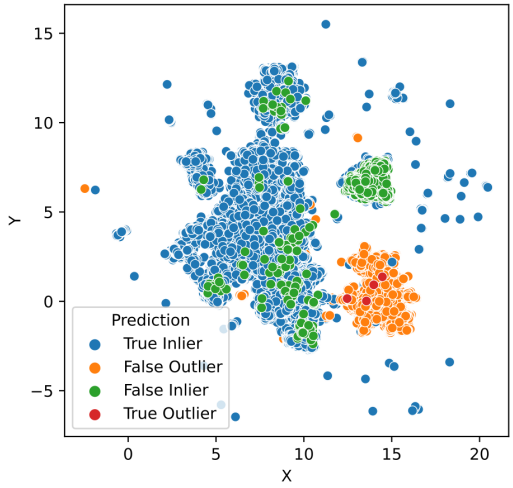
There are some notable ways this failed completely though. Running HDBSCAN or k-Means on raw Doc2Vec extracted vectors did not result in any meaningful clusters. This means that the Doc2Vec vectors profit immensely from dimensionality reduction. Secondly however, using PCA in combination with HDBSCAN also proved unsuccessful. While the combination of PCA and k-Means was successful, UMAP vectors proved the best for visualizing the data in 2D, with a much cleaner separation of the outlier data and likely a better separation within the inlier data (although this has not been tested) (figures 32a and 32b).



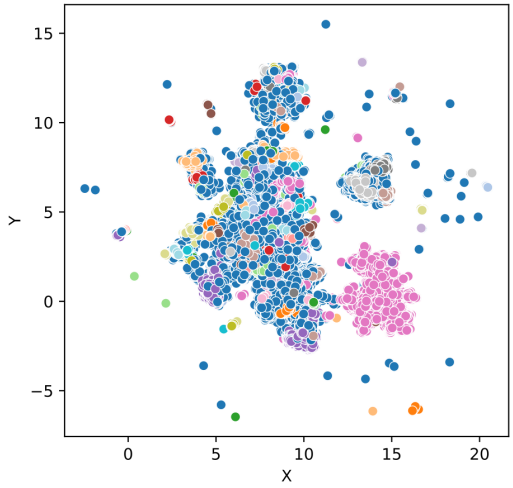
(a) t-SNE (b) PCA (c) UMAP - k-Means

Figure 32: Comparison of 2D vectors when applying the different dimensionality reduction algorithms t-SNE, PCA and UMAP. Doc2Vec vectors trained on AP News. 2D Visualization of the IMDB-20 News dataset with labels for all different classes. (a) Shows the 2D reduction using t-SNE and (b) using PCA. (c) Shows the results using UMAP to reduce to 256 dimensions and applying k-Means clustering, visualized on 2D UMAP vectors. Choosing the first k-Means cluster gives an $F1_{macro}$ of 0.9769.

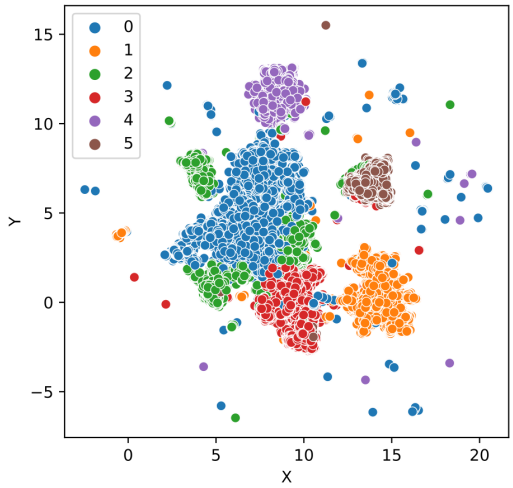
These results manifested again using a new dataset with Wikipedia articles as inliers and Amazon reviews as outliers. This time the UMAP - ivis pipeline failed to detect the relevant part of the dataset as outliers completely. Using UMAP and HDBSCAN resulted in over 100 clusters (on a dataset with 50.000 data points), from which three could be easily picked to get a good $F1_{macro}$ of 0.898. However again k-Means outperformed this, finding the outlier data reliable within one cluster both on UMAP and PCA reduced vectors for a significant boost of the $F1_{macro}$ to 0.962.



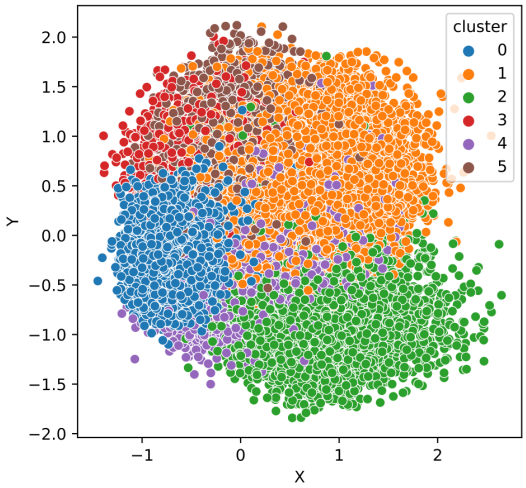
(a) ivis: $F1_{macro}$ 0.476



(b) HDBSCAN: $F1_{macro}$ 0.898



(c) k-Means: $F1_{macro}$ 0.9623



(d) PCA - k-Means: $F1_{macro}$ 0.9623

Figure 33: 2D visualization on the Wikipedia-Amazon outlier dataset. Using UMAP and then (a) ivis, (b) HDBSCAN or (c) k-Means. (d) Shows using PCA and then k-Means. For HDBSCAN and k-Means clusters were handpicked to calculate score.

The conclusion from this qualitative analysis is as follows:

- UMAP or PCA are necessary to make Doc2Vec extracted vectors work properly, while UMAP is preferable since it gives a much clearer visualization in 2D to analyze the results of the clustering. This is in line with the results of [AKC20], showing a big performance boost for clustering using UMAP on a variety of datasets.
- k-Means proved to be more more capable and flexible than HDBSCAN on this specific task and using Doc2Vec vectors, while also being considerably faster.
- Clustering seems like the most robust solution to analyze unlabelled datasets and check for outliers. Especially when these outliers might come in a variety of forms and form different clusters.

4 Supervised Anomaly Detection

4.1 Background

(Semi)-supervised anomaly detection can work with labels which allows for a user made decision which data points are considered inliers and outliers. Crucially, this means that within complex multimodal datasets such as the RVL-CDIP dataset for documents from a variety of 16 different classes, outliers within a specific class can be declared inliers within the context of the whole dataset (meaning all data points with a certain class label are declared as inliers). The classifier can thus learn to better handle a variety of different documents, even if they occur rarely in the training dataset (also known as novelties). Secondly, because the assumption that the majority of similar looking data are *inliers* while a small portion of the data which looks dissimilar are *outliers* can be lifted, as supervised anomaly detection models can attempt to build new representations of the data, such that the labelled inlier data is close to each other in the new vector space and a tight *decision boundary* can be drawn around the inliers.

Supervised anomaly detection can be considered a type of binary classification task with a heavy class imbalance. Many ideas of successful algorithms to solve this problem resemble what is called zero-shot or few-shot learning in literature, meaning a classifier is supposed to correctly label classes at test time, that have no or very few samples in the dataset during training time. Because the definition of what *inliers* look like is clearer in a supervised manner of training, such an outlier detector can better be employed as part of the production pipeline to determine whether new data points are *inliers* or *outliers* (similar to other supervised classifiers).

4.1.1 Fully Supervised

The fully supervised setting in outlier detection is the rarest case and usually cannot be considered realistic. While there can be datasets with a lot of labelled inliers and outliers, most classic supervised classifiers will fail to generalize on most available datasets. That is because every data point which is unlike the inlier data is considered an outlier, while the outliers in the training data can only serve as an example of a small part of this general "outlier class". Nevertheless there are several techniques to infer good outlier prediction during supervised classification: A simple baseline can be set by simply training a deep classifier in an unsupervised way as usual. The last

layer is then usually a softmax that assign the probability the classifier gives for each class as a prediction. Intuitively, if a data point does not look like any of the classes the classifier was trained on, none of the probabilities should be particularly high. Therefore one could introduce a **threshold** and consider every data point that fails to have a predicted probability higher than the threshold for any of the classes an outlier [HG18]. Building up on this idea [DT18] propose adding an auxiliary branch to an existing classifier network that explicitly returns an anomaly score between 0 and 1 that can then be thresholded.

A related promising idea to improve results of a supervised classifier is **Outlier Exposure** proposed by [HMD19]. The idea behind OE is that vast amounts of data are nowadays freely available for training. Therefore the multiclass classifiers with the two aforementioned techniques to detect unseen outliers can improve their results by simply leveraging this extra data and introducing it to the network as outliers during training. The same idea can also work for outlier detector networks that do density estimation.

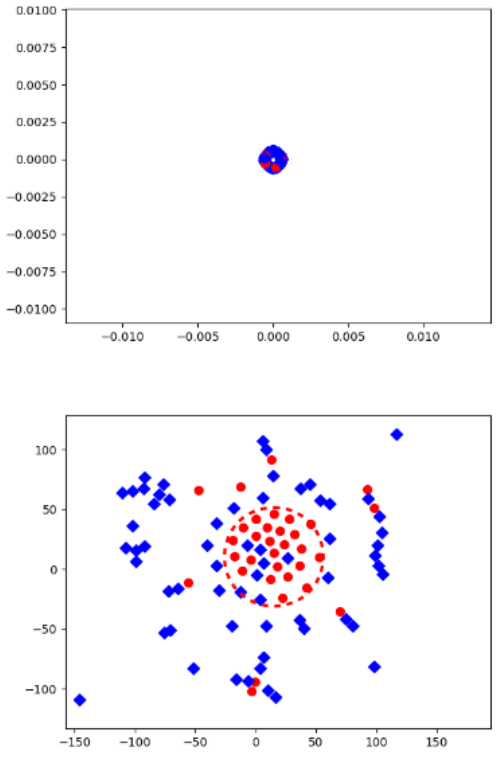


Figure 34: Collpasing One-Class

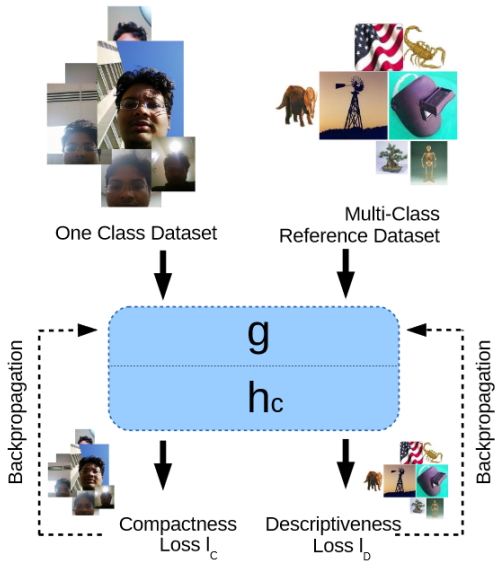


Figure 35: One-class classification by using the Compactness Loss and Descriptiveness Loss.

4.1.2 One-class Classification

One-class classification aims to draw a tight decision boundary around the inlier data. Naively a network could therefore be trained to represent inlier data points as close to each other as possible, as to be able to classify everything outside the boundary as outliers. This can be done by defining a *Compactness Loss* which computes the Euclidean distance of output vectors within each batch, as to minimize it and is equal to the learning objective of the classic *Support Vector Data Description* (SVDD) which attempts to find the minimum hypersphere containing all inlier samples. However usually such training (even if only done as fine-tuning on an existing multiclass classifier) in a deep learning setting will lead to all of the output vectors collapsing into a single point, which would trivially minimize the loss term. To avoid this, [PP19] propose using both the Compactness Loss as well as a *Descriptiveness Loss* to train the same network (see figures 34 and 35). The Descriptiveness is a simple cross-entropy loss as would be commonly used for a classifier, punishing misclassification of data points. In addition to the inlier data (one class dataset), the training dataset therefore has to contain a labelled multiclass reference dataset. The idea during training is then to alternate between training on batches of the one-class data, using the network's branch with n features as output and the Compactness Loss, and batches of the multiclass dataset, using a branch that ends in a softmax function and uses the Descriptiveness Loss. The resulting network with the branch for the Compactness loss can then be used during inference to output a feature vector of size and with the property that all inlier data vectors lie together as close as possible in the vector space while still being able to differentiate between data points due to the influence of the Descriptiveness Loss preventing a trivial solution. After training any simple supervised classifier such as an SVM can be used to do binary classification and obtain an anomaly score. Importantly the reference dataset does not have to be from the same dataset but can be a related dataset of the same type of data which has class labels.

There have been multiple other attempts to solve the one-class problem using neural networks. Similar to the above example, Ruff et al., 2018 create a Deep SVDD and prevent the hypersphere collapse with architectural constraints [Ruf+18] [Ruf+20b]. The approach to train a deep SVDD has also been proven to extend well to the use case of text documents, using word embeddings and self-attention to create context vectors which are also highly interpretable [Ruf+19]. Chalapathy et al., 2019 on the other hand use a simple feed forward network on top of features extracted from an autoencoder, trained to find a hypersphere that separates the (now learned)

feature vectors from the origin, as is the learning objective of the OC-SVM [CMC19]. Lastly one-class model performance can be increased by using a few labelled outlier points [Ruf+20a] [Ruf+20b] or by training on auxiliary tasks using pseudo-labels (such as predicting image rotation) [Hen+19].

There have been several attempts to solve the problem of anomaly detection using deep contrastive learning. This is typically done by using a *Siamese Network*: The same network (with the same weight parameters) is used on multiple data points, trying to maximize a distance between data points from different classes (in this case inliers and outliers). Specifically the *triplet loss* is widely used: A data point is sampled as the anchor with another data point from the same class as a positive sample and from a different class as the negative sample. The loss is then defined in such a way that the distance of the representation vectors for the anchor and the positive sample are smaller than the distance between the vectors for the anchor and the negative sample.

$$\mathcal{L}(A, P, N) = max(||f(A) - f(P))|| - ||f(A) - f(N)||^2 + \alpha, 0)$$

Pang et al., 2020 [Pan+20] have implemented such a network, showing that with very few outlier samples (weakly-supervised), it is possible to learn well performing features and seperation for outlier detection.

4.2 Methodology

This empirical section will focus on testing supervised outlier detection for scanned documents using their textual and visual information. The focus is on generalizing from some known inlier and outlier data to new types of outlier data. The code for all experiments can be found on GitHub: github.com/pschonev/deepanomaly4docs.

4.2.1 Pipeline

The fully supervised pipeline (see figure 36) starts by pre-processing the data (see chapter 4.2.3) and then sampling a part of the data and splitting it into 80% training and 20% test data. Features are then extracted from the data using a pre-trained language model as discusses previously. Since Transformer models have proven to

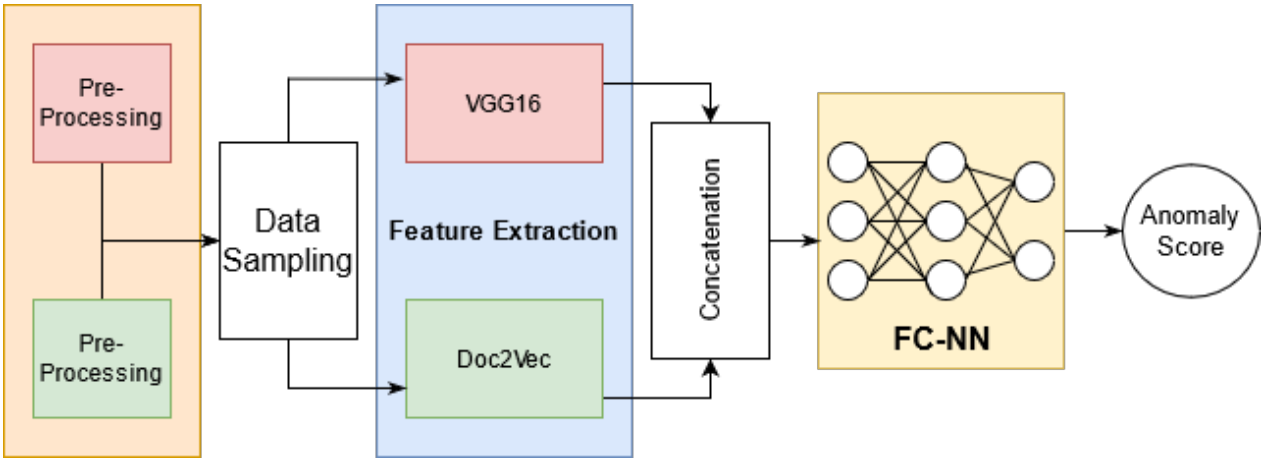


Figure 36: This shows the pipeline for supervised with optional internal view on feature extraction with late fusion (concat after extraction from the networks)

be slightly worse than Doc2Vec models but much more computationally expensive, Doc2Vec is used exclusively. Using these embeddings a fully connected neural network is then trained with the class labels for all of the data converted to 1 for inliers and 0 for outliers (according to inlier/outlier split described in chapter 4.2.2). The neural net is built using Keras [Cho+15] and uses backpropagation using Adam optimizer, binary cross-entropy loss and a dropout of 0.3.

Inspired by the success of [Bak+20] to combine textual and visual output vectors of the respective language and image models to achieve state of the art classification scores for RVL-CDIP, the pipeline can also extract features from textual and visual models and concatenate them, before doing prediction on the combined vector using the neural network. However notably, this pipeline is not going to be anywhere as powerful since it lacks fine-tuning and combined training of the image and language models and merely does feature extraction from their pre-trained state. In the case of the image model this is the VGG-16 model [SZ15] as implemented in the Keras library, pre-trained on ImageNet.

The one-class pipeline is based on the ideas in [PP19] discussed in chapter 4.1: Pre-processing, sampling and feature extraction are first done as shown in the pipeline explained above (see figure 37). Then however the data is split into inlier data and reference data (the outliers). The fully connected neural network (h1) is then however trained on the two different subsets of data with two different ending branches and their own losses. The first branch (b1) does multiclass classification using the

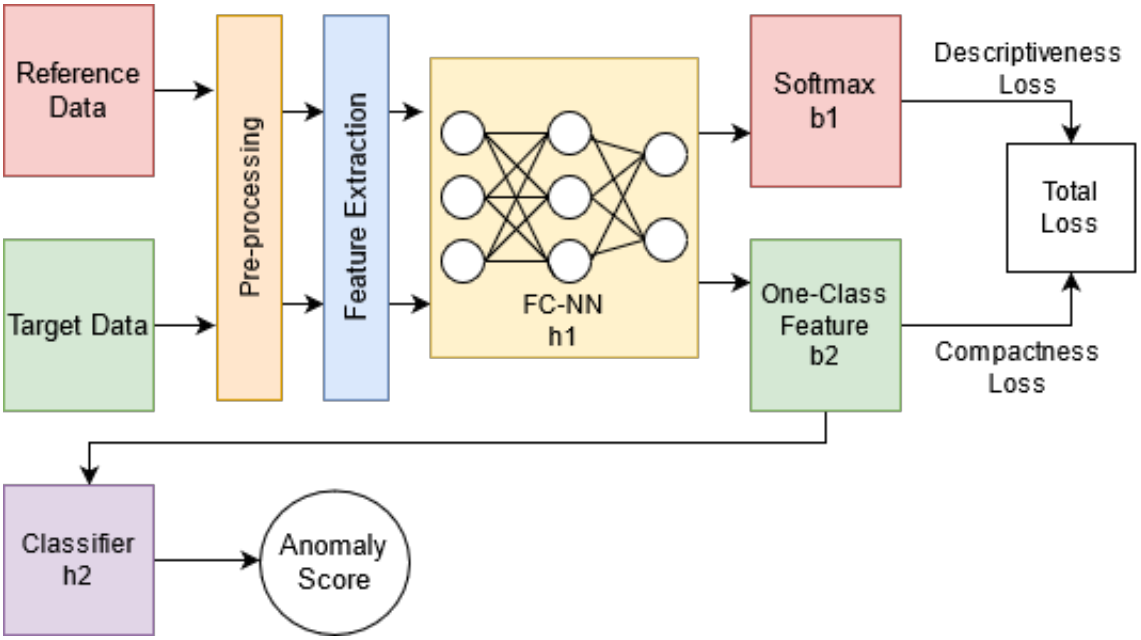


Figure 37: This shows the pipeline for one-class classification with the two different losses: h1 is the fully connected neural net used to train on both the reference data with the descriptiveness loss and the branch b1 as well as the target data and the compactness loss on the branch b2. The learned features from the b2 branch are then used to train a small classifier h2 and infer an anomaly score.

reference data, ending in a softmax function and applying a cross-entroy loss. The second branch (b2) is used when putting in the inlier data and ends in k features. The loss term is then the euclidean distance of samples within the batch, which should be minimized to represent all the inliers closely together in the final k-dimensional vector space represented by this branches output. Training is alternated between the two branches. After training the network h1, a final small fully connected neural network is trained on top of the output of b2 for inlier and outlier data points as a binary classifier to obtain an outlier score.

The sampling method for the training and test dataset is Monte Carlo cross-validation again and unless otherwise stated, the split of inliers and outliers in the test dataset is 80:20.

To test the capability of the models to generalize to new outliers, the **one-unseen** task is defined as follows: During training the model gets to see all inlier and outlier

data, except for one outlier class that is unseen during training. When testing the model is then asked to distinguish between the inlier data and this one unseen outlier class.

4.2.2 Data

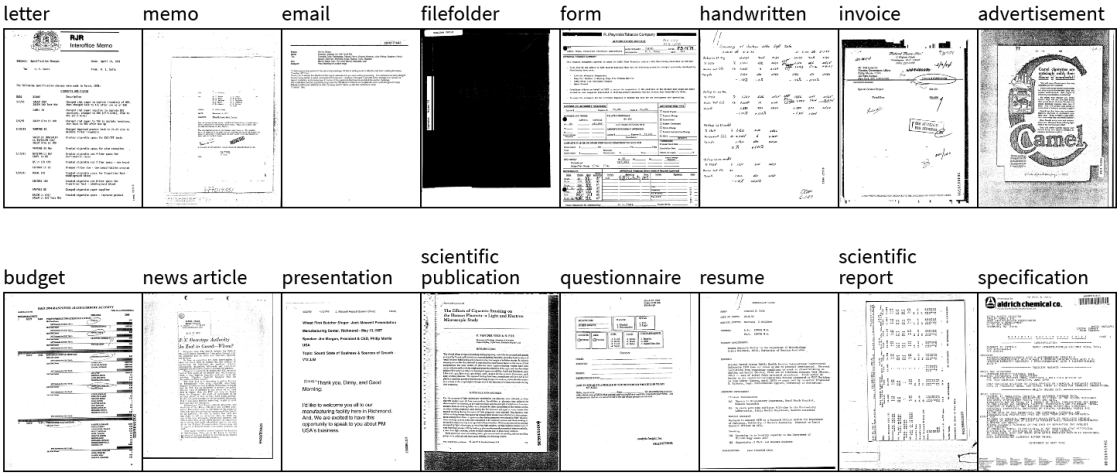


Figure 38: RVL-CDIP dataset samples for all 16 classes.

The dataset used for the supervised experiments is **RVL-CDIP** (Ryerson Vision Lab Complex Document Information Processing) [Har15]. The dataset consists of 400.000 greyscale images, evenly distributed among 16 classes. The pictures show scans of text documents from a variety of classes (see figure 38). To reflect a real life working situation, where outlier detection is used as part of a document classification and information extraction pipeline for private customer data in business situations, the dataset is split into inliers and outliers as such: Letters, emails, forms and invoices are considered inliers while everything else is considered an outlier.

4.2.3 Pre-Processing

The RVL-CDIP dataset is a dataset of images of scanned documents. Therefore, to extract text data from the images, the first processing step is to run Optical Character Recognition (OCR). The OCR service of choice was Tesseract, which is an open source software project started in the 1980s and provides OCR for more than

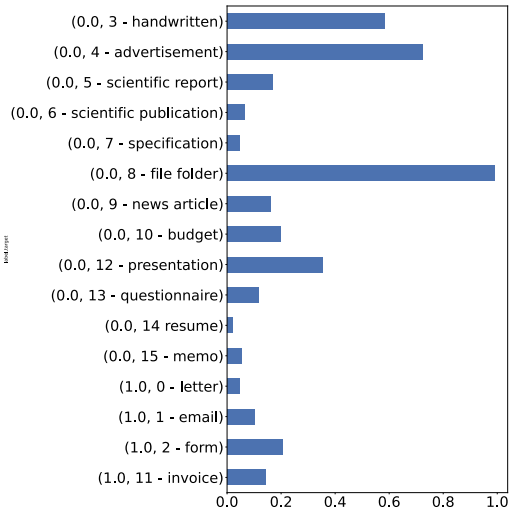


Figure 39: Document counts per class for documents under 250 characters.

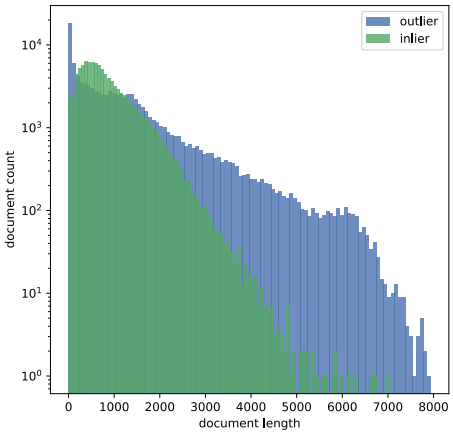


Figure 40: Documents with text length histogram

100 languages. Tesseract consists of a classic computer vision pipeline to recognize characters and recently (in 2019) added an LSTM network for line detection [tes21]. It can output text files with plain text extracted from each document as well as coordinates for each token (as are used in other document classification networks, see chapter 2.3) but only the plain text is used in this project.

The plain text output inevitably contains OCR errors, meaning single wrongly detected characters. This will negatively influence performance for extracting document embeddings from models that work on word level such as Transformers with word2vec input as well as Doc2Vec but was not attempted to be corrected. Documents were however removed from the training data when they did not contain a certain minimum amount of characters.

Because documents might not contain enough text to produce a good text representation and/or OCR can be faulty when extracting the text, it was analyzed how many characters the resulting text documents contained. As can be seen in figure 40, inlier data was mostly evenly distributed with a peak at 1000 character length. Outlier data on the other hand contains many documents with a very small amount of characters. This is mainly due to to the *file folder* class (see figure 39) containing no text at all. Similarly *advertising* and to a lesser extend *presentation* might not contain a lot of text, while *handwritten* documents have the problem that the text

cannot be extracted with Tesseract. Outlier data also has significantly more wordy documents (over 4000 characters), which almost exclusively come from the *scientific publication* class and some being *news articles* and *resumes*. Considering each of the documents will also contain OCR errors making single words unprocessable and as to not confuse the classifier/outlier detector with uninformative document vectors representing short texts or no text at all, it was decided to set a boundary for minimum text length of 250 and not use the classes *file folder*, *handwritten* and *advertisement* at all during training.

To feed the images into a pre-trained VGG16 model, they are first reduced to a size of 256x256 pixel and converted to contain 3 channels (note that they don't need 3 color channels as they are greyscale images but this is required as input). Using the Keras implemented pre-processing for VGG16 the images are then converted from RGB to BGR and zero centered on each channel with respect to ImageNet.

Also used are the *Amazon* dataset [NLM19] [11] containing user-submitted reviews taken from Amazon, as an additional outlier dataset, together with data from the previously mentioned 20 News, IMDB and Wikipedia datasets.

4.2.4 Metrics

In addition to F1 and $F1_{macro}$ scores as fixed threshold metrics, results are reported as accuracy for inlier and outlier classes. The accuracy is defined as:

$$Accuracy = \frac{True\ Positive + True\ Negative}{N}$$

Furthermore, since the model output for each data point is a score between 0 and 1, that represents the predicted probability for the data point to be an outlier, threshold independent scores can be used to compare the models across many different thresholds. The first score given is the area under receiver operating characteristic curve (**AUC-ROC**). The ROC curve is a plot of the False Positive Rate (FPR) on the x-axis and true positive rate (TPR, also recall or sensitivty) on the y-axis:

$$FalsePositiveRate = \frac{False\ Positive}{False\ Positive + True\ Negative}$$

[11] https://jmcauley.ucsd.edu/data/amazon/

$$TruePositiveRate = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

The baseline is horizontal line from (0,0) to (1,1) as can be seen in figure 54. The AUC-ROC can then be used as a score to sum up the performance of a model over many thresholds, with the baseline always at 0.5, regardless of whether the dataset is balanced or imbalanced. Conversely the precision recall curve (**PRC**) in the binary classification case has a baseline depending on the ratio of the two classes (see figure 55) [SR15]. The PRC is a plot defined by the formerly defined TPR - or recall - and precision:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

The PRC can be summed up with the average precision (AveP), which is approximately equivalent to taking the area under the curve. It is defined as the sum of the precision at each threshold, weighted by the increase of the recall compared to the previous threshold:

$$AveragePrecision = \sum_{n}^{Recall_n - Recall_{n-1}} Precision_n$$

Both AUC-ROC and AveP scores are given, however AveP is is preferred, as it has been empirically shown to reveal more problems in imbalanced datasets [SR15].

The baseline is always defined by the theoretical ZeroR classifier, which predicts every data point to be in the majority class.

4.3 Results

This section contains all the results on different ways to train with labelled data and trying to correctly detect new and unseen outlier data with the trained model.

The first thing that was tried was a **fully supervised** binary classification task, using a fully connected neural network with Doc2Vec extracted text vectors as input.

The results give insight on the difficulty of the task with the RVL-CDIP dataset. Additionally the previously discussed idea of **Outlier Exposure** was tested, giving the model additional types of outlier data. Then the simple fully supervised approach was compared contrasted with a **one-class** training method, putting more emphasis on the model to learn to represent the inlier data closely together while keeping them separate from all other possible outlier data. Lastly the effects of a **multimodal** approach with input vectors of both text and visual features were tested, as well as giving the model a few examples of "unseen" outlier classes in a **weakly supervised** setting.

4.3.1 Fully Supervised

The first set of experiments were done by training a fully connected neural network on top of textual document features (see figure 36) and treating the problem as binary classification with heavy class imbalance and the 16 classes split as described previously.

As seen in table 3, testing was done with contaminations of 0.05, 0.1 and 0.2 and the more imbalanced the dataset, the lower the performance of the classifier. The same observation is made when decreasing the amount of samples per class in the dataset n. This shows that the general rule that more training data results in better test performance holds true. Overall the classifier tends to classify samples as inliers by default and thus has better performance on inlier accuracy when the dataset is heavily imbalanced, which however shifts as the contamination gets higher.

To gain a better understanding of the data, the training architecture was used as above and the training dataset created as such: One class was chosen as the inlier class with 5.000 samples and another class was used as the outlier class for runs of varying contamination of 0.05, 0.1 and 0.15. The test dataset used the same contamination and the resulting matrix of $f1_{macro}$ scores can be seen in figure 41.

The overall arithmetic mean is 0.68, which shows that in general the classifier was able to learn a useful distinction between the two classes. Notably, most combinations were able to be separated with an above baseline performance. Some classes worked particularly badly as inliers. These are the aforementioned *handwritten*, *advertisement* and *file folder* classes, which did not contain enough text to create meaningful feature vectors with a Doc2Vec model and therefore the classifier was unable to learn properly. Conversely the classes that contained documents with par-

c	n_{class}	$F1_{macro}$	F1	acc_{in}	acc_{out}	AUC-ROC	AUC-PRC
0.05	1,000	0.611	0.945	0.966	0.224	0.740	0.958
	6,000	0.681	0.953	0.970	0.338	0.808	0.970
	30,000	0.748	0.962	0.977	0.451	0.881	0.982
	80,000	0.795	0.968	0.982	0.535	0.913	0.987
0.10	1,000	0.668	0.945	0.953	0.360	0.789	0.966
	6,000	0.727	0.954	0.961	0.467	0.848	0.977
	30,000	0.771	0.960	0.964	0.561	0.897	0.986
	80,000	0.808	0.965	0.965	0.653	0.925	0.990
0.20	1,000	0.689	0.934	0.918	0.522	0.823	0.974
	6,000	0.737	0.947	0.937	0.582	0.876	0.983
	30,000	0.773	0.953	0.940	0.676	0.914	0.988
	80,000	0.799	0.960	0.949	0.708	0.937	0.993

Table 3: Results supervised binary classification by contamination and training dataset size. c is the contamination of the dataset.

ticularly long text - *scientific publication*, *news article* and *resume* - lend themselves very well as inliers for training. Therefore a strong relationship between text length and performance as inlier data during training can be established.

However when looking at performance as outlier data, things are less clear. The low-text-class *handwritten* is one of the best classes to detect as an outlier but so is *resume*. Meanwhile *letter* and *email* have very bad performance as outliers, which is inconsequential for later evaluations as these are used as inlier classes. However so do the outlier classes *news article* or *memo*. When comparing to figure 42 it can be seen that the performance for classes as inlier or outlier tested here strongly correlates to the accuracy of their prediction accuracy when using data from (almost) all classes during training. This means that the bad performance of e.g. *memo* as an outlier will pull down overall performance for the model's prediction results.

Figure 42 shows the accuracy per class when training the model with full training datasets containing all inlier and outlier classes. As mentioned before, the accuracies for outliers correlate to the ones measured before when classifying the different classes pairwise. Notably, all inlier classes maintain accuracies of over 90% as would be expected for a classifier dealing with a highly imbalanced dataset.

The next experiment shows how well the model was able to **generalize from one**

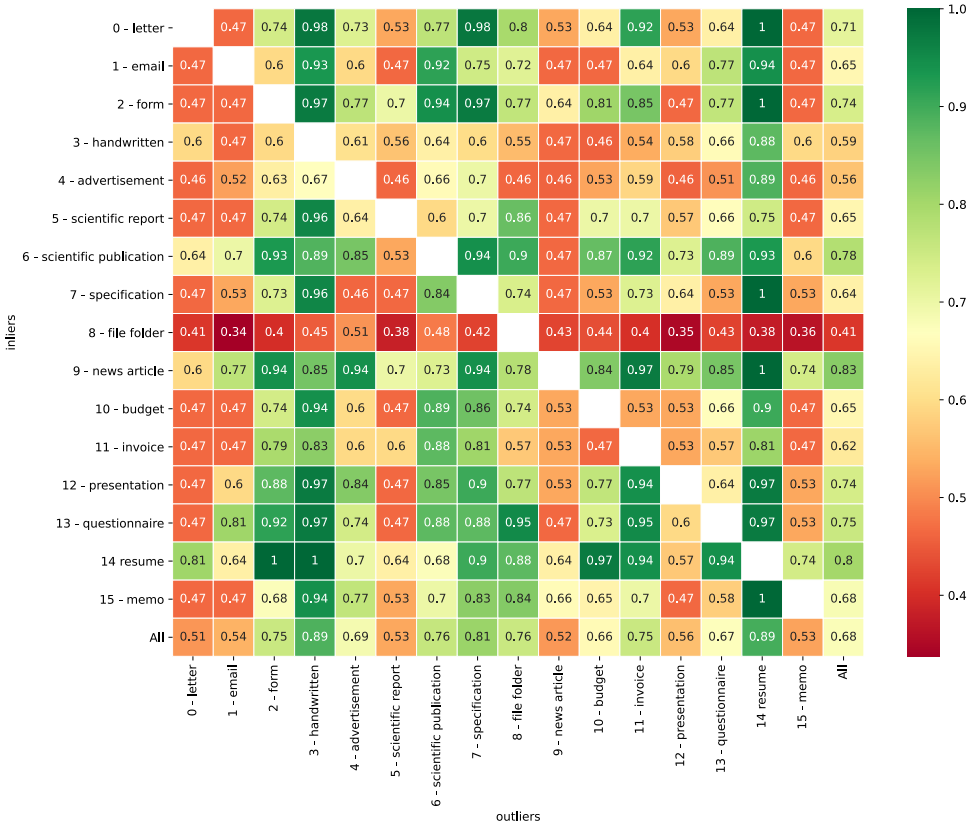


Figure 41: One class chosen as the inlier class and one as the outlier class for the train and test set. Contamination is 0.2 and score shown $f1_{macro}$.

outlier class to all other unseen outlier classes: During training only data from all inlier classes as well as one outlier class was used, while during testing predictions are made for all unseen outlier classes and the inliers. Results are shown in figure 43 and are quite mixed. For several classes the performance does not reach the baseline of 0.9 (with the test training data being split 90:10). There seems to be a trend of classes with continuous texts performing better at this task, while documents with table style layout and short non-continuous texts might perform worse. However overall results are unclear and it should be noted that generalizing to so many unseen classes is hard in a standard fully connected architecture:

Lastly the most relevant experiment for the case of a dataset with mostly inlier

and some outlier classes was conducted: Prediction on data from unseen outlier classes (previously described as **one-unseen** task). To do this, the training dataset was built using all inlier classes as well as all outlier classes except for one. This unseen outlier class was then predicted during testing, together with the inliers. Results can be seen in figure 47a (a) and 47b (b). In this test the fully connected neural network was able to beat the baseline of 0.8 for every tested unseen class. Additionally it is able to beat the traditional methods of using an SVM or OCSVM in the majority of cases. Results for most classes are in line with previous results. However classes like *specification* or *scientific report* significantly deviate in their results and are much harder or easier to generalize to and distinguish from inlier classes than revealed by previous tests.

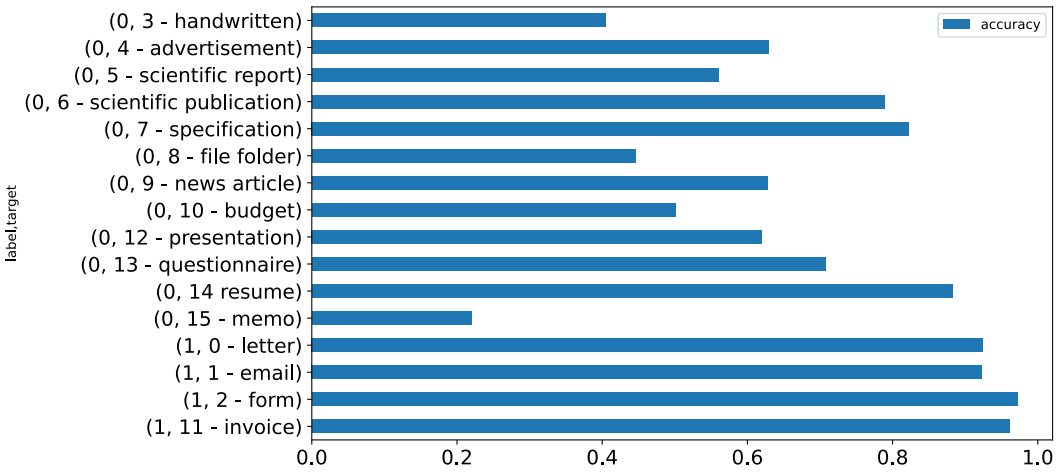


Figure 42: Results of training a binary classifier for inliers and outliers, looking at the accuracy of prediction for each class. Contamination is 0.1, so the baseline for the inliers is 0.9.

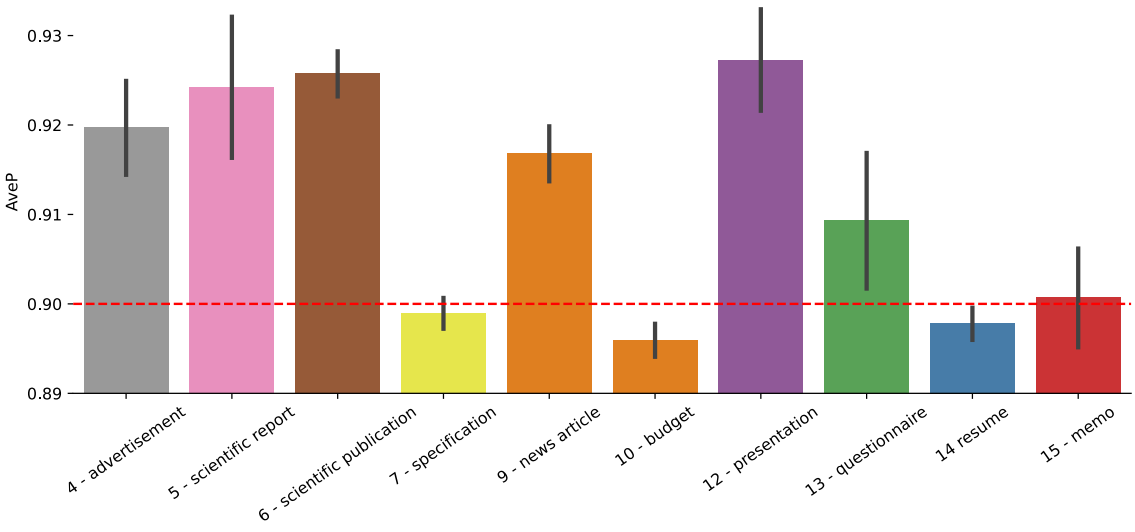


Figure 43: Generalizing from one outlier class in the training dataset to all outlier classes in the test dataset. Average Precision for the test set of all inliers and all other outlier classes not seen during training.

4.3.2 Outlier Exposure

To alleviate the problem of too few data points and a highly imbalanced dataset, inspiration was taken from the idea of **Outlier Exposure** [Hen+19] (see chapter 4.1) and documents from the unrelated datasets IMDB, 20 News, Amazon, All News and Wikipedia (see chapter 3.2.2) were randomly sampled and used as additional data for the outlier class. As can be seen in the results of table 4, this has no impact on the performance for the binary classification task. The only thing that can be observed is a shift in the threshold, with accuracy for inliers at the threshold of 0.5 lowering, while accuracy for outliers increases. Looking at the averaged results for the one-unseen task in table 5, there seems to be a minimal performance difference, when using Outlier Exposure. In figure 44 it can be seen that the effect of OE for most classes is slightly positive, with only one major exception for the *specification* class where using too much additional data significantly worsened the ability of the model to generalize to it.

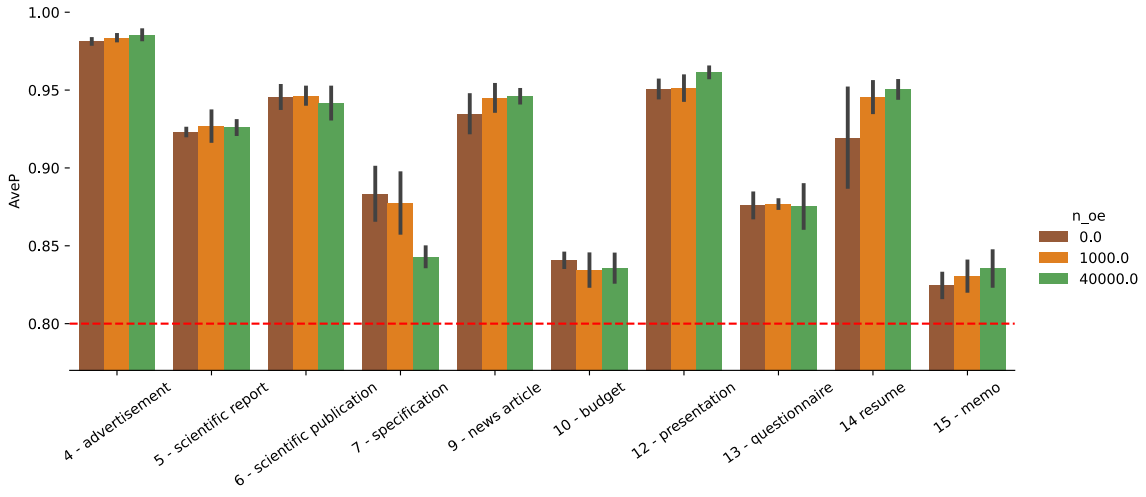


Figure 44: One-unseen class with additional OE data with 80:20 data split and Average Precision.

n_{OE}	$F1_{macro}$	F1	acc_{in}	acc_{out}	AUC-ROC	AUC-PRC
0	0.724	0.956	0.965	0.466	0.864	0.980
50	0.730	0.956	0.962	0.486	0.864	0.980
500	0.737	0.955	0.959	0.505	0.861	0.979
2,000	0.734	0.953	0.955	0.508	0.855	0.978
10,000	0.731	0.951	0.949	0.522	0.858	0.979
40,000	0.731	0.951	0.951	0.514	0.856	0.978

Table 4: Supervised Outlier Exposure results on binary classification task between inlier and outlier classes.

n_{OE}	$F1_{macro}$	F1	acc_{in}	acc_{out}	AUC-ROC	AUC-PRC
0	0.615	0.904	0.970	0.242	0.750	0.908
1,000	0.621	0.906	0.970	0.253	0.758	0.912
40,000	0.633	0.906	0.966	0.283	0.755	0.910

Table 5: Supervised Outlier Exposure averaging over all classes for the one-unseen task.

4.3.3 One Class

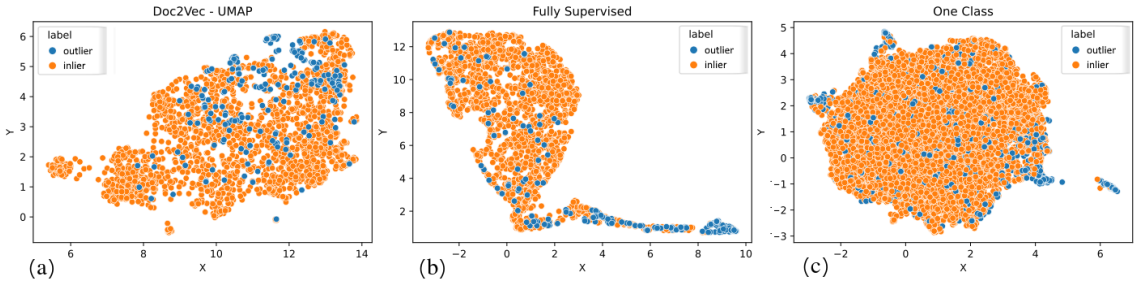


Figure 45: Vector representations using UMAP to reduce vectors to 2D. The classes are inliers (1) and outliers (0). (a) shows raw Doc2Vec vectors, (b) using the second to last layer of the fully supervised binary classifier and (c) using the second to last layer of the one-class classifier.

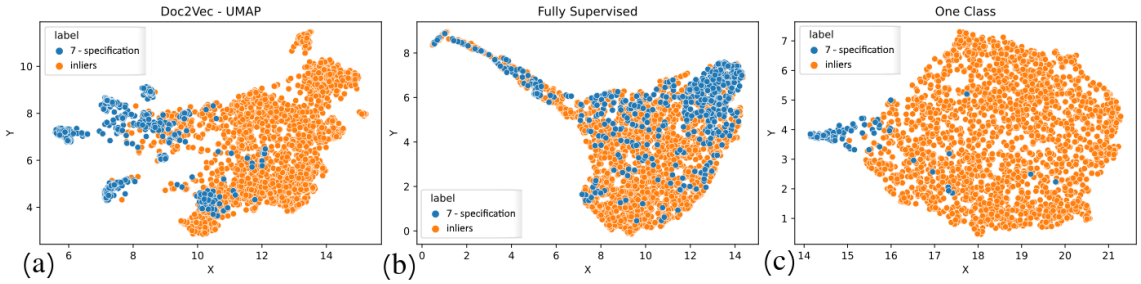


Figure 46: Same as above but for the one-unseen task, showing only a single outlier class in the test set.

For the one-class classification, as laid out in chapter 4.2.1, the text vectors extracted from Doc2Vec were first transformed to lay close together in a new (smaller) vector space, while ensuring that the representation was still meaningful and did not collapse by additionally training to distinguish between *reference data* (which are the outlier classes from the RVL-CDIP dataset itself). The effect of this can be seen in figure 45: Looking at the extracted vectors without any alteration, reduced to 2D using UMAP, there are some visible shapes and structures (note, as discovered in the Unsupervised chapter, this is likely due to UMAP discovering meaningful patterns in the data by itself when transforming to 2D). However the outliers occur scattered and mixed with the inlier data. After learning representations during training with a fully supervised classifier, the outliers mostly clump outside an area of inliers. The same can be seen when first using the one-class classification training, however here the inliers are clearly condensed around one point in the vector space with outliers

Model	$F1_{macro}$	F1	acc_{in}	acc_{out}	AUC-ROC	AveP
ZeroR	0.5	0.888	1.0	0.0	0.5	0.8
NN	0.623	**0.906**	0.972	0.254	0.748	0.906
OCNN-NN	0.645	0.870	0.864	0.461	0.748	0.919
OCNN-OCSVM	0.449	0.898	**1.000**	0.000	0.688	0.897
OCNN-SVM	0.673	0.885	0.885	0.492	**0.780**	**0.927**
OCSVM	0.544	0.716	0.600	**0.647**	0.623	0.860
SVM	**0.702**	0.899	0.904	0.535	0.719	0.889
20 News						
OCNN-NN	0.531	0.826	0.825	0.255	0.560	0.839
OCNN-OCSVM	0.449	0.898	1.000	0.000	0.649	0.882
OCNN-SVM	0.531	0.880	0.942	0.133	0.594	0.852

Table 6: Average of one-unseen task results for all classes. Comparing the performance of different architectures with and without one-class neural network as well as using 20 News data as reference data instead of the outlier classes from RVL-CDIP during training.

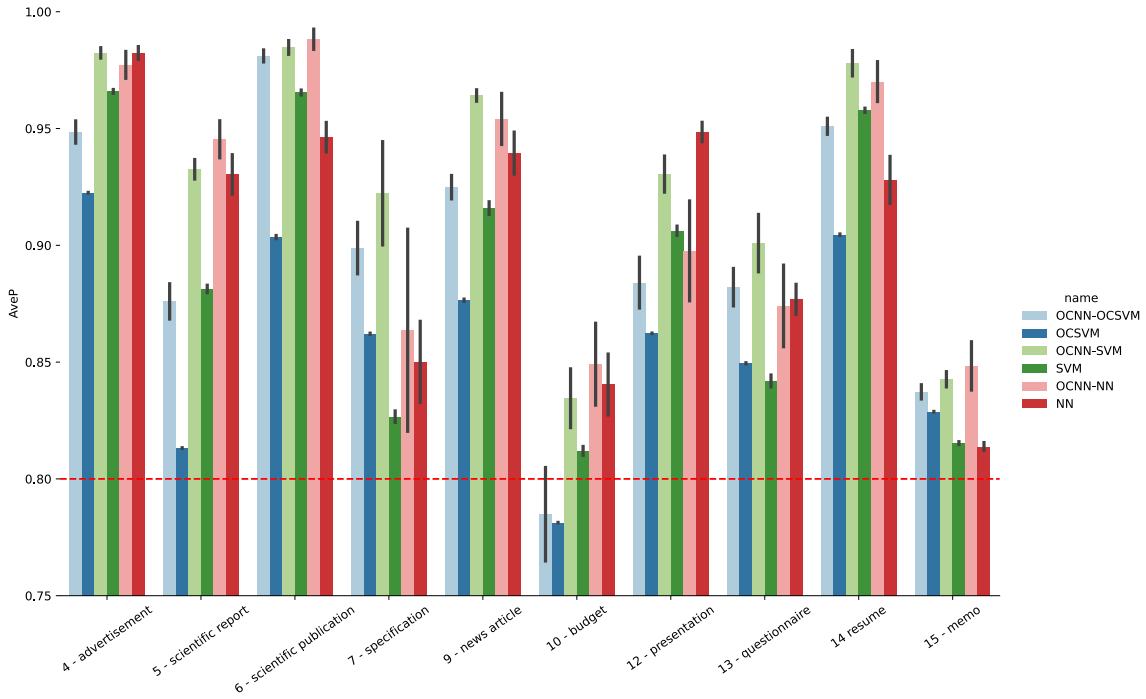
laying mostly on the outside of this sphere. This can be seen even more obviously when considering the training scenario of one new outlier visualized in figure 46.

The one-class approach led to a significant improvement in performance when doing one-unseen outlier testing, as seen in figure 47a. Using the one-class neural network (OCNN) before using an OCSVM or SVM is able to severely increase performance over using the raw Doc2Vec vectors as input for every unseen class. The difference between the standard neural network and the neural network on top of the OCNN are more varied, however overall with the exception of the performance for the *presentation* class, OCNNs were able to get the best results.

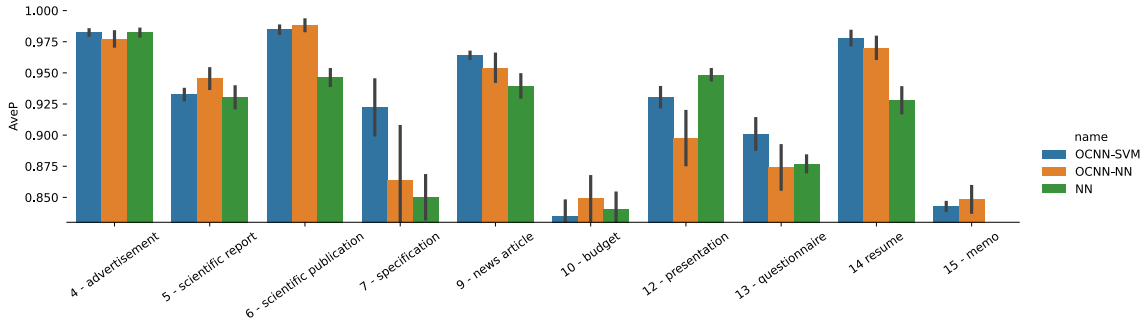
When averaging over the results of all classes and looking at different metrics (table 6), it can be seen that all tested models outperform the baseline set by the ZeroR classifier for AUC-ROC and AveP scores. Additionally both of these show a high correlation. Looking at the threshold based scores however reveals some problems: The optimal threshold of the model with the highest AUC-ROC and AveP - the OCNN-SVM - does not seem to be at 0.5, as it is outperformed in both F1 and $F1_{macro}$ by other architectures. Both OCSVM based models are even further from the optimum at 0.5, as the OCNN-OCSVM classifies everything as inliers and

the pure OCSVM is very biased in favor of classifying data points as outliers at this threshold.

Lastly, the reference data was originally intended to be from another dataset entirely. This approach was tested and compared with the reference data being the outliers from the RVL-CDIP dataset itself. As can be seen in figure 48 and table 6, using 20 news as reference data proved to be significantly worse than using the RVL-CDIP data. Notably, the performance for the class *news article* was just average, even though the reference data consists only of news articles. However as shown in the previous results in chapter 3.3.1 comparing Doc2Vec models with the same parameters but different data sources, the choice of data can have a big influence on performance. So the results for using the 20 News dataset as a reference might not hold true for other datasets.



(a)



(b)

Figure 47: (a) Model comparison for one-unseen task, a 80:20 split test set and showing the Average Precision. Black lines on bars are standard deviation, red line is a baseline for the Zero Rule. (b) Zoomed in view comparing only performances of OCNN with NN or SVM with a pure NN approach.

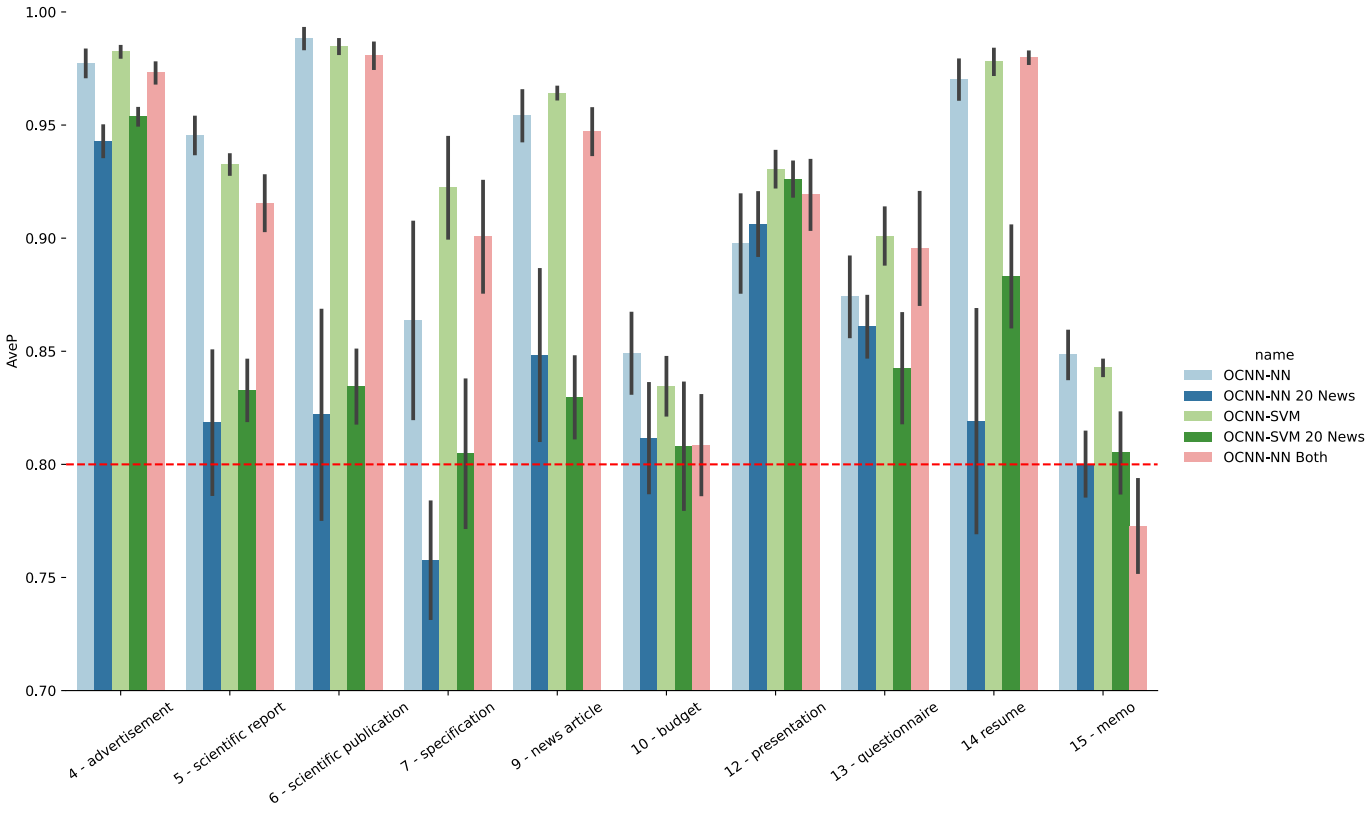


Figure 48: Model comparison supervised results using 20 News as data reference compared to using RVL-CDIP outliers. Black lines on bars are standard deviation, red line is a baseline for ZeroR.

4.3.4 Multimodal

In addition to using only text embeddings, the RVL-CDIP dataset lends itself to create visual embeddings, since the raw data exist as images. As can be seen in table 7, using image vectors extracted from VGG16 pre-trained on ImageNet is strictly outperformed by using the textual vectors from Doc2Vec on the one-unseen task. This can be explained by the fact that ImageNet is a dataset of mostly natural images. So while VGG16 has learned some visual fundamentals to distinguish between different looking documents, it is not able to understand fine details without fine-tuning on a document dataset first.

However when reducing the image vectors to the 300 dimensional size of the text vectors and combining the two, this multimodal feature vector handily outperforms the purely textual or visual approach both across all different thresholds as well as for the specific threshold at 0.5.

Looking at the results for the AveP for the one-unseen task, split up by the different classes (figure 49, it is clear that the textual vectors were able to outperform the image vectors on every class, with a possible exception of *presentations* and *news articles*. Image vectors also had by far the most variance in their results. The combined vectors were able to outperform both other vectors on every class again. Most notably this was also the case when the image vector results were mostly below the baseline, such as with the *questionnaire* class.

Model	$F1_{macro}$	F1	acc_{in}	acc_{out}	AUC-ROC	AveP
Text	0.645	0.870	0.864	0.461	0.748	0.919
Image	0.590	0.888	0.950	0.234	0.648	0.865
Text+Img	**0.725**	**0.917**	**0.951**	**0.495**	**0.827**	**0.944**

Table 7: Model performances with feature vectors of only Doc2Vec (Text), only VGG16 (Image) or a combination. Average over all classes when testing the one-unseen task.

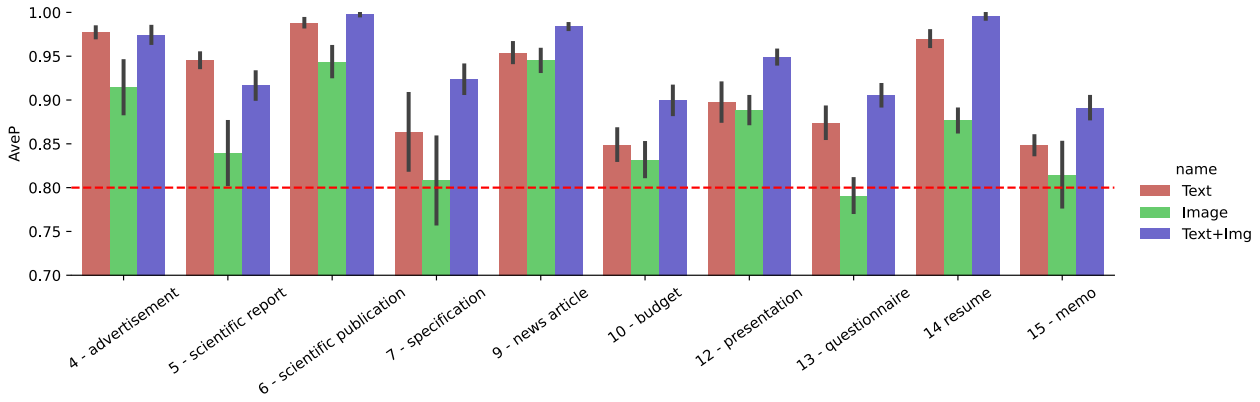


Figure 49: Average Precision for purely textual, visual and combined vectors for each class when running the one-unseen task.

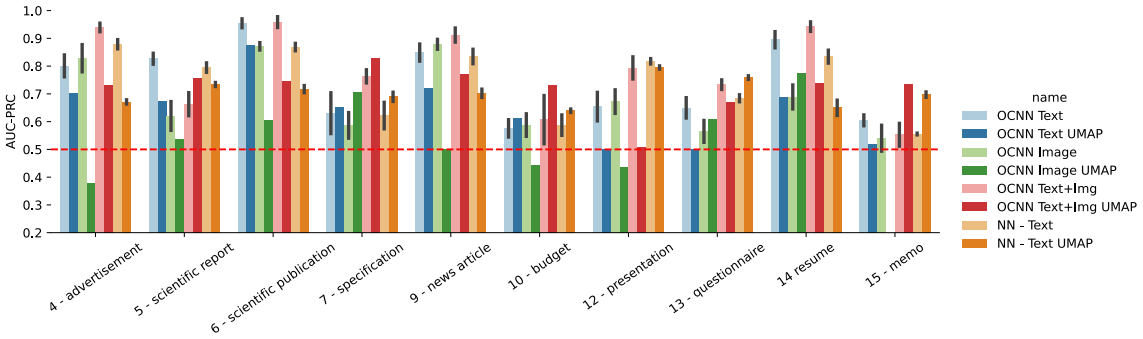


Figure 50: Difference between AUC-ROC results for OCNN-NN networks running the one-unseen task for different outlier classes and using different vector inputs. In almost all cases UMAP makes results worse than using the raw vector input, showing that neural networks can find more useful non-linear relationships when trained in a supervised manner.

4.3.5 Weakly Supervised

In many scenarios an outlier class is not completely unknown and instead a couple of samples can be used for training. This is described as 'weakly supervised' when using only a few samples. This was attempted in with the existing pipeline for text embeddings with one-class classification and a neural network classifier for the one-unseen task. As is obvious from 51, using 5 or 15 samples of the "unseen" outlier class during training produced varying results. On average the AUC-PRC could be improved from **0.915** when having a completely unseen class to **0.924** when showing 5 samples. Keeping in mind that the Zero Rule baseline is at 0.9 (due to a contamination of 0.1), this seems like a notable improvement. However when using 15 samples during training, this drops back to **0.916**, which can be interpreted such that the performance variance when using any weakly supervision is merely statistical noise.

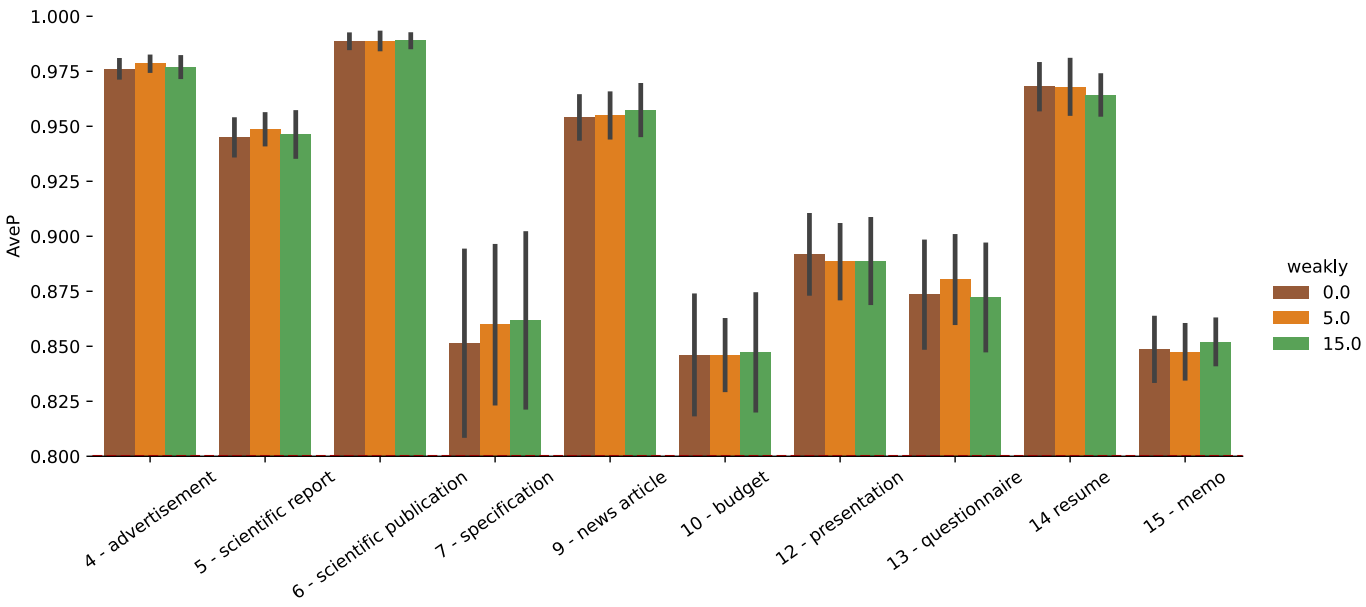


Figure 51: One-unseen task results for weakly supervision. Zero, five or fifteen samples for the unseen outlier class in the training dataset.

5 Summary and Outlook

Anomaly detection is an active field of research that is relevant for many use cases. This work provided an overview of the field in general, using a variety of deep and shallow algorithms, and specifically its applications for the detection of outliers in text document datasets. Additionally an overview was given for different techniques to convert text documents to feature vectors that are further processable by outlier detection algorithms, based on textual, visual and combined features extracted from the documents. Experiments were done to empirically show the effectiveness of these techniques to successfully find outliers in text document datasets, the results of which are summarized here.

5.1 Unsupervised Anomaly Detection

Unsupervised Anomaly Detection describes the case of starting with a new dataset that lacks any labels, both for inliers or outliers. The task is then to clean this dataset by finding the outlier data, to remove or otherwise handle it before using the dataset for further applications. The results of the experiments done for this thesis are as follows:

- A number of different algorithms have been shown to successfully separate the data into inliers and outliers, with the best ones being the classic **OCSVM**, using an **autoencoder's** reconstruction loss as the anomaly score and using the contrastive and triplet-loss-based neural network and dimensionality reduction technique **ivis**.
- There are big performance differences when using dimensionality reduction algorithms before applying the outlier detection algorithms on the document feature vectors, with **UMAP** showing the strongest performance gains.
- In regards to representations, **Doc2Vec** extracted feature vectors have been shown to outperform powerful Transformer models such as the **Longformer** without fine-tuning, as well as other word vector based techniques.
- It could be shown that including a bit of manual labour and expertise can make the process of unsupervised outlier detection achieve the best performance and a lot more stability by **clustering** the data and then picking the correct inlier

and outlier clusters. Again, this was massively improved by using the correct dimensionality reduction before clustering (as well as for visualization).

Improvements could be made by using a more powerful pre-trained Transformer model and **fine-tuning** it on the used dataset. Additionally, there have been more advances in **self-supervision and contrastive learning**, to improve the quality of unsupervised representation learning. When using a pipeline that returns an outlier score, this may be integrated into an iterative **active learning** process, whereby more and more of the dataset could be labelled, guided by the uncertainty of the model which document data points are outliers. The thus obtained labels can improve the overall outlier detection performance by ending up with a more powerful model that is trained in a semi-supervised manner. This is in fact already possible when using UMAP and ivis, as both support partial labelling.

5.2 Supervised Anomaly Detection

Supervised Anomaly Detection describes the separation of outlier data from inlier data using all or partially labelled datasets for training. The focus here was on finding models that can generalize to new, unseen and different outlier data after training. The experiments had the following results:

- Deep neural networks proved to have better performance than the traditional models SVM and OCSVM.
- The idea of transforming the feature vectors in such a way, that inlier vectors lie closer together in the feature space, in order to keep them separate from outliers, was successful. In this **one-class** style training, mode collapse could successfully be prevented by training the model not only on the inlier data but also on different classes of reference data, that it was supposed to correctly classify.
- **Multimodal** feature vectors, consisting of concatenated feature vectors of textual features extracted from a pre-trained Doc2Vec model and visual features extracted from a pre-trained VGG-16 model, showed big improvements in performance, even despite the relatively bad performance of using a VGG-16 model trained on ImageNet and without fine-tuning on the document data.

The supervised case, using deep neural networks, could likely see some big performance increases when **fine-tuning** the feature generating language model (a Transformer model) and visual model (VGG-16 or the more recent EfficientNet) on document data. Especially the badly performing visual models might benefit from this, as they are usually pre-trained on ImageNet, which is very different from text documents. Not only can these models be better fine-tuned on the document data, but also get integrated in the training process, with the loss signal during the one-class classification training running through both the one-class classifier as well as the Transformer and visual model. Finally, there are different methods to achieve deep one-class classification and other deep models for anomaly detection, which could be tried for outlier detection for text documents and the field is constantly advancing [Ruf+21].

Appendices

A Unsupervised Anomaly Detection

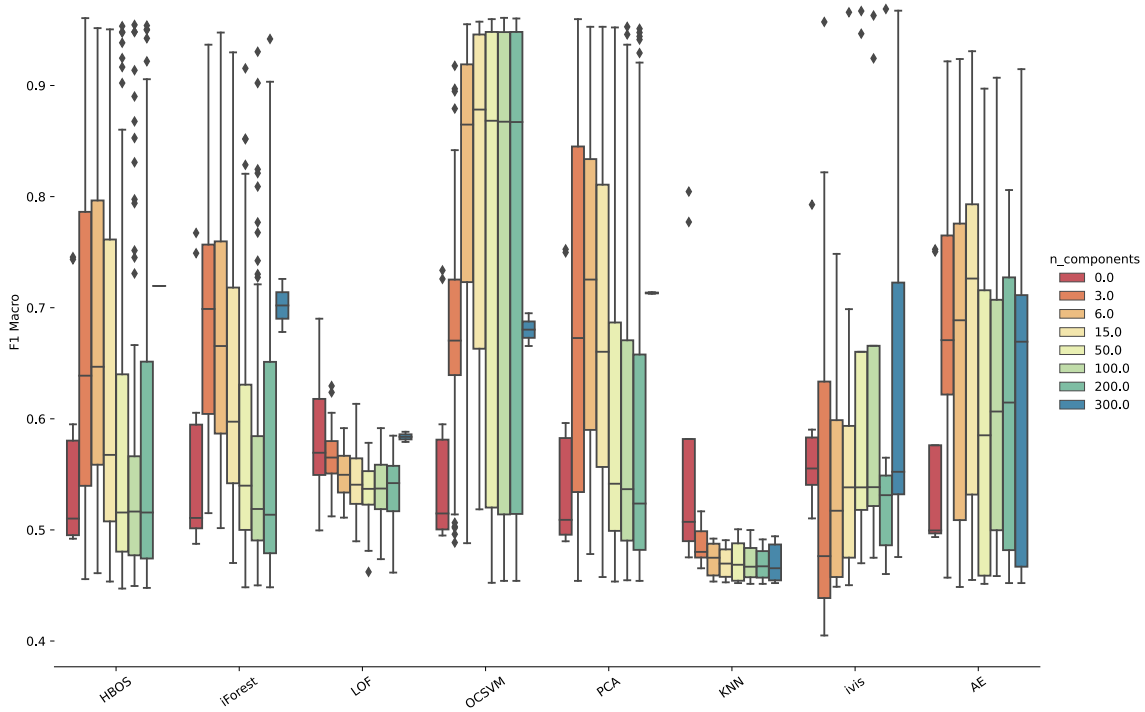


Figure 52: Unsupervised outlier detector and n-dimension combination using UMAP.

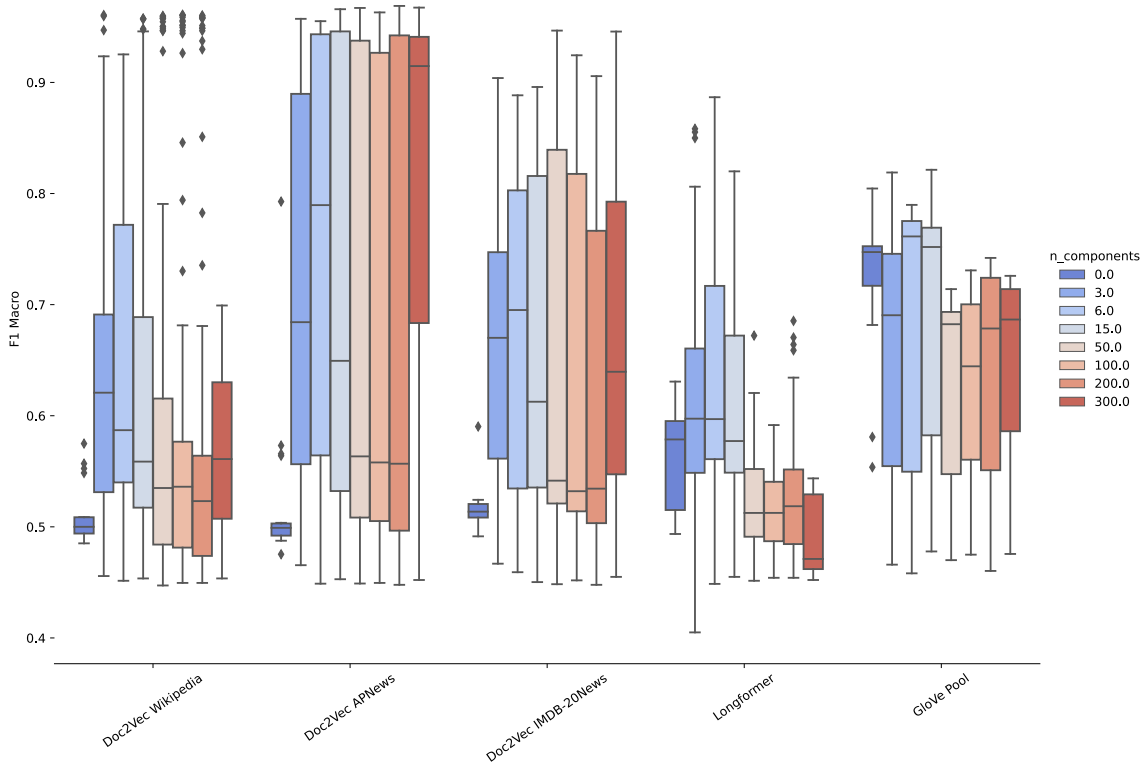


Figure 53: Unsupervised model and n-dimension combination using UMAP.

B Supervised Anomaly Detection

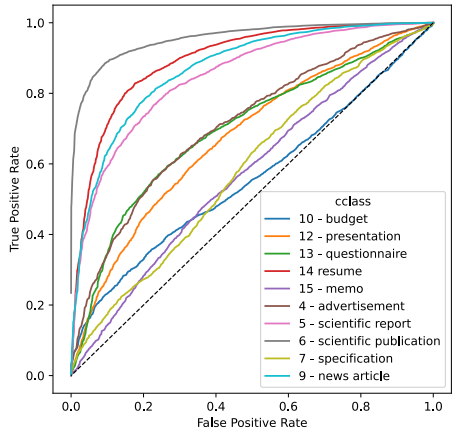


Figure 54: ROC - combined vectors and OCNN-NN architecture for the one-unseen task, by class.

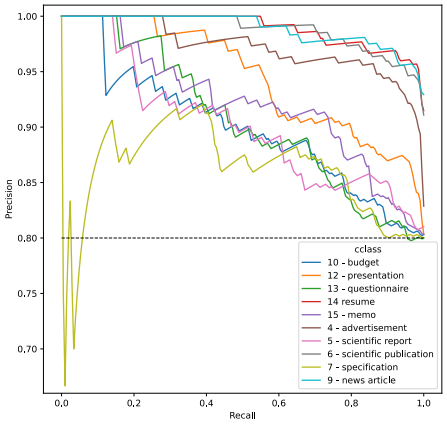


Figure 55: PRC - combined vectors and OCNN-NN architecture for the one-unseen task, by class.

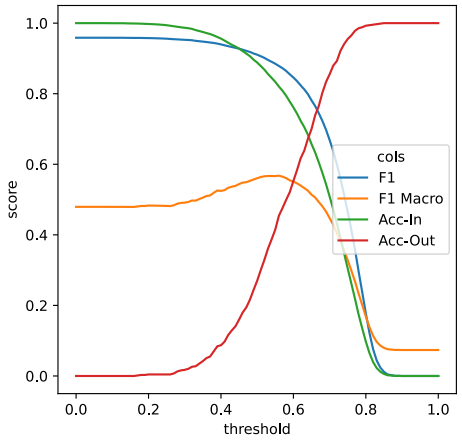


Figure 56: Scores and threshold curve. Text vectors only.

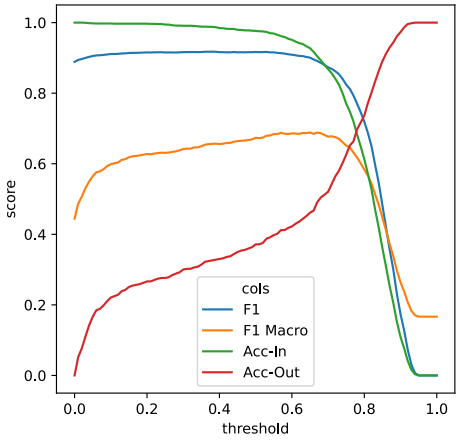


Figure 57: Scores and threshold curve. Combined text and image vectors.

References

[Afz+17] Muhammad Zeshan Afzal et al. "Cutting the Error by Half: Investigation of Very Deep CNN and Advanced Training Strategies for Document Image Classification". en. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* (Nov. 2017). arXiv: 1704.03557, pp. 883–888. DOI: 10.1109/ICDAR.2017.149. URL: http://arxiv.org/abs/1704.03557 (visited on 12/07/2020).

[Akb+19a] Alan Akbik et al. "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Association for Computational Linguistics, June 2019, pp. 54–59. DOI: 10.18653/v1/N19-4010. URL: https://www.aclweb.org/anthology/N19-4010.

[Akb+19b] Alan Akbik et al. "FLAIR: An easy-to-use framework for state-of-the-art NLP". In: *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 2019, pp. 54–59.

[AKC20] Mebarka Allaoui, Mohammed Lamine Kherfi, and Abdelhakim Cheriet. "Considerably Improving Clustering Algorithms Using UMAP Dimensionality Reduction Technique: A Comparative Study". en. In: *Image and Signal Processing*. Ed. by Abderrahim El Moataz et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 317–325. ISBN: 978-3-030-51935-3. DOI: 10.1007/978-3-030-51935-3_34.

[Ala] Jay Alammar. *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*. URL: http://jalammar.github.io/illustrated-bert/.

[ALM17] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. "A Simple but Tough-to-Beat Baseline for Sentence Embeddings". en. In: *ICLR 2017* (2017), p. 16.

[Bak+20] Souhail Bakkali et al. "Visual and Textual Deep Feature Fusion for Document Image Classification". en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Seattle, WA, USA: IEEE, June 2020, pp. 2394–2403. ISBN: 978-1-72819-360-1. DOI: 10.1109/CVPRW50498.2020.00289. URL: https://ieeexplore.ieee.org/document/9150829/ (visited on 11/19/2020).

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. COLT '92. Association for Computing Machinery, July 1992, pp. 144–152. ISBN: 978-0-89791-497-0. DOI: 10.1145/130385.130401. URL: https://doi.org/10.1145/130385.130401.

[Boj+17] Piotr Bojanowski et al. *Enriching Word Vectors with Subword Information*. arXiv: 1607.04606. June 2017. URL: http://arxiv.org/abs/1607.04606.

[BPC20] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. arXiv: 2004.05150. Apr. 2020. URL: http://arxiv.org/abs/2004.05150 (visited on 05/21/2020).

[Bre+00] Markus M Breunig et al. "LOF: Identifying Density-Based Local Outliers". In: *Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data, Dalles, TX, 2000* (2000), p. 12.

[Bro+20] Tom B. Brown et al. *Language Models are Few-Shot Learners*. arXiv: 2005.14165. July 2020. URL: http://arxiv.org/abs/2005.14165.

[Bro19] Jason Brownlee. *How to Visualize Filters and Feature Maps in Convolutional Neural Networks*. May 2019. URL: https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/.

[CC19] Raghavendra Chalapathy and Sanjay Chawla. *Deep Learning for Anomaly Detection: A Survey*. en. arXiv: 1901.03407. Jan. 2019. URL: http://arxiv.org/abs/1901.03407 (visited on 02/19/2020).

[Che17] Minmin Chen. "Efficient Vector Representation for Documents through Corruption". In: *CoRR* abs/1707.02377 (2017). arXiv: 1707.02377. URL: http://arxiv.org/abs/1707.02377.

[Cho+15] François Chollet et al. *Keras*. https://keras.io. 2015.

[CMC19] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. *Anomaly Detection using One-Class Neural Networks*. arXiv: 1802.06360. Jan. 2019. URL: http://arxiv.org/abs/1802.06360 (visited on 02/26/2020).

[CMS13] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. "Density-Based Clustering Based on Hierarchical Density Estimates". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Jian Pei et al. Lecture Notes in Computer Science. Springer, 2013, pp. 160–172. ISBN: 978-3-642-37456-2. DOI: 10.1007/978-3-642-37456-2_14.

[CNM16] Van Loi Cao, Miguel Nicolau, and James McDermott. "A Hybrid Autoencoder and Density Estimation Model for Anomaly Detection". en. In: *Parallel Problem Solving from Nature – PPSN XIV*. Ed. by Julia Handl et al. Vol. 9921. Cham: Springer International Publishing, 2016, pp. 717–726. ISBN: 978-3-319-45822-9 978-3-319-45823-6. DOI: 10.1007/978-3-319-45823-6_67. URL: http://link.springer.com/10.1007/978-3-319-45823-6_67 (visited on 02/26/2020).

[Con+18] Alexis Conneau et al. "What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, July 2018, pp. 2126–2136. DOI: 10.18653/v1/P18-1198. URL: https://www.aclweb.org/anthology/P18-1198.

[Das+18] Arindam Das et al. *Document Image Classification with Intra-Domain Transfer Learning and Stacked Generalization of Deep Convolutional Neural Networks*. arXiv: 1801.09321. Aug. 2018. URL: http://arxiv.org/abs/1801.09321.

[Dev+19] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv: 1810.04805. May 2019. URL: http://arxiv.org/abs/1810.04805 (visited on 08/31/2020).

[Di +19] Federico Di Mattia et al. "A Survey on GANs for Anomaly Detection". In: *arXiv:1906.11632 [cs, stat]* (June 2019). arXiv: 1906.11632. URL: http://arxiv.org/abs/1906.11632.

[Dom+18] Rémi Domingues et al. "A comparative evaluation of outlier detection algorithms: Experiments and analyses". en. In: *Pattern Recognition* 74 (Feb. 2018), pp. 406–421. ISSN: 00313203. DOI: 10.1016/j.patcog.2017.09.037. URL: https://linkinghub.elsevier.com/retrieve/pii/S0031320317303916 (visited on 09/01/2020).

[DT18] Terrance DeVries and Graham W. Taylor. *Learning Confidence for Out-of-Distribution Detection in Neural Networks*. 2018. eprint: 1802.04865. URL: https://arxiv.org/abs/1802.04865.

[EKX96] Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)* (1996), p. 6.

[Eze20] Aysu Ezen-Can. *A Comparison of LSTM and BERT for Small Corpus*. arXiv: 2009.05451. Sept. 2020. URL: http://arxiv.org/abs/2009.05451 (visited on 09/21/2020).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[GD12] Markus Goldstein and Andreas Dengel. *Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm*. 2012.

[GU16] Markus Goldstein and Seiichi Uchida. "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data". en. In: *PLOS ONE* 11.4 (Apr. 2016). Ed. by Dongxiao Zhu, e0152173. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0152173. URL: https://dx.plos.org/10.1371/journal.pone.0152173 (visited on 09/15/2020).

[Har15] Konstantinos G Derpanis Harley Alex Ufkes. "Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval". In: *International Conference on Document Analysis and Recognition (ICDAR 2015)* (2015).

[He+15] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385 [cs]* (Dec. 2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[Hen+19] Dan Hendrycks et al. "Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty". en. In: (2019), p. 12.

[Hes+17] Joel Hestness et al. *Deep Learning Scaling is Predictable, Empirically*. arXiv: 1712.00409. Dec. 2017. URL: http://arxiv.org/abs/1712.00409.

[HG18] Dan Hendrycks and Kevin Gimpel. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*. en. arXiv: 1610.02136. Oct. 2018. URL: http://arxiv.org/abs/1610.02136 (visited on 02/19/2020).

[HMD19] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. *Deep Anomaly Detection with Outlier Exposure*. en. arXiv: 1812.04606. Jan. 2019. URL: http://arxiv.org/abs/1812.04606 (visited on 02/19/2020).

[Jou+16] Armand Joulin et al. *Bag of Tricks for Efficient Text Classification*. arXiv: 1607.01759. Aug. 2016. URL: http://arxiv.org/abs/1607.01759.

[Kra91] Mark A. Kramer. "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE Journal* 37.2 (1991), pp. 233–243. ISSN: 1547-5905. DOI: https://doi.org/10.1002/aic.690370209. URL: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012, pp. 1097–1105. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[Kus+15] Matt Kusner et al. "From Word Embeddings To Document Distances". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 957–966. URL: http://proceedings.mlr.press/v37/kusnerb15.html.

[KW19] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019). arXiv: 1906.02691, pp. 307–392. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000056. URL: http://arxiv.org/abs/1906.02691.

[LB16] Jey Han Lau and Timothy Baldwin. *An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation*. arXiv: 1607.05368. July 2016. URL: http://arxiv.org/abs/1607.05368 (visited on 07/06/2020).

[Liu+19] Nelson F. Liu et al. "Linguistic Knowledge and Transferability of Contextual Representations". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, June 2019, pp. 1073–

1094. DOI: 10.18653/v1/N19-1112. URL: https://www.aclweb.org/anthology/N19-1112.

[Llo82] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137. ISSN: 1557-9654. DOI: 10.1109/TIT.1982.1056489.

[LM14] Quoc V. Le and Tomas Mikolov. *Distributed Representations of Sentences and Documents*. arXiv: 1405.4053. May 2014. URL: http://arxiv.org/abs/1405.4053 (visited on 10/06/2020).

[LTZ12] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation-Based Anomaly Detection". In: *ACM Transactions on Knowledge Discovery from Data* 6.1 (Mar. 2012), pp. 1–39. ISSN: 1556-4681, 1556-472X. DOI: 10.1145/2133360.2133363. URL: https://dl.acm.org/doi/10.1145/2133360.2133363.

[LZV18] Chundi Liu, Shunan Zhao, and Maksims Volkovs. *Unsupervised Document Embedding With CNNs*. arXiv: 1711.04168. Feb. 2018. URL: http://arxiv.org/abs/1711.04168.

[Maa+11] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[Mah+20] Niall O' Mahony et al. *Deep Learning vs. Traditional Computer Vision*. arXiv: 1910.13796. 2020. DOI: 10.1007/978-3-030-17795-9. URL: http://arxiv.org/abs/1910.13796.

[McC+19] Ryan McConville et al. "N2D: (Not Too) Deep Clustering via Clustering the Local Manifold of an Autoencoded Embedding". In: *CoRR* abs/1908.05968 (2019). arXiv: 1908.05968. URL: http://arxiv.org/abs/1908.05968.

[MH08] Laurens van der Maaten and Geoffrey E. Hinton. "Visualizing High-Dimensional Data Using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[MHM18] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. en. arXiv: 1802.03426. Dec. 2018. URL: http://arxiv.org/abs/1802.03426 (visited on 09/15/2020).

[Mik+13a] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. arXiv: 1310.4546. Oct. 2013. URL: http://arxiv.org/abs/1310.4546.

[Mik+13b] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv* (Jan. 2013). eprint: 1301.3781. URL: https://arxiv.org/abs/1301.3781.

[Min+21] Shervin Minaee et al. *Deep Learning Based Text Classification: A Comprehensive Review*. arXiv: 2004.03705. Jan. 2021. URL: http://arxiv.org/abs/2004.03705.

[MLS13] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. *Exploiting Similarities among Languages for Machine Translation*. arXiv: 1309.4168. Sept. 2013. URL: http://arxiv.org/abs/1309.4168.

[Moh+17] Mehdi Mohammadi et al. *Deep Learning for IoT Big Data and Streaming Analytics: A Survey*. arXiv: 1712.04301. Dec. 2017. URL: http://arxiv.org/abs/1712.04301.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. URL: http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[MYZ13] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2013), p. 6.

[NHB17] Marwa Naili, Anja Habacha, and Henda Ben Ghezala. "Comparative study of word embedding methods in topic segmentation". In: *Procedia Computer Science* 112 (Dec. 2017), pp. 340–349. DOI: 10.1016/j.procs.2017.08.009.

[NLM19] Jianmo Ni, Jiacheng Li, and Julian McAuley. "Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019, pp. 188–197. DOI: 10.18653/v1/D19-1018. URL: https://www.aclweb.org/anthology/D19-1018.

[NLW19] Kai Nakamura, Sharon Levy, and William Yang Wang. *r/Fakeddit: A New Multimodal Benchmark Dataset for Fine-grained Fake News Detection*. en. arXiv: 1911.03854. Nov. 2019. URL: http://arxiv.org/abs/1911.03854 (visited on 02/19/2020).

[Pal+16] Hamid Palangi et al. "Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.4 (Apr. 2016). arXiv: 1502.06922, pp. 694–707. ISSN: 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2016.2520371. URL: http://arxiv.org/abs/1502.06922.

[Pan+20] Guansong Pang et al. *Deep Weakly-supervised Anomaly Detection*. en. arXiv: 1910.13601. Jan. 2020. URL: http://arxiv.org/abs/1910.13601 (visited on 02/19/2020).

[Pas+10] Cláudia Pascoal et al. "Detection of Outliers Using Robust Principal Component Analysis: A Simulation Study". In: *Advances in Intelligent and Soft Computing*. Vol. 77. journalAbbreviation: Advances in Intelligent and Soft Computing. Dec. 2010, pp. 499–507. ISBN: 978-3-642-14745-6. DOI: 10.1007/978-3-642-14746-3_62.

[PMP20] Subhojeet Pramanik, Shashank Mujumdar, and Hima Patel. *Towards a Multi-modal, Multi-task Learning based Pre-training Framework for Document Representation Learning*. arXiv: 2009.14457. Sept. 2020. URL: http://arxiv.org/abs/2009.14457.

[PP19] Pramuditha Perera and Vishal M. Patel. "Learning Deep Features for One-Class Classification". en. In: *IEEE Transactions on Image Processing* 28.11 (Nov. 2019). arXiv: 1801.05365, pp. 5450–5463. ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2019.2917862. URL: http://arxiv.org/abs/1801.05365 (visited on 11/17/2020).

[PRS19] Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. "To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks". In: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Association for Computational Linguistics, Aug. 2019, pp. 7–14. DOI: 10.18653/v1/W19-4302. URL: https://www.aclweb.org/anthology/W19-4302.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[Raz+14] Ali Sharif Razavian et al. *CNN Features off-the-shelf: an Astounding Baseline for Recognition*. arXiv: 1403.6382. May 2014. URL: http://arxiv.org/abs/1403.6382.

[RRS00] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. "Efficient algorithms for mining outliers from large data sets". In: *ACM SIGMOD Record* 29.2 (May 2000), pp. 427–438. ISSN: 0163-5808. DOI: 10.1145/335191.335437. URL: https://doi.org/10.1145/335191.335437.

[ŘS10] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[RSL17] Natali Ruchansky, Sungyong Seo, and Yan Liu. *CSI: A Hybrid Deep Model for Fake News Detection*. arXiv: 1703.06959. Sept. 2017. DOI: 10.1145/3132847.3132877. URL: http://arxiv.org/abs/1703.06959.

[Ruf+18] Lukas Ruff et al. "Deep One-Class Classification". en. In: *International Conference on Machine Learning*. ISSN: 1938-7228 Section: Machine Learning. July 2018, pp. 4393–4402. URL: http://proceedings.mlr.press/v80/ruff18a.html (visited on 02/28/2020).

[Ruf+19] Lukas Ruff et al. "Self-Attentive, Multi-Context One-Class Classification for Unsupervised Anomaly Detection on Text". en. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 4061–4071. DOI: 10.18653/v1/P19-1398. URL: https://www.aclweb.org/anthology/P19-1398 (visited on 02/19/2020).

[Ruf+20a] Lukas Ruff et al. *Deep Semi-Supervised Anomaly Detection*. arXiv: 1906.02694. Feb. 2020. URL: http://arxiv.org/abs/1906.02694 (visited on 08/18/2020).

[Ruf+20b] Lukas Ruff et al. *Rethinking Assumptions in Deep Anomaly Detection*. arXiv: 2006.00339. May 2020. URL: http://arxiv.org/abs/2006.00339 (visited on 08/18/2020).

[Ruf+21] Lukas Ruff et al. *A Unifying Review of Deep and Shallow Anomaly Detection*. arXiv: 2009.11732. 2021. DOI: 10.1109/JPROC.2021.3052449. URL: http://arxiv.org/abs/2009.11732.

[Sch+01] Bernhard Schölkopf et al. "Estimating the Support of a High-Dimensional Distribution". In: *Neural Computation* 13.7 (July 2001), pp. 1443–1471. ISSN: 0899-7667. DOI: 10.1162/089976601750264965. URL: https://doi.org/10.1162/089976601750264965 (visited on 10/06/2020).

[Sch+17] Thomas Schlegl et al. *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*. arXiv: 1703.05921. Mar. 2017. URL: http://arxiv.org/abs/1703.05921.

[Sch+18] Marco Schreyer et al. *Detection of Anomalies in Large Scale Accounting Data using Deep Autoencoder Networks*. en. arXiv: 1709.05254. Aug. 2018. URL: http://arxiv.org/abs/1709.05254 (visited on 02/19/2020).

[Sch+99] Bernhard Schölkopf et al. "Support Vector Method for Novelty Detection". en. In: *NIPS 1999* (1999), p. 7.

[She+18] Dinghan Shen et al. *Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms*. arXiv: 1805.09843. May 2018. URL: http://arxiv.org/abs/1805.09843.

[Shy+05] Mei-Ling Shyu et al. "A Novel Anomaly Detection Scheme Based on Principal Component Classifier". In: *Young Lin T., Ohsuga S., Liau CJ., Hu X. (eds) Foundations and Novel Approaches in Data Mining. Studies in Computational Intelligence, vol 9. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11539827_18*. 2005, p. 10.

[SR15] Takaya Saito and Marc Rehmsmeier. "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". In: *PLOS ONE* 10.3 (Mar. 2015), e0118432. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0118432. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0118432.

[Stu+19] Linda Studer et al. *A Comprehensive Study of ImageNet Pre-Training for Historical Document Image Analysis*. arXiv: 1905.09113. May 2019. URL: http://arxiv.org/abs/1905.09113.

[SXL17] Lei Shu, Hu Xu, and Bing Liu. *DOC: Deep Open Classification of Text Documents*. arXiv: 1709.08716. Sept. 2017. URL: http://arxiv.org/abs/1709.08716 (visited on 02/19/2020).

[SZ15] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv: 1409.1556. Apr. 2015. URL: http://arxiv.org/abs/1409.1556.

[Szu+19] Benjamin Szubert et al. "Structure-preserving visualisation of high dimensional single-cell datasets". en. In: *Scientific Reports* 9.1 (June 2019). Number: 1 Publisher: Nature Publishing Group, p. 8914. ISSN: 2045-2322. DOI: 10.1038/s41598-019-45301-0. URL: https://www.nature.com/articles/s41598-019-45301-0 (visited on 09/21/2020).

[tes21] tesseract-ocr. *tesseract*. [Online; accessed 27. Feb. 2021]. Feb. 2021. URL: https://github.com/tesseract-ocr/tesseract.

[TM17] Chris Tensmeyer and Tony Martinez. *Analysis of Convolutional Neural Networks for Document Image Classification*. arXiv: 1708.03273. Aug. 2017. URL: http://arxiv.org/abs/1708.03273 (visited on 12/07/2020).

[Tos+20] Shubham Toshniwal et al. "A Cross-Task Analysis of Text Span Representations". In: *Proceedings of the 5th Workshop on Representation Learning for NLP*. Association for Computational Linguistics, July 2020, pp. 166–176. DOI: 10.18653/v1/2020.repl4nlp-1.20. URL: https://www.aclweb.org/anthology/2020.repl4nlp-1.20.

[Vas+17] Ashish Vaswani et al. *Attention Is All You Need*. arXiv: 1706.03762. Dec. 2017. URL: http://arxiv.org/abs/1706.03762.

[VJ01] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. IEEE Comput. Soc, 2001, pp. I-511-I–518. ISBN: 978-0-7695-1272-3. DOI: 10.1109/CVPR.2001.990517. URL: http://ieeexplore.ieee.org/document/990517/.

[Wie+16] John Wieting et al. *Towards Universal Paraphrastic Sentence Embeddings*. arXiv: 1511.08198. Mar. 2016. URL: http://arxiv.org/abs/1511.08198.

[Wol+20] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL].

[Wu+18] Lingfei Wu et al. *Word Mover's Embedding: From Word2Vec to Document Embedding*. en. arXiv: 1811.01713. Oct. 2018. URL: http://arxiv.org/abs/1811.01713 (visited on 07/06/2020).

[Xie+17] Xuemei Xie et al. *Real-Time Illegal Parking Detection System Based on Deep Learning*. arXiv: 1710.02546. Oct. 2017. DOI: 10.1145/3094243.3094261. URL: http://arxiv.org/abs/1710.02546.

[Xu+20a] Yang Xu et al. *LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding*. arXiv: 2012.14740. Dec. 2020. URL: http://arxiv.org/abs/2012.14740.

[Xu+20b] Yiheng Xu et al. "LayoutLM: Pre-training of Text and Layout for Document Image Understanding". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Aug. 2020). arXiv: 1912.13318, pp. 1192–1200. DOI: 10.1145/3394486.3403172. URL: http://arxiv.org/abs/1912.13318 (visited on 11/19/2020).

[ZCZ19] Jie Zhou, Xingyi Cheng, and Jinchao Zhang. *An end-to-end Neural Network Framework for Text Clustering*. en. arXiv: 1903.09424. Mar. 2019. URL: http://arxiv.org/abs/1903.09424 (visited on 07/05/2020).

[Zha20] Yue Zhao. *yzhao062/anomaly-detection-resources*. original-date: 2018-05-16T20:02:54Z. Feb. 2020. URL: https://github.com/yzhao062/anomaly-detection-resources (visited on 02/19/2020).

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Potsdam, 10. Juli 2021 Philipp Schoneville