SoftwareX 19 (2022) 101122

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

TextCL: A Python package for NLP preprocessing tasks

Alina Petukhova *, Nuno Fachada

Lusófona University, COPELABS, Campo Grande, 376, Lisbon, Portugal

ARTICLE INFO

Article history:
Received 5 June 2021
Received in revised form 9 May 2022
Accepted 1 June 2022

Keywords:
Natural language processing
Text filtering
Outlier detection

ABSTRACT

Preprocessing text data sets for use in Natural Language Processing tasks is usually a time-consuming and expensive effort. Text data, normally obtained from sources such as, but not limited to, web scraping, scanned documents or PDF files, is typically unstructured and prone to artifacts and other types of noise. The goal of the TextCL package is to simplify this process by providing multiple methods suited for text data preprocessing. It includes functionality for splitting texts into sentences, filtering sentences by language, perplexity filtering, and removing duplicate sentences. Another functionality offered by the TextCL package is the outlier detection module, which allows to identify and filter out texts that are different from the main topic distribution of the data set. This method allows selecting one of several unsupervised outlier detection algorithms, such as TONMF (block coordinate descent framework), RPCA (robust principal component analysis), or SVD (singular value decomposition) and apply it to the text data.

Code metadata

Current code version	v1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00105
Code Ocean compute capsule	Not applicable
Legal Code License	MIT
Code versioning system used	Git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Python≥3.5 numpy≥1.16.5, <1.20.0 pandas≥1.0.3 torch≥1.5.0, ≤1.7.1 pytorch_pretrained_bert≥0.6.2 langdetect≥1.0.8 lxml≥4.6.2 protobuf≥3.14.0 nltk≥3.4.5 scikit-learn≥0.24.1 testresources≥2.0.1
If available Link to developer documentation/manual	https://alinapetukhova.github.io/textcl/docs/
Support email for questions	alina.petukhova@ulusofona.pt

1. Motivation and significance

There is an immense amount of unstructured text generated every day from a variety of different sources. As information continuously grows, it is a critical task to preprocess data for a given domain in order to perform analysis, identify correlations, and build natural language processing (NLP) models. NLP is an area of research in Artificial Intelligence focused on processing and using text and speech data, and may include predictive, category classification, or text generation models. The quality of these models strongly depends on the quality of input data.

* Corresponding author.
E-mail address: alina.petukhova@ulusofona.pt (Alina Petukhova).

https://doi.org/10.1016/j.softx.2022.101122

There are a few common flaws that can be seen in raw text depending on the way it was obtained. If data comes from an optical character recognition (OCR) platform, tables and columns are typically not processed correctly and may add noise to the models. Additionally, some parts of large text scopes may contain sentences from languages other than the target language of the model. In real-world data, derived from a number of sources, texts may contain duplicate content due to the use of templates or reuse of the same content blocks. For text generation tasks, content duplication can cause model overfitting, and this must be considered in the preprocessing pipeline.

The most commonly used library for text preprocessing is `spaCy` [1]. It is an open-source library for text preprocessing. The most common functions used in the library are tokenization, lemmatization, part-of-speech tagging, entity recognition, punctuations, and stopword removal. Another commonly used library for text preprocessing is NLTK (Natural Language Toolkit) [2]. In general, spaCy is faster and more efficient than NLTK [3]. However, `spaCy` does not include pre-created models for some applications, such as sentiment analysis, which is typically easier to perform with NLTK. TextCl extends the functionality of `spaCy` by including functions for filtering sentences by language, perplexity filtering, removal of duplicate sentences, and outlier detection.

Another important problem to consider during text preprocessing is data distribution. In unstructured text data sets, there are often cases where documents do not belong to the main topic of the text. For example, a scientific article in a political news data set, or a fax cover sheet mixed with medical records. These are examples of outliers in the data distribution, and it is very useful to be able to detect them and filter them out of the main text scope in an unsupervised fashion.

A number of Python packages for outlier detection are available, for example `PyOD` [4], `PyNomaly` [5], and `alibi-detect` [6]. `PyOD` package contains 40 outlier detection algorithms, with the support of recent algorithms like ECOD [7]. `PyNomaly` is based on the LoOP (Local Outlier Probabilities) method [8], with scores normalized to the range 0–1. Both `PyOD` and `PyNomaly` do not support text data as an input. `Alibi-detect` focuses on outlier, adversarial, and drift detection, supporting 25 different algorithms. Although, it could be used for unstructured data such as images or text, there is no support for outlier detection for text data. TextCL was developed to fill this niche, offering several algorithms for detecting outliers specifically in text data, namely TONMF, RPCA and SVD.

More generally, the goal of TextCL is to address text preprocessing problems, offering Python implementations of several existing algorithms for language filtering, similarity filtering, perplexity filtering, and outlier detection in one library.

2. Software description

2.1. Software architecture

TextCL is written in Python 3 and provides algorithms and functions to accomplish common preprocessing and outlier detection steps for text data sets. It follows a procedural programming style, and the provided functions are organized as shown in Fig. 1. Functions are defined at a high level, enabling users to take advantage of several methods for text preparation and investigate the best parameters for a given text data set. The code is developed on top of well-established Python libraries, such as `numpy` [9], `sklearn` [10], and `pandas` [11], while integrating modern libraries specialized in deep learning and text operations like `nltk` [2], `torch` [12], `pytorch_pretrained_bert` [13], and `langdetect` [14]. It consists of two modules, **preprocessing** and **outliers_detection**, with the first operating as a sentence-level transformer and the latter as a document-level transformer. The first module includes **language_filtering**, **jaccard_sim_filtering**, and **perplexity_filtering**. Before using these functions, initial text should be split into sentences with the **split_into_sentences** function. The second module is represented by the **outliers_detection** function, which has a selection of algorithms to use for detecting outliers, as detailed in Fig. 1. Each function is designed in a consistent way, offering parameters to selected the desired column of the data frame and specify filtering thresholds. The output is standardized and compatible with `pandas` data frame format.

2.2. Software functionalities

The **preprocessing** module contains the sentence level transformers. The following list describes each of these functions in additional detail:

`language_filtering()` This function is used to filter sentences by language. Inputs to this function should be a `pandas` data frame with a sentence column, a threshold value, and a target language. The threshold value (or language score) is used for filtering and has a default value of 0.99. All sentences below the threshold will be filtered out.

`jaccard_sim_filtering()` This function is used to filter sentences by Jaccard similarity. It represents each sentence as an array of tokens and finds the intersection between two arrays. Similarity score is calculated based on the sentence intersections and if it is below the given threshold, the sentence will be filtered out.

`perplexity_filtering()` This function is used to filter sentences by perplexity. The first step creates contextual tokens to capture latent syntactic-semantic information and then uses GPT as a language model to calculate probabilities of the token's order in the sentence. Perplexity calculated as exp(*loss*) (where loss is the language modeling loss of a particular token) and compared with the given threshold. More details can be found in Ref. [15].

The second module, **outliers_detection**, includes functions to process a full document as a part of the text's scope. They operate as document-level transformers and have the following specifications:

`outlier_detection()` This function is used to detect outliers in a list of sentences based on contextual information using one of the several unsupervised methods provided. Outliers can be categorized as sentences which have significantly different meanings compared to the other sentences in the list. Input parameters include the `pandas` data frame with texts, the method to use for outlier detection (`''tonmf''`, `''rpca''`, or `''svd''`), and a norm to normalize the obtained matrix and detect irregular texts (default is l^2, i.e. the Euclidean norm). The first step is to generate a textual representation in the vector space by creating text embeddings as a bag of words, and then the selected algorithm is used to detect outliers in the list. The package supports several methods for outlier detection:

`tonmf()` Uses the TONMF algorithm to determine the outlier matrix. The solution is based on the non-negative matrix factorization with the extension of the block coordinate descent framework [16].

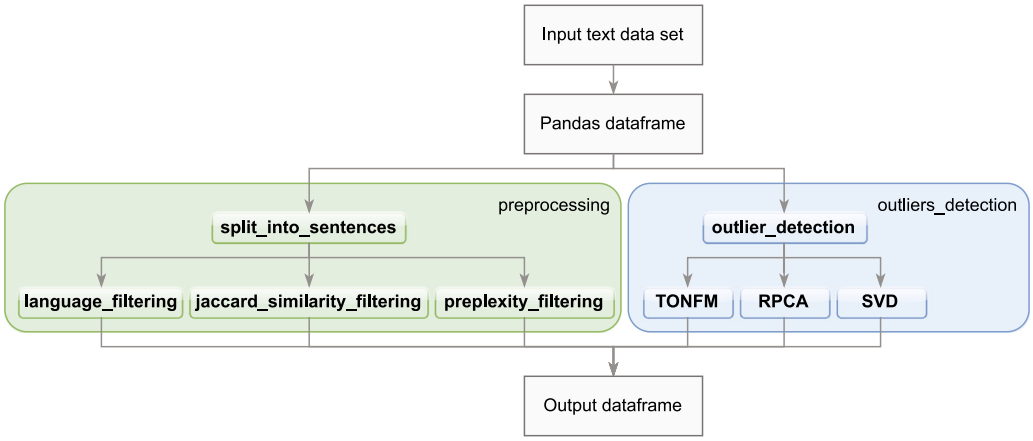


Fig. 1. Architecture of the TextCL package.

`rpca_implementation()` Uses the Robust Principal Component Analysis (RPCA) algorithm to determine the outlier matrix. RPCA uses low-rank approximation and yields two matrices: a low-rank matrix L and a sparse matrix S. After normalization, the S matrix represents the outlier score for the document [17,18].

`svd()` Uses single value decomposition (SVD) to determine the outlier matrix, presented as a multiplication of the square root of the diagonal elements of the rectangular diagonal matrix S and the complex unitary matrix V.

3. Illustrative examples

These example demonstrate how to preprocess text, cleaning it up for the modeling stage. Usually, text preprocessing is performed manually and is very time-consuming. The TextCL package helps to identify and filter out (1) sentences in languages other than the target language, (2) linguistically unconnected and/or corrupted sentences, and (3), duplicate sentences.

Another feature of the package is the ability to identify and filter out outliers from the text scope. As outliers, we consider texts that do not contextually belong to the main topic of the text. It is important to be able to identify these anomalies without having labeled data, or, in other words, to be able to use a general approach for finding outliers in unstructured text.

This example will use a subset of the BBC data set [19] with additional manually generated sentences. This data set is provided in the package's repository and will be subsequently referred to as the modified BBC data set. Nonetheless, TextCL can be used with any text data set loaded as a `pandas` data frame.

As a first step, load TextCl and auxiliary data processing packages (`numpy` and `pandas`):

```
import textcl
import pandas as pd
import numpy as np
```

Prepare the input data from the modified BBC data set bundled with TextCL. The example will demonstrate package capabilities on the subset of the BBC News data set. This subset contains 5 topics (business, entertainment, politics, sport, tech) with manually inserted texts to demonstrate the capabilities of the package. Load the modified BBC data set into memory:

```
input_df =
    pd.read_csv('prepared_bbc_dataset.csv').reset_index()
```

The initial subset used in this example is presented in Table 1. To be able to process/filter sentences from the text separately, it is necessary to split them as rows in a `pandas` data frame. The `split_into_sentences()` function is used for this purpose. The data loaded in the previous section has a column named "text". By default, the `split_into_sentences()` function expects to find texts in this column. However, an alternative name for this column can be specified in the function's `text_col` parameter. Thus, splitting the data set texts into sentences is done as follows:

```
df = textcl.split_into_sentences(input_df)
```

After splitting the text data set into sentences the `df` data frame will have the structure presented in Table 2.

3.1. Preprocessing on the sentence level

3.1.1. Filtering on language

The input to this function should be a `pandas` data frame (with "sentence" column), a threshold value, and a target language. The function, which depends on the `langdetect` [14] package, returns the probabilities of the sentences belonging to the specified language. All sentences below this threshold will be filtered out. The next code statement shows how to filter sentences written in English with a threshold of 0.99. If applied to the modified BBC data set, it will remove the manually inserted Turkish and Russian sentences from the data set.

```
df = textcl.language_filtering(input_df, threshold=0.99,
     language='en')
```

3.1.2. Filtering on Jaccard similarity

The following code shows how to apply the `jaccard_sim_filtering()` function to filter sentences by Jaccard similarity with threshold of 0.8. If the similarity score is above the specified threshold, a sentence will be filtered out.

```
df = textcl.jaccard_sim_filtering(input_df, threshold=0.8)
```

3.1.3. Filtering on perplexity score

The function `perplexity_filtering()` is used to filter sentences by perplexity, i.e., when sentences are linguistically incorrect and/or unconnected with the remaining text.

Table 1
Initial subset of BBC data set.

Index	Topic	Text
0	Business	WorldCom bosses' $54 m payout Ten former direc...
1	Business	Profits slide at India's Dr Reddy Profits at ...
2	Business	Liberian economy starts to grow The Liberian ...
3	Business	UluslararasıPara Fonu (IMF), Liberya ekonomis...
4	Entertainment	Singer Ian Brown 'in gig arrest' Former Stone...
5	Entertainment	Blue beat U2 to top France honour Irish band ...
6	Entertainment	Housewives lift Channel 4 ratings The debut o...
7	Entertainment	Домохозяйки подняли рейтинги канала ...
8	Entertainment	Housewives Channel 4 reytinglerini yükseltti A...
9	Politics	Observers to monitor UK election Ministers wi...
10	Politics	Lib Dems highlight problem debt People vulner...
11	Politics	Minister defends hunting ban law The law bann...
12	Sport	Legendary Dutch boss Michels dies Legendary D...
13	Sport	Connors boost for British tennis Former world...
14	Sport	Sociedad set to rescue Mladenovic Rangers are...
15	Tech	Mobile games come of age The BBC News webs...
16	Tech	PlayStation 3 processor unveiled The Cell pro...
17	Tech	PC photo printers challenge printed pictures c...
18	Tech	PC photo printers challenge pros Home printed...
19	Tech	processor come pros 43 t6 43 Table data 342 5 ...
20	Tech	Janice Dean currently serves as senior meteoro...

Table 2
Split by sentences BBC data set. For brevity, only a few sentences of the text at index 0 are shown.

Index	Topic	Text	Sentence
0	Business	WorldCom bosses' $54 m payout Ten former direc...	WorldCom bosses' $54 m payout Ten former direc...
0	Business	WorldCom bosses' $54 m payout Ten former direc...	James Wareham, a lawyer representing one of t...
0	Business	WorldCom bosses' $54 m payout Ten former direc...	The remaining $36 m will be paid by the directo...
0	Business	WorldCom bosses' $54 m payout Ten former direc...	But, a spokesman for the prosecutor, New York ...
0	Business	WorldCom bosses' $54 m payout Ten former direc...	Corporate governance experts said that if the...

In general, perplexity is a measurement of how well a probability distribution or probability model predicts a sample. In the case of the text data, we will be checking the probability of the next word to be in the given sentence. A low perplexity indicates the probability distribution is good at predicting the word.

The first step creates contextual tokens to capture latent syntactic-semantic information provided by the `pytorch_pretrained_bert` package [20] with pretrained `openai-gpt` tokenizer and using GPT as a language model with `OpenAIGPTLMHeadModel`. Perplexity is calculated as exp(*loss*), where *loss* is the language modeling loss of a particular token. The next block of code shows how to filter sentences from a data frame by perplexity using a specific threshold value.

```
df = textcl.perplexity_filtering(input_df, threshold=1000)
```

After applying all the preprocessing functions to the initial subset, as shown in the previous code snippets, the number of sentences reduced from 319 to 246. Turkish and Russian texts, partial duplicates, and linguistically incorrect sentences were removed. The final output is presented in Table 3. TextCL's tutorial, available at https://git.io/JsFYT, offers additional details and insights on the presented workflow.

Table 3
The modified BBC data set after preprocessing with TextCL.

Index	Text
0	WorldCom bosses' $54 m payout Ten former direc...
1	Profits slide at India's Dr Reddy Profits at ...
2	Liberian economy starts to grow The Liberian ...
4	Singer Ian Brown 'in gig arrest' Former Stone...
5	Blue beat U2 to top France honour Irish band ...
6	Housewives lift Channel 4 ratings The debut o...
9	Observers to monitor UK election Ministers wi...
10	Lib Dems highlight problem debt People vulner...
11	Minister defends hunting ban law The law bann...
12	Legendary Dutch boss Michels dies Legendary D...
13	Connors boost for British tennis Former world...
14	Sociedad set to rescue Mladenovic Rangers are...
15	Mobile games come of age The BBC News website...
16	PlayStation 3 processor unveiled The Cell pro...
18	PC photo printers challenge pros Home printed...
20	Janice Dean currently serves as senior meteoro...

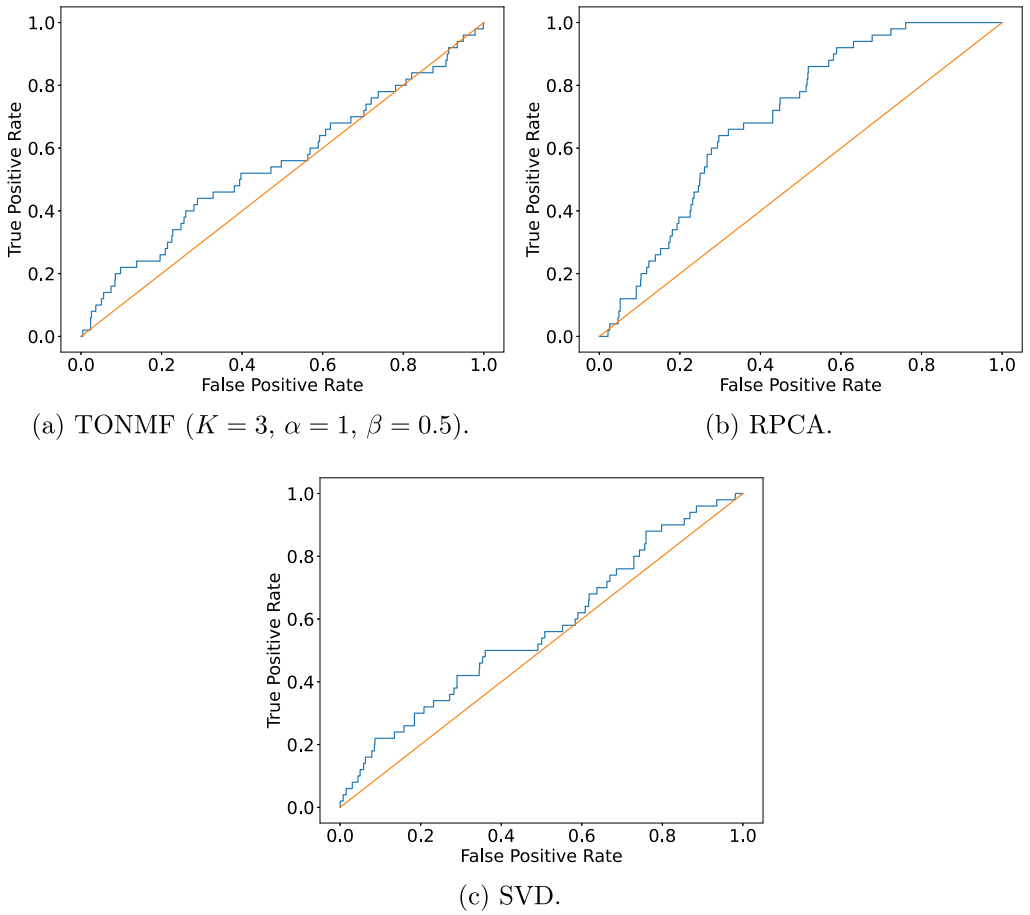
3.2. Outlier filtering

Outlier detection is an important task in text data mining. It can help us find unusual patterns that may be interesting and useful. Text data is uniquely challenging to outlier detection because of its sparsity and highly-dimensional nature.

There are many different techniques for anomaly detection, but they generally fall into one of two categories: supervised or unsupervised. Supervised methods require a training data set of normal and abnormal examples in order to learn a model which can then be used to classify new data points. Unsupervised methods do not require such a training data set, and instead learning patterns from the structure of the data in order to identify outliers.

In this package, we used methods based on Non-Negative Matrix Factorization (NMF) [21,22] to detect the text topics in an unsupervised fashion, so that we are able to detect outliers without labeling of topics.

The way it works is that NMF decomposes high-dimensional vectors into a lower-dimensional representation. These lower-dimensional vectors are non-negative, which also means their coefficients are non-negative. With this approach, we can use the fact that NMF is similar to probabilistic latent semantic indexing (pLSI) [23] and latent Dirichlet allocation (LDA) [24] generative models. From the original matrix, A, NMF yields two matrices, W and H. The former represents the topics detected in the text data set, while the latter contains the weights for those topics. In other words, A is the documents-by-words matrix, W is



(a) TONMF ($K = 3$, $\alpha = 1$, $\beta = 0.5$). (b) RPCA.

(c) SVD.

Fig. 2. ROC-curves for outlier detection in the BBC data set using the three outlier detection methods implemented in TextCL.

the topics-by-words matrix, and H is the documents-by-topics matrix.

The `outlier_detection()` function is used to detect outliers in a list of sentences based on contextual information using unsupervised methods. Text embeddings are created using a bag-of-words model as an input for the implemented algorithms. The main input parameters for this function are the `pandas` data frame containing the texts, the method to use for outlier detection (TONMF by default), and the type of norm (l^2 by default) to normalize the obtained matrix and detect unrelated texts.

The `outlier_detection()` function was tested for the "tech" category, that includes a manually inserted text outlier (*index* = 20). This outlier contains a person's profile instead of "tech" text. The next line of code selects rows which have "tech" in the *topic* column.

```
df = df[df.topic == 'tech']
```

With the "tech" news selected, we can run the `outlier_detection()` function with the RPCA algorithm:

```
df, _ = textcl.outlier_detection(df, method='rpca',
    Z_threshold=1.0)
```

The text with *index* = 20 was removed since it described a person's profile instead of tech news. The outliers can also be visualized with the use of the different algorithms. Code samples can be found in the package's tutorial.

The ROC curves produced by the different outlier detection techniques are shown in Fig. 2.

4. Impact

The main impact of the created software package on the community is its combination of the different filtering methods into one library. TextCL significantly simplifies text preprocessing for a variety of classification, prediction, text generation, and other NLP tasks in both academic and professional contexts.

The package provides the first Python implementation of the TONMF algorithm and allows the choice of different normalization functions for outlier detection. All algorithms in the package are designed to have simple calls with flexible parameters, allowing users with minimal Python experience to filter data frames and, by utilizing just one function, generate complex low-rank approximation matrices and detect outliers. Input parameters allow users to fine-tune the algorithms for the best fit of the processed data set. For example, for the specific data set used in the article, experimenting with the different outlier detection algorithms resulted in areas under curve from 0.531 (TONMF) to 0.618 (RPCA), as shown in Fig. 2.

The TextCL package was initially designed with a focus on the field of the news data sets [19,25–27]. However, TextCL is not limited to this specific area and can be used with a variety of textual data sets to improve input quality for later NLP modeling. The quality of preprocessing can be improved by finding the best parameter combination for the target data set area.

5. Conclusions

In this paper we presented TextCL, an open-source and versatile Python library for detecting and filtering various anomalies in

text data. It helps users to reduce time spent in data preprocessing and focus on the implementation of more complex models. The functionality of the package as well as the source code is fully documented.

The implemented outlier detection algorithms are used as benchmarks in the field, but show limited generalization capabilities. We believe that a future direction for this library is the development and/or implementation of new algorithms for this purpose.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by Fundação para a Ciência e a Tecnologia, Portugal under grant UIDB/04111/2020 (COPELABS).

References

[1] Honnibal M, Montani I, Van Landeghem S, Boyd A. SpaCy 2: Industrial-strength natural language processing in Python. 2020, http://dx.doi.org/10.5281/zenodo.1212303.

[2] Loper E, Bird S. NLTK: The natural language toolkit. 2002, http://dx.doi.org/10.48550/ARXIV.CS/0205028, arXiv, URL https://arxiv.org/abs/cs/0205028.

[3] Omran FNAA, Treude C. Choosing an NLP library for analyzing software documentation: A systematic literature review and a series of experiments. In: Proceedings of the 14th international conference on mining software repositories. MSR, vol. 17, IEEE Press; 2017, p. 187–97. http://dx.doi.org/10.1109/MSR.2017.42.

[4] Zhao Y, Nasrullah Z, Li Z. PyOD: A python toolbox for scalable outlier detection. J Mach Learn Res 2019;20(96):1–7, URL http://jmlr.org/papers/v20/19-011.html.

[5] Constantinou V. PyNomaly: Anomaly detection using local outlier probabilities (loop). J Open Source Softw 2018;3(30):845. http://dx.doi.org/10.21105/joss.00845.

[6] Van Looveren A, Klaise J, Vacanti G, Cobb O, Scillitoe A, Samoilescu R. Alibi detect: Algorithms for outlier, adversarial and drift detection. 2019, URL https://github.com/SeldonIO/alibi-detect.

[7] Li Z, Zhao Y, Hu X, Botta N, Ionescu C, Chen GH. ECOD: Uunsupervised outlier detection using empirical cumulative distribution functions. 2022, CoRR, abs/2201.00382 arXiv:2201.00382.

[8] Kriegel H-P, Kröger P, Schubert E, Zimek A. LoOP: Local outlier probabilities. In: International conference on information and knowledge management, proceedings. CIKM, vol. 09, New York, NY, USA: Association for Computing Machinery; 2009, p. 1649–52. http://dx.doi.org/10.1145/1645953.1646195.

[9] Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with numpy. Nature 2020;585(7825):357–62. http://dx.doi.org/10.1038/s41586-020-2649-2.

[10] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.

[11] pandas development team T. pandas-dev/pandas: Pandas. 2020, http://dx.doi.org/10.5281/zenodo.3509134.

[12] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems 32. Curran Associates, Inc.; 2019, p. 8024–35, URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[13] Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, et al. Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations. Online: Association for Computational Linguistics; 2020, p. 38–45, URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[14] Shuyo N. Language detection library for java. 2010, URL http://code.google.com/p/language-detection/.

[15] Radford A, Narasimhan K, Salimans T, Sutskever I. Improving language understanding by generative pre-training. Tech. rep., OpenAI; 2018, URL https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf.

[16] Kannan R, Woo H, Aggarwal CC, Park H. Outlier detection for text data: An extended version. 2017, arXiv preprint arXiv:1701.01325.

[17] Ganguli D. Robust-PCA. 2020, URL https://github.com/dganguli/robust-pca.

[18] Candès EJ, Li X, Ma Y, Wright J. Robust principal component analysis? J ACM 2011;58(3):1–37. http://dx.doi.org/10.1145/1970392.1970395.

[19] Greene D, Cunningham P. Practical solutions to the problem of diagonal dominance in kernel document clustering. In: Proceedings of the 23rd international conference on machine learning. ICML, vol. 06, New York, NY, USA: ACM; 2006, p. 377–84. http://dx.doi.org/10.1145/1143844.1143892.

[20] Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, et al. Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations. Online: Association for Computational Linguistics; 2020, p. 38–45, URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[21] Lee D, Seung H. Learning the parts of objects by non-negative matrix factorization. Nature 1999;401:788–91. http://dx.doi.org/10.1038/44565.

[22] Lee D, Seung H. Algorithms for non-negative matrix factorization. Adv Neural Inform Process Syst 2001;13.

[23] Hofmann T. Probabilistic latent semantic indexing. In: Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval, vol. 22. 1999, p. 50–7. http://dx.doi.org/10.1145/312624.312649.

[24] Blei DM, Ng AY, Jordan MI. Latent dirichlet allocation. J Mach Learn Res 2003;3:993–1022. http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993, URL http://portal.acm.org/citation.cfm?id=944937.

[25] L33 - Yahoo news ranked multi-label corpus, version1.0. 2017, URL https://webscope.sandbox.yahoo.com/catalog.php?datatype=l&did=84&guccounter=1.

[26] Gulli A. AG's corpus of news articles. 2005, URL http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html.

[27] Misra R. News category dataset. 2018, http://dx.doi.org/10.13140/RG.2.2.20331.18729.