Mining Text Outliers in Document Directories

Edouard Fouché*, Yu Meng†, Fang Guo†, Honglei Zhuang†, Klemens Böhm*, Jiawei Han†
*Karlsruhe Institute of Technology (KIT), Germany
†University of Illinois at Urbana-Champaign, USA
*{edouard.fouche, klemens.boehm}@kit.edu
†{yumeng5, fangguo1, hzhuang3, hanj}@illinois.edu

***Abstract*—Nowadays, it is common to classify collections of documents into (human-generated, domain-specific) directory structures, such as email or document folders. But documents may be classified wrongly, for a multitude of reasons. Then they are outlying w.r.t. the folder they end up in. Orthogonally to this, and more specifically, two kinds of errors can occur: (O) Out-of-distribution: the document does not belong to any existing folder in the directory; and (M) Misclassification: the document belongs to another folder. It is this specific combination of issues that we address in this article, i.e., we mine text outliers from massive document directories, considering both error types.**

We propose a new proximity-based algorithm, which we dub kj-Nearest Neighbours (kj-NN). Our algorithm detects text outliers by exploiting semantic similarities and introduces a self-supervision mechanism that estimates the relevance of the original labels. Our approach is efficient and robust to large proportions of outliers. kj-NN also promotes the interpretability of the results by proposing alternative label names and by finding the most similar documents for each outlier. Our real-world experiments demonstrate that our approach outperforms the competitors by a large margin.

***Index Terms*—Text Mining, Anomaly Detection, Data Cleaning, Document Filtering, Nearest-Neighbour Search.**

I. Introduction

A. Motivation

Recent technological developments have led to an ever-growing number of applications producing, sharing and managing text data. Document repositories, such as email accounts, digital libraries or medical archives, have grown so large that it is now a necessity to classify elements into categories, e.g., folders. This is far from trivial, as text corpora tend to be highly multi-modal, i.e., there are many classes/folders. Documents also may have ambiguous semantics or more than one semantic focus, so that they do not perfectly fit into just one class.

Humans can order content to some extent and routinely do so: Users organise their emails into folders, authors classify their contributions into existing taxonomies, and physicians issue diagnoses as part of their duties. However, human classification is inherently sloppy and unreliable. Humans may classify an email into an inadequate folder, assign scientific papers to the wrong field, or – even more critical – issue a wrong diagnosis.

Sometimes one may not even notice these errors because the correct class is unknown. For instance, an email does not fit into any existing folder, a paper belongs to an emerging field, or a physician observes a new disease.

Detecting documents classified erroneously is difficult [1], [2]. This is because folder structures typically are domain-specific and are user-defined taxonomies. At the same time, documents can be misplaced in numerous, unforeseeable ways, which may or may not be folder-specific. So seeing this problem as a supervised one – where a ground truth is available – would be inadequate. Next, orthogonally to this 'semantic' level, two types of errors/outliers can occur:

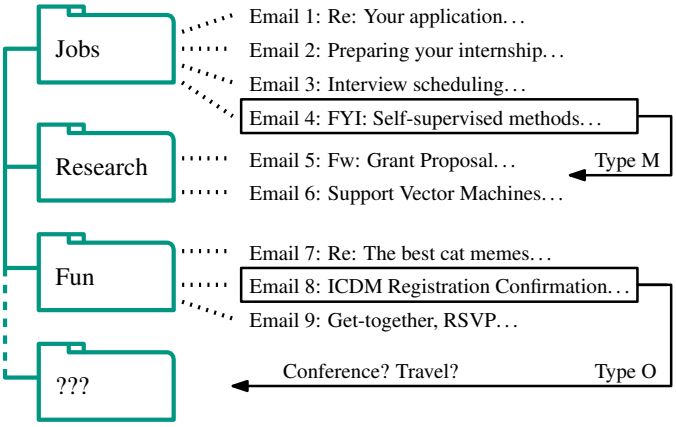

Figure 1. Email Archiving: An Illustrative Example.

- **(O) Out-of-distribution**: A document does not belong to any existing folder. (The user should create a new class.)
- **(M) Misclassification**: A document belongs to another folder in the directory. (The user misclassified it.)

We illustrate this in Figure 1, with (fictitious) emails ordered into folders by a user: Email 4 is a Type M outlier, as it belongs to the folder 'Research'. Email 8 in turn is a Type O outlier, because it does not fit into any existing folder — we should create a new one. Intuitively, a document is a Type O outlier when it does not appear to be similar to documents of any single class. In contrast, a document is a Type M outlier when it appears to be most similar to documents from another class.

In this paper, we focus on mining text outliers in document directories. It is difficult because documents are outlying w.r.t. their semantics, which is not easy to capture (even for humans). An additional issue, which makes this current article unique, is that both outlier types must be detected jointly. Namely, the existence of one type harms the detection of the other one. The noise introduced by Type M outliers hinders the detection of Type O outliers. Conversely, Type O outliers may be detected as Type M by mistake, leading to a poor treatment of such outliers. Existing methods only deal with one of these outlier

types, cf. Section II. Interestingly, we will see that the joint detection of both outlier types leads to better performance for each type than approaches only dealing with one of them.

B. Contributions

While tools exploiting the semantics of text data have improved tremendously in recent years – in particular with the success of embedding methods [3] – text outlier detection has not been sufficiently addressed concerning the challenges just mentioned. Thus, we make the following contributions:

We explore the problem of text outlier detection in document directories. The task is challenging because text can be outlying in numerous ways, and directories are domain-specific. At the same time, text outliers fall into two categories: Types O/M. To our knowledge, we are first to propose an integrated outlier detection framework for text documents that builds on this conceptual distinction.

We propose a new approach to detect text outliers, which we name kj-Nearest Neighbours (kj-NN). Our approach leverages similarities of documents and phrases, based on state-of-the-art embedding methods [4], to detect both Type O/M outliers. By extracting semantically relevant labels and the documents similar to each outlier, it also supports interpretability.

We introduce a 'self-supervision' mechanism to make our approach more robust. Our idea is to weigh each decision by the relevance of neighbouring documents. A document is said to be relevant w.r.t. a given class when its semantics, characterised by its closest phrases in the embedding space, is representative of its class.

We conduct extensive experiments to compare our method to competitors and provide example outputs. The experiments show that our approach improves the current state of the art by a large margin while delivering interpretable results.

We release our source code on GitHub[1], together with our benchmark data, to ensure reproducibility.

Paper Outline: We review the related work in Section II and introduce our notation in Section III. Section IV outlines our framework, and Section V describes our algorithm for text outlier detection. We detail our experiments and results in Sections VI and VII. We conclude in Section VIII.

II. Related Work

To our knowledge, none of the existing methods handles both outlier types. So we categorise related work into two classes: (1) Type O and (2) Type M outlier detectors.

Type O outlier detectors. Type O is the standard definition of outliers, and detecting such outliers has been studied for decades. Conventional outlier detection approaches typically fall into the following classes: distance-based [5], [6], neighbour-based [7], [8], probabilistic-based [9], [10] and subspace-based methods [11]–[15]. Examples of conventional methods are the well-known Local Outlier Factor (LOF) [7] or, more recently, Randomised Subspace Hashing (RS-Hash) [11]. However, textual data is typically extremely sparse, and thus few of the above proposals can be directly extended to detect outlier documents, as they do not model semantics.

There exist a few methods addressing text outliers: [16] proposes a generative approach, which models the embedding space as a mixture of von Mises-Fisher (vMF) distributions. They identify 'outlier regions' that deviate from the majority of the embedded data. [17] proposes TONMF, a Non-negative Matrix Factorisation (NMF) approach based on block coordinate descent optimisation. Recently, [18] proposes Context Vector Data Description (CVDD), a one-class classification model leveraging pre-trained language models and a multi-head attention mechanism. However, all of these methods treat outlier detection as a one-class classification problem, i.e., they try to describe an 'abnormal' class and a 'normal' class; none of them addresses Type M outliers.

Type M outlier detectors. Type M outliers represent misclassified text. While this type of outlier is ubiquitous, it has received little attention so far. Few publications try to address the problem directly. Traditional supervised text classification methods, assuming the ground truth to be error-free, have no choice but to tolerate Type M outliers during training. While one can extend the existing supervised document classification models (e.g., [19]–[22]) to mitigate the effect of Type M outliers, they are in turn not helpful to detect Type O outliers.

As we explained earlier, the existence of both outlier types calls for methods that can detect them simultaneously. In that respect, our method is the first of its kind. In our experiments, we compare our approach against the methods above and show that we outperform all of them. We refer the reader to [23] for an extensive overview of existing outlier detection methods.

III. Notations

Let $D = \{d_1, d_2, \ldots, d_{|D|}\}$ and $P = \{p_1, p_2, \ldots, p_{|P|}\}$ be a set of documents and phrases. We define $\mathcal{O} = D \cup P$ as the set of all text objects.

A common practice in the community is to project each text object in an n-dimensional embedding space as a preprocessing step. In this paper, we use a recent technique [4] capable of projecting both words and documents into the same embedding space, i.e., each object o has an embedding vector representation $V : \mathcal{O} \mapsto \mathbb{R}^n$ with n dimensions, with typically $n \geq 100$. The vectors are all normalised, thus $\|V(o)\| = 1,\ \forall o \in \mathcal{O}$.

We quantify the similarity between a pair of phrases, documents or phrases/documents with a function $S : \mathcal{O}^2 \mapsto [0, 1]$ where 1 is the highest possible similarity (identity) and 0 is the lowest one. S is the normalised cosine similarity:

$$S(V(o_i), V(o_j)) = \frac{V(o_i) \cdot V(o_j) + 1}{2} \quad \forall (o_i, o_j) \in \mathcal{O}^2. \quad (1)$$

For simplicity, we equivalently refer to the vectorial representation of each object, i.e., $o \equiv V(o)$, in what follows. We also assume that there exists an initial classification of documents into a finite set of classes $C = \{c_1, \ldots, c_{|C|}\}$, expressed as a function $y : D \mapsto C$.

Our self-supervision mechanism relies on estimating the *representativeness* of each phrase $p \in P$ w.r.t. each class $c \in C$. We denote it as a function $r : P \times C \mapsto \mathbb{R}^+$.

[1] https://github.com/edouardfouche/MiningTextOutliers

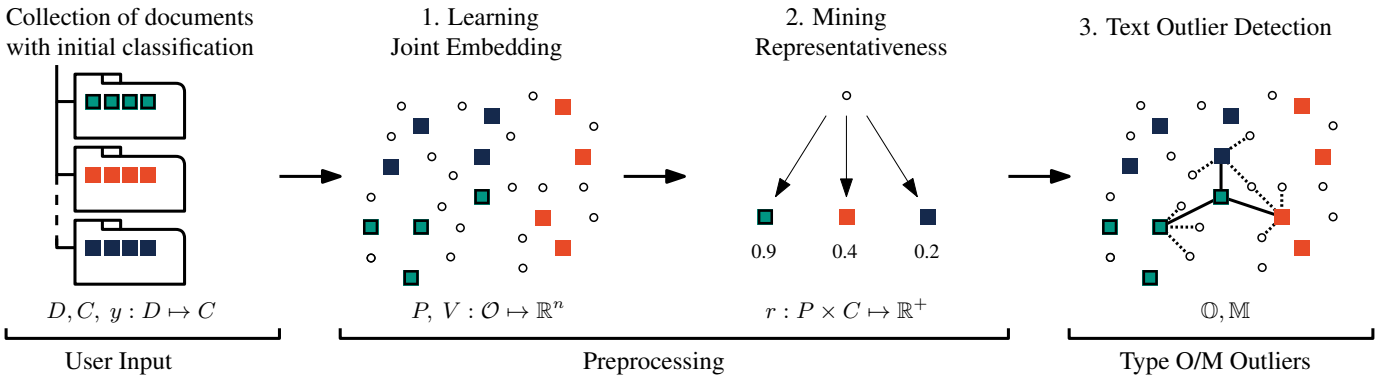

Figure 2. Our Framework: A High-Level Overview. In the illustrations, squares are documents and dots are phrases.

IV. Our Framework

Figure 2 provides a high-level overview of our framework. We assume as input a collection of documents D, with an initial but imperfect classification $y : D \mapsto C$ provided by the user. Squares stand for documents, and the colours represent their initial class. Documents are composed of phrases, represented in turn as dots. We design our framework in three steps:

1. Learning Joint Embedding: Text outlier detection relies on textual similarity/dissimilarity measures, which can be effectively captured by embeddings. We segment the phrases P from every document using AutoPhrase [24] and obtain the phrase and document embeddings $V : \mathcal{O} \mapsto \mathbb{R}^n$ in a joint spherical space via Joint Spherical Embedding (JoSE) [4].

The benefits of using JoSE are twofold: (1) JoSE trains document and phrase embeddings jointly in the same space, where text similarity between phrases and documents can be directly derived. (2) JoSE captures directional similarity by training in the spherical space, which characterises textual similarity more effectively than Euclidean embeddings.

2. Mining Representativeness: We estimate the representativeness of each phrase for each class, based on the initial classification. The representative phrases for a class are indicative of the semantics of documents. As in [25], we define the representativeness as a function of three criteria:

- **Integrity:** A phrase with high integrity in a given corpus is meaningful, understandable and of high quality.
- **Popularity:** A phrase is popular in a given class if it has many occurrences.
- **Distinctiveness:** A phrase is distinctive if it distinguishes a class from the other classes.

For each phrase $p \in P$ and class $c \in C$, we estimate the integrity $int(p, c) \in [0, 1]$, popularity $pop(p, c) \in \mathbb{R}^+$ and the distinctiveness $disti(p, c) \in [0, 1]$ as described in [26]. The representativeness is the product of those three criteria:

$$r(p, c) = int(p, c) \cdot pop(p, c) \cdot disti(p, c). \tag{2}$$

The idea is that, even if each class may contain ambiguous documents or erroneous labels, these are 'rare', so the impact on the estimation of representativeness is low. Our experiments show that our method is robust against erroneous labels.

3. Text Outlier Detection: We introduce kj-NN, an outlier detector inspired by the well-known k-NN classifier [27]. The main novelty is that inferring the class of each element (document) does not only base on the class of its k nearest documents but also on their relevance. We estimate the relevance of a document as the average representativeness of its j nearest sub-elements (phrases) for its class. If a document d is closer to documents which are (i) relevant and (ii) from another single class, then d likely is a Type M outlier. On the other hand, if d is similarly close to relevant documents of various classes, then d likely is a Type O outlier. This final step yields two ranked lists $\mathbb{O}$ and $\mathbb{M}$ of outliers for Type O and Type M, respectively.

In our approach, self-supervision consists of estimating the relevance of the original label, recognising that less relevant labels must have a lower influence on our predictions. The rationale is to perceive irrelevant documents as such because they either were misclassified or have ambiguous semantics.

In the next section, we present the technical details of our outlier detector. We first provide a Bayesian formalisation, then describe the practical implementation of kj-NN.

V. Text Outlier Detection with kj-NN

A. Formalisation

We define the k nearest documents and the j nearest phrases of a document $d \in D$ as follows:

$$\mathcal{K}(d) = \{x \in D \backslash d : |\{x' \in D \backslash d : S(x', d) > S(x, d)\}| < k\},$$
$$\mathcal{J}(d) = \{x \in P : |\{x' \in P : S(x', d) > S(x, d)\}| < j\}.$$

We call $\mathcal{K}(d)$ and $\mathcal{J}(d)$ the k- and j-neighbourhood of d. Then one can build a classifier based on local densities. For each document $d \in D$, the k-neighbourhood provides an estimate of the density within each class, and the j-neighbourhood provides a pseudo-posterior probability for each neighbour.

Bayesian inference formulates the posterior probability of class membership of document $d \in D$ as follows:

$$\Pr(c|d) = \frac{\Pr(d|c)\Pr(c)}{\Pr(d)}. \tag{3}$$

Now think of a sphere of volume v centred at d that contains k points. Then we can express the likelihood as

$$\Pr(d|c) = \frac{|\mathcal{K}_c(d)|}{|D_c|v}, \tag{4}$$

where $\mathcal{K}_c(d)$ is the set of documents in $\mathcal{K}(d)$ of class c, and D_c is the set of all documents of class c. The class prior is

$$\Pr(c) = \frac{|D_c|}{|D|}. \tag{5}$$

Since v, $|D|$ and $\Pr(d)$ are independent of c, we have

$$\Pr(c|d) \propto |\mathcal{K}_c(d)|. \tag{6}$$

To minimise the misclassification probability, one must assign d to the class with the highest density in its neighbourhood.

This only works under the assumption that all labels are correct, which is unrealistic in our setting. Our idea is to weight each document d' in $\mathcal{K}_c(d)$ by a pseudo-posterior probability $\widehat{\Pr}(c|d')$, capturing our belief that they indeed are of class c:

$$\Pr(c|d) \propto \sum_{d'}^{\mathcal{K}_c(d)} \widehat{\Pr}(c|d'), \tag{7}$$

where we define $\widehat{\Pr}(c|d')$ to be proportional to the representativeness $r(p,c)$ of the phrases $p \in \mathcal{J}(d')$ for class c:

$$\widehat{\Pr}(c|d') \propto \sum_{p}^{\mathcal{J}(d')} r(p,c). \tag{8}$$

With this specification of $\widehat{\Pr}(c|d')$, we exploit additional information from the j-neighbourhood as 'self-supervision signals'. In contrast, the standard k-NN classification rule assumes that $\Pr(c|d') = 1, \forall d' \in \mathcal{K}_c(d)$, i.e., the labels are accurate. Finally, predicting the class of d boils down to

$$\hat{y}(d) = \arg\max_{c \in C} \Pr(c|d) = \arg\max_{c \in C} \sum_{d'}^{\mathcal{K}_c(d)} \sum_{p}^{\mathcal{J}(d')} r(p,c). \tag{9}$$

Whenever $\hat{y}(d) \neq y(d)$, we may declare that a user misclassified document d, i.e., it is a Type M outlier. However, the reliability of such predictions may vary widely. For example, if each class has a very similar posterior probability, deciding for one or the other might not be meaningful. When a document does not prominently belong to any existing class, we must declare a Type O outlier. We quantify the reliability of a prediction via the entropy of the posterior probabilities, which measures the uncertainty of the prediction:

$$I(d) = -\sum_{c \in C} \Pr(c|d) \cdot \log \Pr(c|d). \tag{10}$$

We obtain the posterior probabilities via normalisation:

$$\Pr(c|d) = \frac{\sum_{d'}^{\mathcal{K}_c(d)} \sum_{p}^{\mathcal{J}(d')} r(p,c)}{\sum_{c}^{C} \sum_{d'}^{\mathcal{K}_c(d)} \sum_{p}^{\mathcal{J}(d')} r(p,c)}. \tag{11}$$

We decide whether a prediction is uncertain using a threshold $\Gamma > 0$ that we set to a percentile p^* of the empirical distribution of the entropy for every document in the corpus:

$$\Gamma > 0 \quad s.t. \quad \frac{|\{d \in D \ : \ I(d) < \Gamma\}|}{|D|} = p^*. \tag{12}$$

In other words, our idea is to declare that $p^*\%$ of the documents with the most uncertain predictions are Type O outliers. For the remaining $1 - p^*\%$ documents, we declare that they are Type M outliers if $\hat{y}(d) \neq y(d)$.

B. Implementation

A well-known caveat of neighbour-based classifiers is that they tend to be sensitive to the choice of parameter k. [28] first proposed to weigh each neighbour by the distance to the queried point. There exist many weighting schemes [29], [30]. While finding the best scheme is out of our scope, the consensus is that weighting improves empirical performance and leads to more flexible parameter choice (see [31]). So we propose the following score:

$$score_{d,c} = \sum_{d'}^{\mathcal{K}_c(d)} S(d,d') \sum_{p}^{\mathcal{J}(d')} S(d',p) \cdot r(p,c), \tag{13}$$

which uses both the inter-document and the document-phrase similarities for weighting. By definition, if $\mathcal{K}_c(d) = \emptyset$, then $score_{d,c} = 0$. We compute the entropy as follows:

$$I(d) = -\sum_{c}^{C} \overline{score}_{d,c} \cdot \log \overline{score}_{d,c} \tag{14}$$

where $\overline{score}_{d,c}$ is the normalised score over the classes $c \in C$. By convention, $\overline{score}_{d,c} \cdot \log \overline{score}_{d,c} = 0$ if $\overline{score}_{d,c} = 0$. Finally, the outcome is a list of outliers for each type:

$$\mathbb{O} = \langle d_i, d_j, \ldots, d_{|\mathbb{O}|} \rangle \quad s.t. \quad I(d_i) > \Gamma \wedge I(d_i) \geq I(d_j),$$
$$\mathbb{M} = \langle d_i, d_j, \ldots, d_{|\mathbb{M}|} \rangle \quad s.t. \quad I(d_i) \leq \Gamma \wedge \hat{y}(d_i) \neq y(d_i) \ldots$$
$$\ldots \wedge I(d_i) \leq I(d_j), \quad \forall i < j, (i,j) \in D^2$$

Here, Γ is set by parameter p^* (see Eq. 12). The prediction is:

$$\hat{y}(d) = \arg\max_{c \in C} score_{d,c} \tag{15}$$

Note that we sort $\mathbb{O}$ by decreasing uncertainty, while we sort $\mathbb{M}$ by increasing uncertainty. The rationale is that the more uncertain the decision for document d, the more likely d is a Type O outlier. On the other hand, the less uncertain a misclassification, the more likely d is a Type M outlier. So we output a ranking of outliers for both. In particular, if users only have a limited amount of time, they may only examine the most 'flagrant' outliers.

Algorithm 1 is our approach as pseudo-code. Since vectors are normalised, the normalised cosine similarity S (cf. Eq. 1) is proportional to the euclidean distance. Thus we can use R* trees to speed up the neighbourhood queries, and we cache the results of the individual queries for each document. From our algorithm, it is easy to see that the complexity of the approach is quasi-linear w.r.t. $|D|$, $|P|$, $|C|$, k and j, i.e., it can scale to very large corpora.

Algorithm 1 KJ-NN(k, j, D, P, y, r, p^*)

Require: k, j, corpus D, phrases P, initial classification $y : D \mapsto C$, representativeness $r : P \times C \mapsto \mathbb{R}^+$, threshold $p^* \in [0, 1]$

$\mathbb{O} = \langle\rangle$; $\mathbb{M} = \langle\rangle$ ▷ Initialisation
$K \leftarrow$ index D with an R*-tree
$J \leftarrow$ index P with an R*-tree
for $d_i \in D$ **do** ▷ Cache neighbourhood queries
  $\mathcal{K}(d_i) \leftarrow$ the k nearest neighbours of d_i in K
  $\mathcal{J}(d_i) \leftarrow$ the j nearest neighbours of d_i in J
for $d_i \in D$ **and** $c \in C$ **do** ▷ Get score and entropy
  $\mathcal{K}_c(d_i) \leftarrow \{d' \in \mathcal{K}(d_i) : y(d') = c\}$
  $score_{d_i,c} = \sum_{d'}^{\mathcal{K}_c(d_i)} S(d_i, d') \sum_{p}^{\mathcal{J}(d_i)} S(d', p) \cdot r(p, c)$
  $I(d_i) = \sum_c^C \overline{score}_{d_i,c} \cdot \log \overline{score}_{d_i,c}$
Choose Γ *s.t.* $|\{d_i \in D \mid I(d_i) < \Gamma\}|/|D| = p^*$
Sort D by increasing $I(d_i)$, $d_i \in D$
for $d_i \in D$ **do** ▷ Populate outlier lists
  if $I(d_i) > \Gamma$ **then** $\mathbb{O} \leftarrow \langle d_i \rangle \cup \mathbb{O}$
  else if $\hat{y}(d_i) \neq y(d_i)$ **then** $\mathbb{M} \leftarrow \mathbb{M} \cup \langle d_i \rangle$
return $\mathbb{O}$, $\mathbb{M}$

VI. Experiment Setup

We evaluate the performance of our approach w.r.t. both types of outliers. We compare with the current state of the art, as well as with competitive baselines and ablations. We create real-world benchmark data sets from publicly available data.

A. Evaluation Measures

Outlier detection typically is an imbalanced classification problem. For Type O outliers, we report the area under the ROC curve (AUC) and the average precision (AP). These are popular measures for the evaluation of outlier detection algorithms. Since Type M outliers are comparably more frequent, we report precision (P), recall (R) and the F1 score. In most applications, users are more concerned with the recall [16]. So we also measure the *recall at a certain percentage*, i.e., the share of detected outliers when the user checks the top X% items (RX) from the ranked list of outliers.

B. Data Sets

We evaluate our approach against an assortment of benchmark data sets of various size, outlier ratio and number of inlier/outlier classes. Since emails and medical records typically contain highly confidential information, they are not adequate for the reproducibility of our study. Instead, we create the following sets of benchmarks from publicly available news articles and paper abstracts:

- **NYT:** We crawl 10,000 articles from 5 topics (*Business*, *Sports*, *Arts*, *Politics*, *Science*) with the New York Times API and add 1% articles (i.e., 100 articles) from 4 other topics (*Real estate*, *Health*, *Education*, *Technology*), i.e., they are Type O outliers. We also increase the ratio of outliers to 2% and 5% and downsample the data by 50%, 20% and 10%. Thus, we create six benchmark data sets: **NYT-1**, **NYT-2**, **NYT-5**, **NYT-50**, **NYT-20**, **NYT-10**.
- **ARXIV:** We crawl 21,467 abstracts of the articles published on ArXiv from 10 computer science categories (*cs.AI*, *cs.CC*, *cs.CL*, *cs.CR*, *cs.CY*, *cs.DB*, *cs.DS*, *cs.LG*, *cs.PL*, *cs.SE*). Then we choose 1 to 5 inlier classes at random and inject 1% outliers from 5 other classes. We repeat the procedure, but let the number of outlier classes vary. In the end, we create nine benchmark data sets: **ARXIV-15**, **ARXIV-25**, **ARXIV-35**, **ARXIV-45**, **ARXIV-55**, **ARXIV-54**, **ARXIV-53**, **ARXIV-52**, **ARXIV-51**.

We detail in Appendix the characteristics of each benchmark. As mentioned, we release the data sets with our source code.

C. Baselines and Competitors

Since none of the existing approaches detects both Type O and Type M outliers, we must compare with two different sets of competitors/baselines. To validate our design choices, we also compare against a set of ablations derived from our method (see in Appendix).

1) Type O: We compare kj-NN w.r.t. Type O outlier detection against the following baselines and competitors:

- **Local Outlier Factor (LOF)** [7] is a well-known density-based outlier detector. We report the best result with parameter $k \in [1, 100]$ in terms of AUC, as performance may vary widely w.r.t. k [32]. We use the implementation from ELKI [33].
- **Randomised Subspace Hashing (RS-Hash)** [11] is a subspace outlier detector. It estimates the outlierness of each data point via randomised hashing. We implement it as described by the authors, and we use the recommended parameters.
- **Average Negative Cosine Similarity (ANCS)** is a baseline measuring the outlierness of a document as the average negative cosine similarity to every other document. We also propose **k-ANCS**, a variant which only uses the k nearest neighbours for each document. We select $k \in [1, 100]$ maximising the AUC.
- **VMF-Q** [16] models word embeddings as a vMF mixture and penalises lexically general words to identify semantically deviating documents. We use the implementation provided by the authors with the recommended parameters.
- **TONMF** [17] uses Non-negative Matrix Factorisation from a bag-of-word representation to detect documents with unusual word frequencies. We use the implementation released by the authors and let the parameters k, α and β vary from 1 to 30. We report the best result in terms of AUC.
- **CVDD** [18] is a one-class classification model using a pre-trained language model and multi-head attention. We use the authors' implementation with the recommended parameters.

2) Type M: To our knowledge, there is no approach explicitly handling the detection of misclassifications (Type M outliers). However, we can adapt virtually any supervised classifier to this task. We compare against the following state-of-the-art approaches for text classification:

- **Word-level CNN (W-CNN)** [19] applies convolution kernels on stacked word embedding matrix of a document followed by pooling operations.
- **Very Deep CNN (VD-CNN)** [20] uses up to 29 convolution layers that perform feature learning starting from characters, aiming to capture the hierarchical semantic structure encoded in characters, n-grams, words and sentences.

- **Attention-Based Hierarchical RNN (AT-RNN)** [34] employs attention mechanisms both at the word and sentence level. It learns to focus on the most relevant words and sentences for text classification.
- **RCNN** [22] combines both bi-directional recurrent structures and max-pooling layers to capture contextual information and extract relevant features.

For each approach, we train the classifier and predict the class of each instance. If the prediction differs from the actual class, we conclude that it is of Type M. The rationale is that such instances do not fit their class as well as other instances. Note that this is a rather common approach for outlier detection.

Since one typically does not have any ground truth in outlier detection tasks, performing a hyper-parameter search is not realistic and falls out of the scope of our study. We run each approach with default parameters.

D. Data Preparation

Since every Type O methods are unsupervised, they operate without labels. To evaluate the approaches, we set the labels of inliers to 0 and the labels of Type O outliers to 1.

LOF, RS-Hash, ANCS and k-ANCS use as input the embedding representation that we mine in our preprocessing step (cf. Section IV). For VMF-Q, TONMF and CVDD, we implement the preprocessing steps recommended by their respective inventors.

For Type M methods, we control the proportion of Type M outliers by randomly assigning a proportion $b \in [0, 0.5]$ of labels to a wrong class, i.e., when $b = 0.5$, we misclassify half of the documents. The algorithms train with those erroneous labels, but we use the original labels for evaluation. We assign Type O outliers to an inlier class at random.

For each method, we first tokenise and segment the raw text data using AutoPhrase [24]. We learn the joint embeddings via JoSE [4] with $n = 100$ dimensions. Overall, we process the input data similarly between every method, taking into account the recommendation of the respective authors. Our goal is to make the comparison as fair as possible.

VII. Results

We perform our experiments on a machine running Ubuntu 18.04 with 20GB RAM and a quad-core CPU at 2.40GHz. We average each of our results from 10 independent runs.

A. Parameter Sensitivity

We first study the sensitivity of our approach to its parameters k, j and p^* against the benchmark NYT-1, w.r.t. varying Type M outlier ratio b in particular. We first simulate an extreme scenario by setting $b = 0.5$, i.e., we assign half of the articles to the wrong topic. We can see from Figure 3 that the Type O recall and precision tend to increase with k and j, but saturate for $j > 30$ and $k > 30$. We also see that p^* captures a trade-off between precision and recall: higher values of p^* leads to higher Type O precision, but lower recall. On the other hand, higher values of p^* lead to higher Type M recall, but lower precision. We see that the Type M precision and recall are

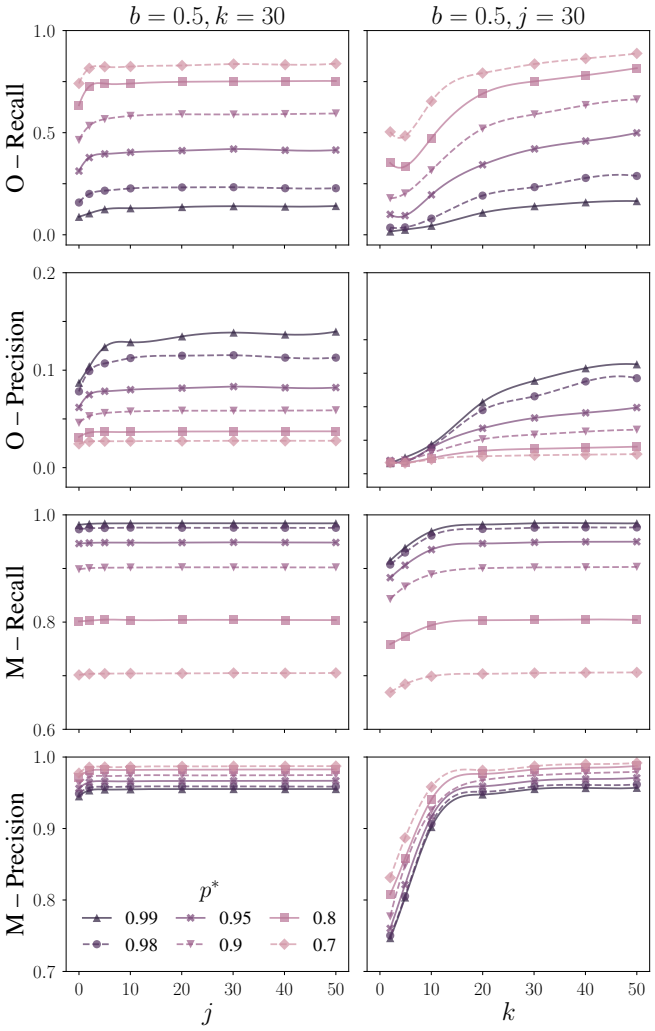

Figure 3. Type O/M — k, j, p^* sensitivity, NYT-1.

high for $p^* \geq 0.9$ for large enough k and j. Given that the initial classification is very noisy ($b = 0.5$), the quality of the detection of both outlier types is impressive.

Next, we set $k = j = 30$ and let b vary from 0 to 0.5. In Figure 4, we see that the Type O recall and precision improve significantly for lower b, but the Type M precision decreases. While imperfect labels negatively impact the quality of Type O outlier detection, the Type M recall appears stable.

Based on our observations, we recommend setting $k = j = 30$ and $p^* = 0.9$. For the remaining of our experiments, we set the parameters as such and assume $b = 0.2$.

B. Performance Comparison

1) Type O: Tables I and V (in Appendix) list the results of our comparison against the NYT and ARXIV benchmark.

First, the results of every approach are generally better against the NYT benchmark. The ARXIV benchmark is much more challenging because the corpus is composed of abstracts from computer science sub-fields, with much semantic overlap.

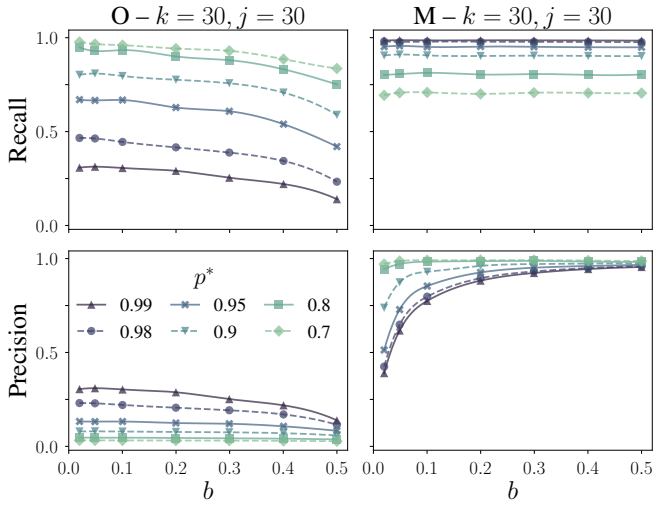

Figure 4. Type O/M — p^*, b sensitivity, NYT-1.

Table I
COMPARISON W.R.T. OUR COMPETITORS (TYPE O, NYT).

		AUC	AP	R1	R2	R5		AUC	AP	R1	R2	R5
LOF	NYT-1	66.45	1.62	3.00	3.00	9.00	NYT-50	60.91	1.77	2.00	6.00	12.00
RS-Hash		46.62	0.87	0.00	1.00	1.00		46.14	0.90	0.00	0.00	2.00
ANCS		66.63	2.57	6.00	10.00	21.00		63.94	4.35	14.00	16.00	20.00
k-ANCS		83.89	3.65	3.00	7.00	19.00		81.08	4.43	12.00	14.00	20.00
TONMF		58.66	7.30	0.00	2.00	9.00		61.42	**31.01**	0.90	0.90	4.90
VMF-Q		76.34	2.21	2.00	3.00	11.00		85.76	4.00	6.00	8.00	22.00
CVDD		78.47	7.75	11.7	18.09	22.34		76.29	15.89	21.62	27.03	29.73
kj-NN		**92.51**	**17.57**	**25.00**	**39.20**	**61.40**		**93.33**	27.76	**37.20**	**46.80**	**62.80**
LOF	NYT-2	60.66	2.45	2.00	2.50	6.00	NYT-20	74.76	4.12	5.00	5.00	15.00
RS-Hash		48.05	1.87	0.50	1.00	5.50		42.91	0.89	0.00	0.00	0.00
ANCS		67.59	5.19	8.00	12.00	18.00		78.52	5.60	10.00	15.00	**40.00**
k-ANCS		82.51	5.94	3.00	5.50	14.50		88.56	6.59	15.00	20.00	30.00
TONMF		54.61	1.78	2.50	3.00	9.00		64.94	6.35	0.00	0.00	15.00
VMF-Q		83.67	6.87	4.00	11.00	20.00		83.98	4.59	5.00	10.00	20.00
CVDD		73.10	10.00	11.58	14.74	22.11		88.83	**24.00**	**25.00**	**25.00**	25.00
kj-NN		**94.51**	**42.64**	**30.40**	**44.50**	**64.50**		**91.42**	6.57	3.00	11.00	38.00
LOF	NYT-5	52.28	5.44	1.80	3.40	7.00	NYT-10	77.93	2.77	0.00	0.00	20.00
RS-Hash		48.76	4.58	0.80	1.40	2.80		56.94	1.76	0.00	0.00	10.00
ANCS		67.67	11.13	6.60	9.80	19.20		83.89	13.65	30.00	40.00	40.00
k-ANCS		75.56	10.45	3.40	6.40	11.40		91.37	10.93	30.00	30.00	30.00
TONMF		52.95	1.50	1.40	2.40	6.40		71.13	29.92	0.00	0.00	0.00
VMF-Q		77.11	12.92	4.60	7.40	15.60		63.39	2.55	0.00	10.00	20.00
CVDD		72.81	18.69	9.04	13.05	21.49		85.13	**44.99**	**42.86**	**42.86**	**42.86**
kj-NN		**97.04**	**71.69**	**19.12**	**36.96**	**68.28**		**91.52**	8.45	10.00	16.00	38.00

Then, we see from both tables that kj-NN outperforms every competitor w.r.t. Type O outlier detection. The performance in terms of recall becomes lower for smaller data sets (e.g., NYT-20 and NYT-10). For small data sets, CVDD and our ANCS/k-ANCS baselines appear competitive.

Finally, we see that our approach handles particularly well multi-modal settings, i.e., with multiple inlier/outlier classes. By design, our approach cannot handle only one inlier class (e.g., ARXIV-15) — this does not fit our scenario. When there only is a single outlier class, LOF performs best (see ARXIV-51).

2) Type M: Our approach also outperforms its competitors w.r.t. Type M outliers. See Tables II and VI. RCNN and VD-CNN have high precision, but much lower recall. Neural networks tend to fit the data very well, including Type M outliers. The performance decreases dramatically with smaller data sets. VD-CNN occasionally ranks high in terms of F1-score. However, those approaches do not rank outliers, so the recall at a certain percentage (e.g., R10, R20) is equivalent to that from a list of detected outliers in random order.

Table II
COMPARISON W.R.T. OUR COMPETITORS (TYPE M, NYT).

		P	R	F1	R10	R20		P	R	F1	R10	R20
W-CNN	NYT-1	54.38	86.04	66.64	27.28	53.51	NYT-50	47.38	87.61	61.50	22.28	47.62
VD-CNN		90.71	69.22	78.52	15.04	30.18		**98.58**	55.44	70.97	49.31	54.95
AT-RNN		67.12	51.88	58.52	32.92	51.34		89.75	69.22	78.16	14.83	29.92
RCNN		**96.02**	9.65	17.54	9.55	9.55		57.46	87.31	69.31	27.23	56.93
kj-NN		95.93	**90.02**	**92.88**	**50.19**	**90.02**		95.78	**90.36**	**92.99**	**50.22**	**90.36**
W-CNN	NYT-2	54.16	88.02	67.06	27.30	54.56	NYT-20	39.34	**91.56**	55.03	20.54	40.59
VD-CNN		88.99	69.69	78.17	14.71	29.61		93.50	81.80	87.26	15.20	30.81
AT-RNN		80.36	49.03	60.90	40.29	48.14		50.44	85.11	63.34	25.25	52.23
RCNN		89.60	5.59	10.52	5.49	5.49		39.63	**91.56**	55.32	20.54	40.59
kj-NN		**94.63**	**91.15**	**92.86**	**50.74**	**91.15**		**93.80**	90.64	**92.19**	**50.52**	**90.64**
W-CNN	NYT-5	51.12	89.07	64.96	25.14	50.38	NYT-10	36.12	88.94	51.38	16.83	35.15
VD-CNN		58.21	82.39	68.22	8.98	18.46		86.20	69.40	76.89	14.10	27.26
AT-RNN		61.71	70.49	65.81	31.38	61.62		36.12	88.94	51.38	16.83	35.15
RCNN		90.82	44.93	60.12	42.86	42.86		36.12	88.94	51.38	16.83	35.15
kj-NN		**92.43**	**93.44**	**92.93**	**52.27**	**93.44**		**88.57**	**90.77**	**89.65**	**50.50**	**89.97**

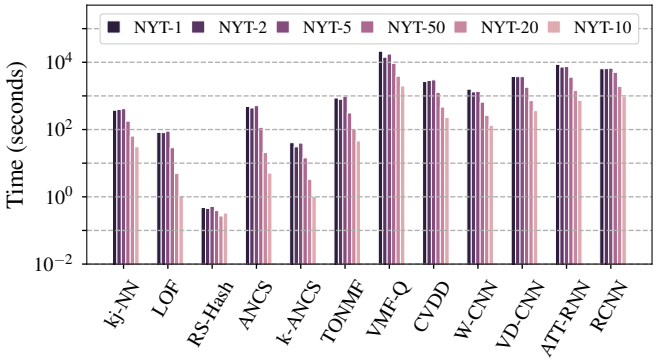

Figure 5. Execution time of each approach.

3) Execution Time: Figure 5 graphs the execution time for each approach. We neglect the common preprocessing time for each of them and report the sum of training and testing time. RS-Hash is extremely fast, but as we saw the performance is not better than guessing. LOF and k-ANCS, which leverage index support, are relatively fast. Our method, kj-NN, is only slightly slower than LOF but seems to scale better (observe the difference in execution time between NYT-1 and NYT-10).

C. Interpretation

The design of our method gives way to interpretable results. To find similar documents, we can simply perform a nearest-neighbour search in the embedding space. We find in turn the most representative phrases for a given outlier d_{out} via a similar approach as in preprocessing, with this time only two classes: An outlier class c_{out} containing the k-neighbourhood of document d, i.e., $\mathcal{K}(d) \cup d$ and a class c_{in} containing the rest of the documents in the corpus.

Figure 6 shows the two top-ranked Type O and Type M outliers we found in NYT-1, along with the most representative phrases and excerpts from the most similar documents.

It is interesting to see that the three nearest neighbours of the Type O outlier either relate to building construction projects or political decisions in education (or both). Actually, the ground

Table III
CHARACTERISTICS OF THE BENCHMARK DATA SETS.

Benchmark	# Inliers	# O	# Classes	# O Classes	O%
NYT-1	10,000	100	5	4	1.00
NYT-2	10,000	200	5	4	2.00
NYT-5	10,000	500	5	4	5.00
NYT-50	5000	50	5	4	1.00
NYT-20	2000	20	5	4	1.00
NYT-10	1000	10	5	4	1.00
ARXIV-51	11,115	111	5	1	1.00
ARXIV-52	11,115	111	5	2	1.00
ARXIV-53	11,115	111	5	3	1.00
ARXIV-54	11,115	111	5	4	1.00
ARXIV-55	11,115	111	5	5	1.00
ARXIV-45	10,136	101	4	5	1.00
ARXIV-35	6299	63	3	5	1.00
ARXIV-25	4619	46	2	5	1.00
ARXIV-15	3267	33	1	5	1.00

truth label '*Education*' even appears at position five within the most representative phrases — a useful information in practice.

With the Type M outlier, the nearest documents all relate to science and business in some way, and the top phrases (energy, fuel, plant, ...) strongly relate to their specific topic.

We then run our approach on the ARXIV benchmark with all the ten classes. Figure 7 shows the top-ranked outliers. The Type O outliers are indeed abnormal: they are not paper abstracts, so one should remove them from the corpus. The Type M outlier is from [35]. The authors chose to publish this article in the category *cs.AI* (Artificial Intelligence), but our approach suggests that it should be under *cs.CL* (Computation and Language). Intuitively, this makes sense, given the general topic of the abstract and the most representative phrases.

VIII. CONCLUSIONS

We have studied the detection of text outliers in document directories. This task is challenging because text outliers are manifold, domain-specific, and the task is unsupervised in nature. We observe that such outliers fall into two types: out-of-distribution (Type O) and misclassified (Type M) documents. We are first to propose an approach that (i) detects text outliers from multiple folders and (ii) effectively distinguishes between both outlier types. Our algorithm, kj-NN, leverages self-supervision signals mined from an initial but imperfect classification. Interestingly, our experiments show that detecting both outlier types simultaneously leads to better performance than with existing approaches, which only deal with one type. Our method also yields results that are interpretable, by finding adequate alternative names for folders and by describing the particular semantics of text outliers.

In the future, we plan to transfer our approach to other domains, e.g., multivariate times series. It would also be interesting to study the performance of our approach for classifying new documents, when having only very few labels.

APPENDIX

Table III shows the features of each benchmark data set, in particular: the number of inliers (# Inliers), Type O outliers (# O), inlier classes (# Classes), Type O outlier classes (# O Classes), and the ratio of Type O outliers (O%).

Additionally, we verify each of our design choices by comparing against the following ablations:

- **A1: No self-supervision**; we set $j = 0$, i.e., our approach boils down to a k-NN classifier as in Eq. 6.
- **A2: No entropy**; we do not consider the entropy of the prediction, i.e., objects can be in both outlier lists $\mathbb{O}$ and $\mathbb{M}$.
- **A3: No neighbourhood**; the predictions only base on the relevance of document d, and not on the relevance of its neighbours, i.e., we set $\mathcal{K}(d) = \{d\}$.
- **A4: Unweighted kj-NN**; we do not weigh the score by the inter-document and document-phrase similarities, as explained in Section V-B. Each neighbouring document and phrase have the same impact on the decision.

From Table IV, we can see that kj-NN consistently outperforms every ablation on average. A2 leads to higher Type M recall, and often high Type O AUC, but much lower Type M precision. A1 and A3 yield worse results for both outlier types. A4 values are consistently below, but only by a few hundredth.

The overall average ranks are as follows:

- **kj-NN:** 1.67, **A2:** 2.50, **A4:** 2.69, **A1:** 3.30, **A3:** 4.29

Thus, we can see that taking the neighbourhood into account improves the performance of our approach the most, followed by self-supervision. Measuring decision uncertainty and weighting by similarity, as in Eq. 13, improves the performance further.

ACKNOWLEDGMENT

This work was supported by the DFG Research Training Group 2153: 'Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation', the German Federal Ministry of Education and Research (BMBF) via Software Campus (01IS17042) and sponsored in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and SocialSim Program No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, and DTRA HDTRA11810026.

REFERENCES

[1] B. Frénay and M. Verleysen, "Classification in the Presence of Label Noise: A Survey," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 5, pp. 845–869, 2014.
[2] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, ser. Intelligent Systems Reference Library. Springer, 2015, vol. 72.
[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *NIPS*, 2013, pp. 3111–3119.
[4] Y. Meng, J. Huang, G. Wang, C. Zhang, H. Zhuang, L. M. Kaplan, and J. Han, "Spherical Text Embedding," in *NeurIPS*, 2019, pp. 8206–8215.
[5] E. M. Knorr and R. T. Ng, "Algorithms for Mining Distance-Based Outliers in Large Datasets," in *VLDB*, 1998, pp. 392–403.
[6] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets," in *SIGMOD*, 2000, pp. 427–438.
[7] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *SIGMOD*, 2000, pp. 93–104.
[8] H. Kriegel, M. Schubert, and A. Zimek, "Angle-Based Outlier Detection in High-dimensional Data," in *KDD*, 2008, pp. 444–452.
[9] J. Kim and C. D. Scott, "Robust Kernel Density Estimation," *J. Mach. Learn. Res.*, vol. 13, pp. 2529–2565, 2012.
[10] M. E. Tipping and C. M. Bishop, "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.
[11] S. Sathe and C. C. Aggarwal, "Subspace histograms for outlier detection in linear time," *Knowl. Inf. Syst.*, vol. 56, no. 3, pp. 691–715, 2018.

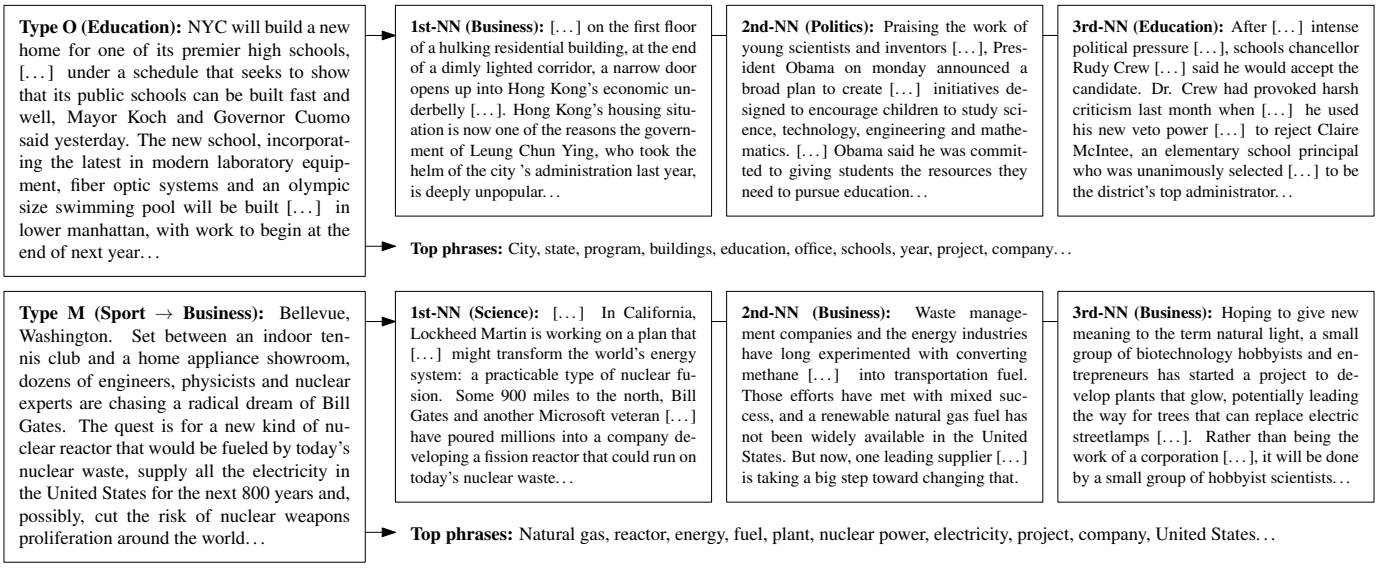

Figure 6. Interpretation of Type O/M outliers (examples from the NYT-1 benchmark). We can see that the Type O outlier relates in some way to its nearest documents — They all relate to building construction projects or political decisions in education (or both). Similarly for the Type M outlier, which was wrongly classified into the folder 'Sport', the nearest documents all relate to science (in particular, energy generation technologies) and business.

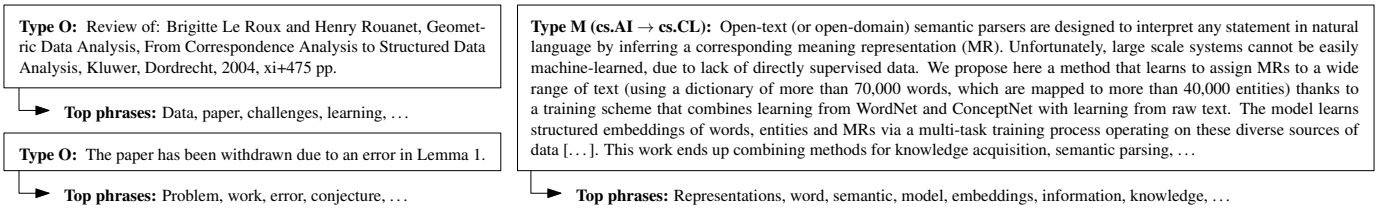

Figure 7. Interpretation of Type O/M outliers (examples from the ARXIV benchmark). We see that the Type O outliers are not paper abstracts, so one should remove them from the data set. The Type M outlier may arguably belong to the category 'Computation and Language', instead of 'Artificial Intelligence'.

Table IV
ABLATION ANALYSIS (NYT & ARXIV) — NOTE THAT ARXIV-15 IS NOT APPLICABLE (NA) HERE.

		Type O		Type M						Type O		Type M						Type O		Type M			
		AUC	AP	P	R	F1	Rank			AUC	AP	P	R	F1	Rank			AUC	AP	P	R	F1	Rank
A1	NYT-1	89.57	16.65	94.56	90.04	92.25	3.6	A1	NYT-10	91.25	8.05	88.18	90.47	89.31	3.6	A1	ARXIV-55	75.84	3.07	70.33	88.24	78.27	3.2
A2		**92.51**	**17.57**	86.43	**99.07**	92.32	2.2	A2		**91.52**	8.45	80.98	**98.50**	88.88	2.6	A2		**76.74**	**3.22**	63.78	**96.82**	76.90	2.4
A3		90.56	8.45	91.85	89.28	90.54	4.6	A3		89.65	14.06	87.30	85.17	86.20	4.6	A3		72.31	2.24	66.06	87.02	75.10	4.8
A4		92.48	17.07	95.88	90.00	92.85	2.8	A4		91.23	**8.49**	88.19	90.57	89.36	2.4	A4		76.60	3.18	70.49	88.19	78.35	2.8
kj-NN		**92.51**	**17.57**	**95.93**	90.02	**92.88**	**1.4**	kj-NN		**91.52**	8.45	**88.57**	90.77	**89.65**	**1.4**	kj-NN		**76.74**	**3.22**	**70.70**	88.23	**78.50**	**1.4**
A1	NYT-2	93.29	**43.23**	93.61	91.23	92.40	2.6	A1	ARXIV-15	NA	NA	NA	NA	NA	NA	A1	ARXIV-54	75.54	3.63	70.19	87.95	78.07	3.6
A2		**94.51**	42.64	83.15	**99.32**	90.52	2.6	A2		NA	NA	NA	NA	NA	NA	A2		**76.65**	**3.85**	63.38	**96.94**	76.65	2.4
A3		91.23	18.62	90.22	89.89	90.05	4.8	A3		NA	NA	NA	NA	NA	NA	A3		76.33	3.40	66.09	86.68	75.00	4.6
A4		94.50	42.12	94.55	91.07	92.78	2.8	A4		NA	NA	NA	NA	NA	NA	A4		76.54	3.82	70.19	88.01	78.10	2.6
kj-NN		**94.51**	42.64	**94.63**	91.15	**92.86**	**1.6**	kj-NN		NA	NA	NA	NA	NA	NA	kj-NN		**76.65**	**3.85**	**70.38**	88.05	**78.23**	**1.2**
A1	NYT-5	96.61	**72.33**	92.12	93.55	92.83	2.6	A1	ARXIV-25	67.75	3.70	95.01	88.93	91.87	4.4	A1	ARXIV-53	78.55	4.91	70.58	88.35	78.47	3.6
A2		**97.04**	71.69	75.92	**99.18**	86.01	2.8	A2		70.64	5.10	91.22	**98.17**	**94.56**	2.2	A2		**79.16**	**5.10**	63.54	**96.75**	76.71	2.4
A3		92.33	41.43	86.12	91.97	88.95	4.6	A3		**81.23**	**5.93**	**97.30**	88.16	92.50	**2.0**	A3		76.48	3.22	65.89	86.98	74.98	4.8
A4		**97.04**	71.63	92.23	93.45	92.84	2.4	A4		70.48	5.09	96.10	88.98	92.40	3.0	A4		79.04	5.08	70.65	88.51	**78.58**	2.4
kj-NN		**97.04**	71.69	**92.43**	93.44	**92.93**	**1.8**	kj-NN		70.64	5.10	96.09	88.85	92.33	3.0	kj-NN		**79.16**	**5.10**	**70.77**	88.46	78.63	**1.4**
A1	NYT-50	91.20	27.29	94.20	90.36	92.24	3.2	A1	ARXIV-35	71.98	3.02	80.03	87.14	**83.43**	2.8	A1	ARXIV-52	77.44	**3.39**	70.86	88.53	78.71	2.4
A2		**93.33**	**27.76**	85.40	**99.34**	91.84	2.4	A2		73.58	**3.27**	72.75	**95.60**	82.62	2.4	A2		**78.10**	3.07	63.76	**96.73**	76.85	2.6
A3		90.78	13.34	92.68	89.11	90.86	4.8	A3		**74.14**	2.72	71.60	84.32	77.44	4.2	A3		69.53	1.56	65.85	86.51	74.78	4.4
A4		93.30	27.44	95.64	90.34	92.91	2.8	A4		73.54	**3.27**	79.92	86.99	83.30	2.8	A4		77.93	3.03	70.90	88.45	78.71	3.0
kj-NN		**93.33**	**27.76**	**95.78**	90.36	**92.99**	**1.2**	kj-NN		73.58	**3.27**	**80.04**	86.98	83.36	**2.0**	kj-NN		**78.10**	3.07	**71.08**	88.43	**78.81**	**2.0**
A1	NYT-20	89.01	5.94	93.10	90.29	91.67	4.0	A1	ARXIV-45	69.42	2.96	70.02	86.64	77.45	4.0	A1	ARXIV-51	**68.41**	**3.03**	70.76	88.33	78.58	2.6
A2		**91.42**	6.57	83.56	**98.90**	90.58	2.6	A2		69.94	**3.28**	63.93	**95.41**	76.56	2.6	A2		65.24	2.19	63.81	**96.89**	76.94	2.8
A3		89.93	**9.45**	91.19	87.44	89.27	3.8	A3		**71.56**	3.17	65.43	84.83	73.87	3.8	A3		57.35	1.12	65.83	86.86	74.90	4.2
A4		**91.42**	6.51	93.60	90.49	92.02	2.2	A4		69.88	3.27	70.09	86.88	77.59	2.6	A4		64.76	2.11	70.80	88.68	78.74	3.0
kj-NN		**91.42**	6.57	**93.80**	90.64	**92.19**	**1.4**	kj-NN		69.94	**3.28**	**70.29**	86.87	**77.70**	**1.6**	kj-NN		65.24	2.19	**70.91**	88.63	**78.79**	**2.0**

Table V
COMPARISON W.R.T. OUR COMPETITORS (TYPE O, ARXIV).

		AUC	AP	R1	R2	R5		AUC	AP	R1	R2	R5		AUC	AP	R1	R2	R5
LOF	ARXIV-15	63.44	1.78	0.00	5.13	7.69	ARXIV-45	62.99	2.43	2.02	2.02	5.05	ARXIV-53	57.37	1.33	2.42	3.23	9.68
RS-Hash		54.22	1.17	2.56	2.56	2.56		49.18	0.99	0.00	0.00	6.06		42.32	0.79	0.00	0.00	0.00
ANCS		47.46	0.96	0.00	0.00	5.13		51.50	1.32	1.01	1.01	2.02		47.55	0.93	0.81	1.61	3.23
k-ANCS		67.14	1.98	0.00	7.69	10.26		70.11	2.23	1.01	3.03	17.17		63.56	1.54	2.42	4.03	9.68
TONMF		57.65	**7.32**	0.00	7.69	15.38		57.49	**17.23**	0.00	0.00	2.02		56.12	4.26	0.00	1.61	8.06
VMF-Q		**71.71**	2.69	**5.13**	**10.26**	15.38		66.59	2.20	3.03	7.07	15.15		77.29	3.16	3.23	8.06	25.00
CVDD		69.85	2.27	0.00	5.13	**17.95**		**71.32**	2.10	2.02	4.04	11.11		69.28	1.98	2.42	4.03	12.90
kj-NN		54.21	1.17	0.00	0.00	0.00		69.94	3.28	**6.46**	**9.09**	**18.99**		**79.16**	**5.10**	**7.74**	**15.65**	**32.42**
LOF	ARXIV-25	68.41	2.17	4.35	6.52	13.04	ARXIV-55	59.47	1.47	3.23	3.23	11.29	ARXIV-52	55.58	1.09	0.81	0.81	4.84
RS-Hash		44.21	0.91	0.00	0.00	4.35		49.00	0.94	0.81	0.81	3.23		52.52	1.08	0.00	1.61	5.65
ANCS		47.88	1.09	2.17	4.35	6.52		51.69	1.80	1.61	3.23	6.45		45.02	0.84	0.00	0.00	0.81
k-ANCS		67.87	2.00	2.17	4.35	10.87		67.59	2.25	3.23	6.45	14.52		66.26	1.61	0.81	3.23	8.87
TONMF		60.03	**6.37**	0.00	4.35	13.04		56.86	0.92	1.61	3.23	9.68		54.04	2.70	1.61	3.23	8.06
VMF-Q		71.71	2.69	5.13	10.26	15.38		72.99	2.65	4.84	8.06	16.94		67.57	2.06	2.42	7.26	13.71
CVDD		**74.55**	2.19	2.17	4.35	8.70		60.92	1.63	1.61	4.84	11.29		55.64	1.39	1.61	4.84	11.29
kj-NN		70.64	5.10	**10.44**	**16.52**	**27.82**		**76.74**	**3.22**	**5.49**	**9.84**	**20.48**		**78.10**	**3.07**	**3.87**	**7.90**	**19.35**
LOF	ARXIV-35	71.00	2.16	0.00	3.23	11.29	ARXIV-54	61.92	1.33	0.00	2.42	6.45	ARXIV-51	**82.57**	**4.50**	**7.26**	**14.52**	**25.81**
RS-Hash		47.77	0.96	1.61	1.61	4.84		47.95	0.92	0.00	0.81	2.42		51.22	1.10	1.61	4.03	6.45
ANCS		52.94	1.07	0.00	0.00	3.23		50.46	1.01	0.81	0.81	3.23		65.91	2.36	5.65	10.48	15.32
k-ANCS		73.43	2.36	1.61	4.84	16.13		68.93	1.83	0.00	1.61	8.87		54.19	1.68	4.03	7.26	11.29
TONMF		56.90	1.16	1.61	3.22	6.45		59.46	**5.94**	0.00	0.00	6.45		55.23	1.01	1.61	1.61	7.26
VMF-Q		60.97	1.28	0.00	0.00	1.61		61.81	1.33	0.81	1.61	6.45		62.54	1.54	1.61	4.84	8.87
CVDD		53.89	1.06	0.00	0.00	1.61		60.30	1.62	1.61	5.65	10.48		66.93	2.54	4.84	8.87	23.39
kj-NN		**73.58**	**3.27**	**6.45**	**10.97**	**20.97**		**76.65**	3.85	**5.81**	**10.00**	**21.61**		65.24	2.19	4.68	7.90	14.36

Table VI
COMPARISON W.R.T. OUR COMPETITORS (TYPE M, ARXIV).

		P	R	F1	R10	R20		P	R	F1	R10	R20		P	R	F1	R10	R20		P	R	F1	R10	R20
W-CNN	ARXIV-25	95.33	82.51	88.46	47.77	82.15	ARXIV-45	51.47	81.26	63.02	24.80	49.40	ARXIV-54	58.07	87.94	69.95	28.03	57.40	ARXIV-52	49.10	84.35	62.07	24.84	49.12
VD-CNN		72.97	59.71	65.68	15.52	31.60		73.68	72.24	72.95	10.48	21.00		**87.22**	85.12	**86.16**	11.38	22.71		**86.92**	**91.59**	**89.19**	11.49	22.77
AT-RNN		65.90	72.24	68.92	32.97	65.94		**74.20**	49.97	59.72	37.70	49.60		69.81	56.33	62.35	34.79	55.77		78.94	61.65	69.23	39.41	60.87
RCNN		96.75	26.01	41.00	25.90	25.90		65.49	26.10	37.32	25.90	25.90		58.10	41.09	48.14	28.98	40.68		45.24	27.02	33.83	22.53	26.67
kj-NN		**96.09**	**88.85**	**92.33**	**49.22**	**88.85**		70.29	**86.87**	**77.70**	**45.78**	**76.89**		70.38	**88.05**	78.23	**46.83**	**77.78**		71.08	88.43	78.81	**47.28**	**78.79**
W-CNN	ARXIV-35	54.97	84.40	66.58	27.45	54.49	ARXIV-55	53.62	87.71	66.55	27.79	53.66	ARXIV-53	47.12	85.91	60.86	24.56	47.97	ARXIV-51	56.00	**86.47**	67.98	28.14	56.49
VD-CNN		66.06	80.08	72.40	11.73	23.01		**92.65**	**91.97**	**92.31**	12.01	24.08		**82.78**	82.64	**82.71**	10.95	21.62		78.89	82.56	**80.68**	10.24	20.42
AT-RNN		**84.37**	38.59	52.96	38.12	38.12		86.50	54.56	66.91	43.07	54.06		58.65	60.31	59.47	29.22	58.48		76.27	59.04	66.56	38.57	58.36
RCNN		80.21	25.02	38.14	24.72	24.72		83.80	33.87	48.24	33.56	33.56		63.25	22.58	33.28	22.33	22.33		**79.67**	25.09	38.16	24.80	24.80
kj-NN		80.04	**86.98**	**83.36**	**48.25**	**83.40**		70.70	88.23	78.50	**46.77**	**78.67**		70.77	**88.46**	78.63	**46.69**	**78.24**		70.91	88.63	78.79	**46.25**	**78.18**

[12] A. Lazarevic and V. Kumar, "Feature Bagging for Outlier Detection," in *KDD*, 2005, pp. 157–166.

[13] E. Müller, I. Assent, P. I. Sánchez, Y. Mülle, and K. Böhm, "Outlier Ranking via Subspace Analysis in Multiple Views of the Data," in *ICDM*, 2012, pp. 529–538.

[14] F. Keller, E. Müller, and K. Böhm, "HiCS: High Contrast Subspaces for Density-Based Outlier Ranking," in *ICDE*, 2012, pp. 1037–1048.

[15] H. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Outlier Detection in Arbitrarily Oriented Subspaces," in *ICDM*, 2012, pp. 379–388.

[16] H. Zhuang, C. Wang, F. Tao, L. M. Kaplan, and J. Han, "Identifying Semantically Deviating Outlier Documents," in *EMNLP*, 2017, pp. 2748–2757.

[17] R. Kannan, H. Woo, C. C. Aggarwal, and H. Park, "Outlier Detection for Text Data," in *SDM*, 2017, pp. 489–497.

[18] L. Ruff, Y. Zemlyanskiy, R. A. Vandermeulen, T. Schnake, and M. Kloft, "Self-Attentive, Multi-Context One-Class Classification for Unsupervised Anomaly Detection on Text," in *ACL (1)*, 2019, pp. 4061–4071.

[19] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *EMNLP*, 2014, pp. 1746–1751.

[20] A. Conneau, H. Schwenk, L. Barrault, and Y. LeCun, "Very Deep Convolutional Networks for Text Classification," in *EACL (1)*, 2017, pp. 1107–1116.

[21] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, "Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification," in *ACL (2)*, 2016, pp. 207–212.

[22] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent Convolutional Neural Networks for Text Classification," in *AAAI*, 2015, pp. 2267–2273.

[23] C. C. Aggarwal, *Outlier Analysis*. Springer, 2013.

[24] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated Phrase Mining from Massive Text Corpora," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 10, pp. 1825–1837, 2018.

[25] F. Tao, H. Zhuang, C. W. Yu, Q. Wang, T. Cassidy, L. M. Kaplan, C. R. Voss, and J. Han, "Multi-Dimensional, Phrase-Based Summarization in Text Cubes," *IEEE Data Eng. Bull.*, vol. 39, no. 3, pp. 74–84, 2016.

[26] C. Zhang and J. Han, *Multidimensional Mining of Massive Text Data*, ser. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers, 2019.

[27] E. Fix and J. L. Hodges, "Discriminatory Analysis – Nonparametric Discrimination: Consistency Properties," *International Statistical Review / Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[28] S. A. Dudani, "The Distance-Weighted k-Nearest-Neighbor Rule," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 6, no. 4, pp. 325–327, 1976.

[29] M. Bicego and M. Loog, "Weighted K-Nearest Neighbor revisited," in *ICPR*, 2016, pp. 1642–1647.

[30] J. Gou, H. Ma, W. Ou, S. Zeng, Y. Rao, and H. Yang, "A generalized mean distance-based k-nearest neighbor classifier," *Expert Syst. Appl.*, vol. 115, pp. 356–372, 2019.

[31] K. Hechenbichler and K. Schliep, "Weighted k-nearest-Neighbor Techniques and Ordinal Classification," 2004, discussion Paper 399, SFB 386, Ludwigs Maximilians University Munich.

[32] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study," *Data Min. Knowl. Discov.*, vol. 30, no. 4, pp. 891–927, 2016.

[33] E. Schubert, A. Koos, T. Emrich, A. Züfle, K. A. Schmid, and A. Zimek, "A Framework for Clustering Uncertain Data," *PVLDB*, vol. 8, no. 12, pp. 1976–1979, 2015.

[34] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy, "Hierarchical Attention Networks for Document Classification," in *NAACL HLT*, 2016, pp. 1480–1489.

[35] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "Towards Open-Text Semantic Parsing via Multi-Task Learning of Structured Embeddings," *CoRR*, vol. abs/1107.3663, 2011.