Detecting Machine-obfuscated Plagiarism

Tomáš Foltýnek[1,4], Terry Ruas[1,2], Philipp Scharpf[3], Norman Meuschke[1,3], Moritz Schubotz[1], William Grosky[2], and Bela Gipp[1]

[1] University of Wuppertal, Rainer-Gruenter-Str. 21, D-42119, Wuppertal, Germany
last@uni-wuppertal.de
[2] University of Michigan-Dearborn, 4901 Evergreen Rd, 48128, Dearborn, USA
{truas,wgrosky}@umich.com
[3] University of Konstanz, Universittsstrae 10, 78464 Konstanz, Germany
first.last@uni-konstanz.de
[4] Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czechia
first.last@mendelu.cz

Abstract. Research on academic integrity has identified online paraphrasing tools as a severe threat to the effectiveness of plagiarism detection systems. To enable the automated identification of machine-paraphrased text, we make three contributions. First, we evaluate the effectiveness of six prominent word embedding models in combination with five classifiers for distinguishing human-written from machine-paraphrased text. The best performing classification approach achieves an accuracy of 99.0% for documents and 83.4% for paragraphs. Second, we show that the best approach outperforms human experts and established plagiarism detection systems for these classification tasks. Third, we provide a Web application that uses the best performing classification approach to indicate whether a text underwent machine-paraphrasing. The data and code of our study are openly available.

Keywords: paraphrase detection · plagiarism detection · document classification · word embeddings

1 Introduction

Plagiarism is a severe form of academic misconduct and a pressing problem for educational and research institutions, publishers, and funding agencies. Students who submit plagiarized works can receive credits without achieving their educational objectives. Researchers who plagiarize can inflate their publication and citation counts, secure research funding for the ideas of others, and advance to job positions for which they are not qualified [42, 13].

To counteract academic plagiarism, many institutions employ *plagiarism detection systems (PDS)*. These tools reliably identify duplicated text yet are significantly less effective in detecting paraphrases, translations, and other concealed forms of plagiarism [19, 43, 24].

Recent studies [39, 37] show that an alarming proportion of students nowadays employ *online paraphrasing tools (OPT)* (also known as text rewriting or

text spinning tools) to obfuscate text taken from other sources. According to Rogerson & McCarthy [39]: *"[...] spinning tools are equally available to academics who may be enticed with the notion of re-purposing already published content as a way of increasing research output."*.

OPT typically employ artificial intelligence approaches to paraphrase a text, e.g., by replacing words with their synonyms [45]. The tools were initially designed for Search Engine Optimization [22] by inflating a website's PageRank. The idea is to re-use the content of the promoted website to create numerous bogus websites that link to the advertised website. OPT serve to alter the content so that Web search engines do not recognize the fraudulent websites as duplicates and disregard them for calculating the PageRank of the promoted site. If successful, the approach negatively affects the users of Web search engines since the inflated PageRanks do not reflect the impact of the websites but rather the effort invested into producing fraudulent websites.

In academia, OPT help to obfuscate plagiarism, facilitate collusion, and support ghostwriters in producing work that appears original. The tools severely threaten the effectiveness of plagiarism detection systems, which are crucial for ensuring academic integrity. Rogerson & McCarthy [39] call for technical solutions to identify machine-paraphrased text and their integration with educational and policy actions to counteract the use of OPT. The International Journal for Educational Integrity even devoted a special issue[5] to this topic.

This paper answers the call of the academic integrity community by devising an automated approach that reliably distinguishes human-written from machine-paraphrased text and providing the solution as a free and open-source Web application. We structure the presentation of our contributions as follows. Section 2 briefly reviews related work on paraphrase identification and the application of dense vector models for natural language processing (NLP). Section 3 describes the training and selection of dense vector models and machine learning classifiers for our task. Section 4 presents the evaluation of the automated classification approach using the judgments of experts and the capabilities of the leading plagiarism detection system Turnitin as baselines. Section 5 summarizes our contributions and presents future work.

2 Related Work

The research on plagiarism detection has yielded many approaches that employ lexical [4, 18], syntactical [33, 44], semantic [41, 27], or cross-lingual text analysis [12, 14]. These approaches reliably detect copied or moderately altered plagiarism. Some approaches can also identify paraphrased and translated text.

Most research on paraphrase identification focuses on quantifying to which degree the meanings of two sentences are identical. Approaches for this task employ lexical and syntactic analysis, semantic similarity measures that are either knowledge-based (i.e., derived from dictionaries, thesauri, or other lexical

[5] `https://edintegrity.biomedcentral.com/mbp`

resources) or corpus-based (e.g., LSA [9], ESA [15], or word embeddings), as well as shallow and deep machine learning [2].

Approaches analyzing nontextual content features, such as academic citations [16, 28], images [25, 11], and mathematical content [26, 29], complement the text analysis approaches to improve the detection of concealed plagiarism.

The problem we address, i.e., distinguishing human-written and machine-paraphrased text at the level of documents and passages is still in its early stages. The work of Zhang et al. [45] is most related to our contributions. The authors provided a tool that determines if two articles are derived from each other and clusters these related articles. However, Zhang et al. did not investigate the task of distinguishing original and machine-fabricated text. Dey et al. [10] applied an SVM classifier to identify semantically similar tweets and other short texts.

Regarding techniques to accomplish the task at hand, the use of dense vectors to represent words in documents has attracted much research in recent years. Word embedding techniques, such as word2vec [31], have alleviated common problems in bag-of-words (BOW) approaches, e.g., scalability issues, and the curse of dimensionality. In addition to word embeddings, representing entire documents [21] and characters [7, 36] in a single fixed-length dense vector is another successful approach. The two techniques can capture latent semantic meaning from textual data using efficient neural network language models. The superiority of token-based embedding models over count-based models has been observed for several NLP problems, such as word similarity, document classification, and sentiment analysis [35]. However, the use of neural language models comes at the cost of requiring large amounts of data to derive the models. Moreover, the embedding process does not observe word order, and its quality strongly depends on the selection of hyperparameters [30, 38, 17, 3].

3 Methodology

To devise an approach for classifying texts as either human-written or machine-paraphrased, we analyzed the performance of pre-trained word embedding models that convert texts into fixed-length vectors. We investigated both classifying entire documents and paragraphs. Classifying paragraphs represents the more realistic detection task since plagiarists more often copy and obfuscate passages rather than whole texts [39, 43]. After training the models, we used features thereof in machine learning classifiers, as we describe hereafter.

3.1 Datasets for Training and Testing

To create training sets, we used all 4,012 *featured articles* from the English Wikipedia because they objectively cover a wide range of topics in great breadth and depth[6] . Senior Wikipedia editors select articles of superior quality as featured articles (approx. 0.1% of all articles). Featured articles typically have nu-

[6] `https://en.wikipedia.org/wiki/Wikipedia:Content\_assessment`

merous authors and undergo many revisions. Thus, they are written in high-quality English and unlikely to exhibit a bias towards the writing style of specific persons. Lastly, the articles are publicly available, which increases the reproducibility of our research.

To obtain a *training set for documents*, we machine-paraphrased (*spun*) all articles using the SpinBot[7] API. The service is the technical backbone of several widely-used OPT, e.g., Paraphrasing Tool[8][39] and Free Article Spinner[9]. Thus, the training set comprises of 8,024 articles (4,012 original, 4,012 spun).

To create a *test set for documents*, we selected 1,990 Wikipedia articles labeled as *good articles* at random. To receive this label, articles must be well-written, verifiable, neutral, broad in coverage, stable, and illustrated by media[6]. We paraphrased all articles using the SpinBot API to obtain the test set of 3,980 articles (1,990 original, 1,990 spun).

To obtain the *training and test sets for paragraphs*, we split the original and spun articles from the document training set (8,024) and the document test set (3,980) into paragraphs. We discarded paragraphs with fewer than three sentences, as these typically represented titles or subtitles. The resulting training set consists of 200,767 paragraphs (98,282 original, 102,485 spun); the test set consists of 79,970 paragraphs (39,241 original, 40,729 spun).

3.2 Word Embedding Models

We evaluated the following pre-trained word embedding models for the classification task: GloVe[10] [34], word2vec[11] [31], fastText[12] [5], and USE[13] [7]. GloVe and fastText use a corpus of Wikipedia articles to derive their vector representations. Word2vec uses Google News articles; USE employs a mixed collection including Wikipedia articles, Web news, question-answer Web pages, discussion fora, and the Stanford Natural Language Inference (SNLI) corpus [7].

Additionally, we trained a paragraph-vector (PV) model [21] from scratch. This model uses a Wikipedia Dump [40] as the training corpus, a distributed bag-of-words training model (DBOW), a window size of 15 words, a minimum count of 5 words, trained word-vectors in skip-gram fashion, averaged word vectors, and 30 epochs. We chose the distributed bag-of-words training model for paragraph vectors (PV-DBOW) over a distributed memory model for paragraph vectors (PV-DM) because of its superiority for semantic similarity tasks [20]. Parameters we do not describe, correspond to the default values in the *gensim*[14] API. All the word embedding models have 300 dimensions, except for USE, which has 512 dimensions. Table 1 summarizes the word embedding models we analyzed.

[7] `https://spinbot.com/API`
[8] `https://paraphrasing-tool.com/`
[9] `https://free-article-spinner.com/`
[10] `https://nlp.stanford.edu/projects/glove/`
[11] `https://code.google.com/archive/p/word2vec/`
[12] `https://fasttext.cc/docs/en/english-vectors.html`
[13] `https://tfhub.dev/google/universal-sentence-encoder/2`
[14] `https://radimrehurek.com/gensim/models/doc2vec.html`

Table 1. Word embedding models in our experiments.

Algorithm	Main Characteristics	Training Corpus	Dimensions
GloVe	Word-word co-occurrence matrix	Wikipedia dump 2014 + Gigaword 5	300
word2vec	Continuous Bag-of-Words (CBOW)	Google News	300
Paragraph Vectors	Distributed Bag-of-Words (PV-DBOW)	Wikpedia Dump 2010	300
fastText-rw	Skip-gram without sub-words	Wikipedia Dump 2017 + UMBC	300
fastText-sw	Skip-gram with sub-words	Wikipedia Dump 2017 + UMBC	300
USE	Deep Average Network	Wikipedia + Various sources	512

Each text is represented as the average of its constituent word vectors according to the word embedding models in Table 1. We accessed the pre-trained model and retrieved the vectors for the words occurring in each of the texts. If none of the words in a document existed in the pre-trained model, the document would have been discarded. However, this case did not occur.

All the models in Table 1, except for PV-DBOW, yield a vector representation for each word. In PV-DBOW, the embedded tokens represent entire texts. Thus, a match of an unseen text, i.e., a text not part of the external training corpus, and the pre-trained PV-DBOW model is unlikely. Inferring the vector representations for unseen texts requires an additional training step. Both training steps, i.e., building the document embeddings model (similar to the model used in word2vec) and inferring the vector representations, require parameter tuning. For all texts in our training and test sets, we performed this extra training step using the following hyperparameters for the *gensim* API: alpha = 10^{-4}, min alpha = 10^{-6}, and 300 epochs. The resulting PV-DBOW document embedding model requires at least 7 GB of RAM to be loaded and used. All word-based embedding models require between 1 GB to 3 GB of RAM. The higher memory consumption of PV-DBOW can make it unsuitable for some use cases.

3.3 Machine Learning Classifiers

After applying the pre-trained models to our training and test sets, we passed on the results to five machine learning classifiers: k Nearest Neighbors (kNN) [1], Random Forests (RF) [6], Logistic Regression (LR) [23], Support Vector Machines (SVM) [8], and Naïve Bayes (NB) [32]. We used multiple classifiers to explore the stability of the word embedding models concerning each classifier's characteristics. We adjusted the parameters for each classifier using a grid-search approach for the parameter values shown in Table 2.

4 Evaluation

Section 4.1 presents the results of applying the combinations of word embedding models and machine learning classifiers to the test sets. Section 4.2 and Section 4.3 establish two baselines for the results of the automated classification approach by indicating how accurately human experts (4.2) and respectively, a leading PDS (4.3), identify machine-paraphrased articles.

Table 2. Grid-search configuration.

Classifier	Parameter	Range
kNN	neighbors	1, 5, 15, 25 ... 95
Logistic Regression	solver	newton-cg, lbfgs, sag, saga
	maximum iteration	500, 1000, 1500
	multi-class	ovr, multinomial
	tolerance	0.01, 0.001, 0.0001, 0.00001
Support Vector Machine	kernel	linear, radial bases function, polynomial
	gamma	0.01, 0.001, 0.0001, 0.0001
	polynomial degree	1, 2, 3, 4, 5, 6, 7, 8, 9
	C	1, 10, 100
Random Forest	number of estimators	100, 325, 550, 775, 1000
	maximum features	auto, sqrt
	maximum depth	10, 32, 77, 100, None
	minimum samples split	2, 5, 10
	minimum samples leaf	1, 2, 4

4.1 Automated Classification

Tables 3 and 4 show the accuracy of the classification approaches at the document level and the paragraph level, respectively. Due to resource limitations, we did not employ kNN and RF for the paragraph classification task but will investigate these classifiers in the future.

At the document level, PV-DBOW outperformed the other techniques for four of the five classifiers, followed by word2vec, and fastText-rw. However, at the paragraph level, PV-DBOW consistently yielded the worst results for all tested classifiers. This finding is in line with results by [20], who reported a performance drop when using PV-DBOW for short documents.

Table 3. Classification accuracy for documents.

Classifier	GloVe	word2vec	PV-DBOW	fastText-rw	fastText-sw	USE
kNN	0.8874	**0.9085**	0.8867	0.8920	0.7696	0.8525
RF	0.9085	0.9397	**0.9606**	0.9246	0.8791	0.8533
LR	0.9457	0.9563	**0.9829**	0.9191	0.6950	0.7734
SVM	0.9716	0.9744	**0.9900**	0.9789	0.9518	0.9437
NB	0.7427	0.7437	**0.8829**	0.7492	0.6920	0.7455

Boldface indicates the best value of a row.
Underlining indicates the best value of a column.

Table 4. Classification accuracy for paragraphs.

Classifier	GloVe	word2vec	PV-DBOW	fastText-rw	fastText-sw	USE
LR	0.7758	**0.8050**	0.5806	0.7757	0.6068	0.6615
SVM	0.7908	0.8225	0.5244	**0.8336**	0.7896	0.7815
NB	0.5390	0.5163	0.5094	0.5229	0.5297	**0.5519**

Boldface indicates the best value of a row.
Underlining indicates the best value of a column.

For paragraph classification, the fastText-rw embedding model, in combination with an SVM classifier, achieved the best result followed by word2vec in combination with SVM. For fastText, we evaluated two training models, one using complete words (-rw) and the other using sub-words (-sw). The sub-words model uses the sum of the character n-grams of its constituent vectors. For example, using n = 3, the word *java* is represented as $\{ja, jav, ava, va\}$ and the word *java* itself. Thus, the sub-model can embed words that are not in the training corpus. In theory, this approach can capture more semantic information from the corpus. However, as Tables 3 and 4 show, on average, the whole word model (-rw) outperformed the sub-word one (-sw).

We conclude from the experiments that OPT often introduce rare and out-of-context words that allows the spun text to be identified. Prentice et al. [37] also reported unusual words as a means to manually identify spun essays.

4.2 Human Baseline

To gauge how well humans can distinguish original from machine-paraphrased text, we conducted a quiz. We randomly selected ten featured Wikipedia articles with various topics. For each article, we extracted the first one or two paragraphs to obtain a text of approximately 100 words. We paraphrased six of the excerpts via the SpinBot API and used the other four extracts unaltered. Using QuizMaker[15], we prepared a Web-based quiz that showed the ten excerpts one at a time. Participants could vote (by clicking one of two buttons) whether the text had been machine-paraphrased and optionally submit a freely worded comment after completing the quiz. We shared the quiz via e-mail and a Facebook group with researchers from the academic integrity community.

During three weeks, 73 subjects completed the quiz. The completion times ranged between 2min 12s and 36min 51s with an average of 9min and 18s. The accuracy of the participants ranged between 40% and 100%, with an average of 78.4%. One subject, who classified all cases correctly, commented: "*I paid special attention to any oddness in the text. I never read student works so carefully.*"

The experiment showed that experienced educators who read carefully and expect to encounter machine-paraphrased text could achieve an accuracy be-

[15] `https://www.quiz-maker.com/`

tween 90% and 100%. However, even in this setting, the average accuracy was below 80%. We expect that the efficiency will be lower in a realistic scenario, in which readers do not pay special attention to spotting machine paraphrases.

4.3 Plagiarism Detection System Baseline

To quantify the benefit that our approach (word2vec + SVM) provides over-current PDS, we compared it to the leading PDS Turnitin. Using the PDS, we checked ten machine-paraphrased articles selected at random from the document test set. In all cases, Turnitin found the correct source. The reported text similarity ranged between 49% and 67%, with an average of 55.2%. In other words, if the entire document was spun, Turnitin reliably identified the text overlap.

In a second experiment, we tested Turnitin's detection effectiveness for documents that mix original and machine-paraphrased text. We created nine documents (each approx. 3,000 words long) that contained between 10% and 90% machine-paraphrased text with the remainder being random text generated by a free online generator[16]. In a document that contained only one machine-paraphrased paragraph (298 words), Turnitin failed to identify the spun text. For the other documents, Turnitin correctly marks parts of the spun text as plagiarized but in 2 cases, fails to identify Wikipedia as the source.

These results are in line with the findings of Rogerson & McCarthy [39], who used two OPT (one of them based on the SpinbBot API) to paraphrase a paragraph from a prior publication. When the unchanged paragraph was used as the input to Turnitin, the system found a 100% match with the source. However, for the two machine-paraphrased versions of the paragraph, Turnitin computed a similarity score of zero for the source.

We conclude from these experiments that if a plagiarist employs OPT to paraphrase a few paragraphs, the resulting similarity is often below Turnitin's threshold, thus causing the plagiarism to remain undetected.

5 Conclusion & Future Work

A combination of the word2vec embedding model and an SVM classifier achieved the best trade-off between accuracy, computation time, and memory consumption for classifying entire documents and paragraphs as original or machine-paraphrased (cf. Section 4.1). Consequently, we chose this approach for realizing the demonstration system available at

`https://spindetector.org`.

The presented approach outperformed human experts in distinguishing original and machine-paraphrased text (cf. Section 4.2). Compared to existing PDS, the method achieved a better detection performance for cases in which a few paragraphs have been machine-paraphrased (cf. Section 4.3). If plagiarists spin

[16] `http://www.randomtextgenerator.com/`

entire documents, PDS can typically identify the source. However, PDS often fail to identify cases in which individual paragraphs have been taken over from a source and been obfuscated using OPT.

The presented classification approach demonstrates the feasibility of devising effective and efficient technical measures to counteract the use of OPT for disguising academic plagiarism. Including the presented methods in plagiarism detection systems can mitigate the weaknesses of current systems and assist educators in more reliably identifying disguised instances of plagiarism.

We are aware that the selection of high-quality Wikipedia articles and the inclusion of a single OPT limits the ability to generalize our findings. Distinguishing well-written Wikipedia articles from their machine-paraphrased counterparts does not entirely reflect the task that educators face in their everyday work. Students for whom English is a second language often use rare or out-of-context words due to their insufficient command of English.

Our future work will address the limitations of the current study by including articles from more repositories (e.g. ArXiv[17] and Reuters[18]) and additional OPT (e.g. Seo Tools Centre[19], EZ Rewriter[20], or Spinner Chief[21]). Moreover, we plan to collect original texts produced by non-native speakers of English and a dataset of texts paraphrased via cyclic machine translation. Plagiarists often employ cyclic machine translation to obfuscate duplicated text. These additions will increase the diversity of the texts used for training and testing and hence, the complexity of the classification task. To increase the classification effectiveness, we will investigate the performance of deep neural network approaches.

We are confident that the good results of the presented approach can be replicated and improved in future work. To ensure the reproducibility of our experiments and to facilitate future research on this task, the data and code for our study, as well as for the Web-based demonstration system, are available at

`https://doi.org/10.7302/bewj-qx93`

References

1. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine Learning **6**(1), 37–66 (1991). https://doi.org/10.1007/BF00153759
2. Altheneyan, A., Menai, M.E.B.: Evaluation of state-of-the-art paraphrase identification and its application to automatic plagiarism detection. International Journal of Pattern Recognition and Artificial Intelligence (2019). https://doi.org/10.1142/S0218001420530043
3. Altszyler, E., Sigman, M., Fernandez Slezak, D.: Corpus specificity in LSA and word2vec: The role of out-of-domain documents. In: Proceedings 3rd Workshop on Representation Learning for NLP. pp. 1–10 (2018). https://doi.org/10.18653/v1/W18-3001

[17] `https://arxiv.org/`
[18] `http://qwone.com/~jason/20Newsgroups/`
[19] `https://seotoolscentre.com/article-rewriter-tool`
[20] `http://www.ezrewrite.com/`
[21] `http://www.spinnerchief.com/`

4. Alvi, F., Stevenson, M., Clough, P.: Plagiarism Detection in Texts Obfuscated with Homoglyphs. In: Proceedings 39th European Conf. on IR Research (ECIR). vol. 10193 LNCS, pp. 669–675 (2017). https://doi.org/10/c6cd
5. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017). https://doi.org/10/gfw9cs
6. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
7. Cer, D., Yang, Y., Kong, S.y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., Kurzweil, R.: Universal sentence encoder for English. In: Proceedings Conf. on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 169–174 (2018). https://doi.org/10.18653/v1/D18-2029
8. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (1995). https://doi.org/10.1023/A:1022627411411
9. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science **41**(6), 391–407 (1990). https://doi.org/10/db4ft5
10. Dey, K., Shrivastava, R., Kaushik, S.: A Paraphrase and Semantic Similarity Detection System for User Generated Short-Text Content on Microblogs. Proceedings Int. Conf.on Computational Linguistics (Coling) **42**, 2880–2890 (2016)
11. Eisa, T., Salim, N., Alzahrani, S.: Figure Plagiarism Detection Using Content-Based Features. In: Proceedings Int. Conf. on Intelligent Computing, Communication and Devices (ICCD-2016). vol. 555 AISC, pp. 17–20 (2017). https://doi.org/10/dgcc
12. Ferrero, J., Agnes, F., Besacier, L., Schwab, D.: Using Word Embedding for Cross-Language Plagiarism Detection. In: Proceedings Conf. of the European Chapter of the Association for Computational Linguistics (EACL). vol. 2, pp. 415–421 (2017)
13. Foltýnek, T., Meuschke, N., Gipp, B.: Academic Plagiarism Detection: A Systematic Literature Review. ACM Computing Surveys **52**(6), 112:1–112:42 (2019). https://doi.org/10.1145/3345317
14. Franco-Salvador, M., Gupta, P., Rosso, P., Banchs, R.E.: Cross-Language Plagiarism Detection Over Continuous-Space- and Knowledge Graph-Based Representations of Language. Knowledge-Based Systems **111**, 87–99 (2016). https://doi.org/10.1016/j.knosys.2016.08.004
15. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In: Proceedings Int. Joint Conf. on Artificial Intelligence (IJCAI). pp. 1606–1611 (2007)
16. Gipp, B., Meuschke, N., Breitinger, C., Pitman, J., Nürnberger, A.: Web-based Demonstration of Semantic Similarity Detection Using Citation Pattern Visualization for a Cross Language Plagiarism Case. In: Proceedings Int. Conf. on Enterprise Information Systems (ICEIS). vol. 2, pp. 677–683 (2014). https://doi.org/10.5220/0004985406770683
17. Goldberg, Y., Hirst, G.: Neural Network Methods in Natural Language Processing. Morgan & Claypool Publishers (2017). https://doi.org/10.2200/S00762ED1V01Y201703HLT037
18. Kanjirangat, V., Gupta, D.: Investigating the Impact of Combined Similarity Metrics and POS Tagging in Extrinsic Text Plagiarism Detection System. In: Proceedings Int.l Conf. on Advances in Computing, Communications and Informatics (ICACCI). pp. 1578–1584 (2015). https://doi.org/10.1109/ICACCI.2015.7275838

19. Kanjirangat, V., Gupta, D.: Study on Extrinsic Text Plagiarism Detection Techniques and Tools. Journal of Engineering Science and Technology Review **9**(5), 9–23 (2016). https://doi.org/10.1109/ICACCI.2015.7275838
20. Lau, J.H., Baldwin, T.: An empirical evaluation of doc2vec with practical insights into document embedding generation. In: Proceedings Workshop on Representation Learning for NLP (2016). https://doi.org/10.18653/v1/w16-1609
21. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings 31st Int. Conf. on Machine Learning. vol. 32, pp. 1188–1196 (2014)
22. Madera, Q., García-Valdez, M., Mancilla, A.: Ad text optimization using interactive evolutionary computation techniques. In: Recent Advances on Hybrid Approaches for Designing Intelligent Systems. pp. 671–680 (2014). https://doi.org/10/dgcd
23. McCullagh, P., Nelder, J.: Generalized Linear Models, Second Edition. Chapman & Hall (1989)
24. Meuschke, N., Gipp, B.: State of the Art in Detecting Academic Plagiarism. International Journal for Educational Integrity **9**(1), 50–71 (2013). https://doi.org/10.5281/zenodo.3482941
25. Meuschke, N., Gondek, C., Seebacher, D., Breitinger, C., Keim, D., Gipp, B.: An Adaptive Image-Based Plagiarism Detection Approach. In: Proceedings 18th ACM/IEEE Joint Conference on Digital Libraries (JCDL). pp. 131–140 (2018). https://doi.org/10.1145/3197026.3197042
26. Meuschke, N., Schubotz, M., Hamborg, F., Skopal, T., Gipp, B.: Analyzing Mathematical Content to Detect Academic Plagiarism. In: Proceedings ACM Conf. on Information and Knowledge Management (CIKM). pp. 2211–2214 (2017). https://doi.org/10.1145/3132847.3133144
27. Meuschke, N., Siebeck, N., Schubotz, M., Gipp, B.: Analyzing Semantic Concept Patterns to Detect Academic Plagiarism. In: Proceedings Int. Workshop on Mining Scientific Publications (WOSP) at the 17th ACM/IEEE Joint Conference on Digital Libraries (JCDL). pp. 46–53 (2017). https://doi.org/10.1145/3127526.3127535
28. Meuschke, N., Stange, V., Schubotz, M., Gipp, B.: HyPlag: A Hybrid Approach to Academic Plagiarism Detection. In: Proceedings 41st Int. ACM SIGIR Conf. on Research & Development in Information Retrieval. pp. 1321–1324 (2018). https://doi.org/10.1145/3209978.3210177
29. Meuschke, N., Stange, V., Schubotz, M., Kramer, M., Gipp, B.: Improving Academic Plagiarism Detection for Stem Documents by Analyzing Mathematical Content and Citations. In: Proceedings ACM/IEEE Joint Conf. on Digital Libraries (JCDL). pp. 120–129 (2019). https://doi.org/10.1109/JCDL.2019.00026
30. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings Workshop Track 1st Int. Conf. on Learning Representations (ICLR) (2013)
31. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings 27th Conf. on Neural Information Processing Systems (NIPS). pp. 3111–3119 (2013)
32. Mitchell, T.M.: Machine learning, International Edition. McGraw-Hill (1997)
33. Mohebbi, M., Talebpour, A.: Texts Semantic Similarity Detection Based Graph Approach. Int. Arab Journal of Information Technology **13**(2), 246–251 (2016)
34. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings Conf. on Empirical Methods in Natural Language Processing (EMNLP). vol. 14, pp. 1532–1543 (2014). https://doi.org/10/gfshwg
35. Perone, C.S., Silveira, R., Paula, T.S.: Evaluation of sentence embeddings in downstream and linguistic probing tasks. ArXiv **abs/1806.06259** (2018)

36. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings Conf. of the North American Chapter of the Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/n18-1202
37. Prentice, F.M., Kinden, C.E.: Paraphrasing Tools, Language Translation Tools and Plagiarism: An Exploratory Study. International Journal for Educational Integrity **14**(1) (2018). https://doi.org/10.1007/s40979-018-0036-7
38. Roberts, K.: Assessing the corpus size vs. similarity trade-off for word embeddings in clinical NLP. In: Proceedings Workshop on Clinical NLP. pp. 54–63 (2016)
39. Rogerson, A.M., McCarthy, G.: Using Internet based paraphrasing tools: Original work, patchwriting or facilitated plagiarism? International Journal for Educational Integrity **13**(1) (2017). https://doi.org/10.1007/s40979-016-0013-y
40. Shaoul, C., Westbury, C.: The Westbury Lab Wikipedia Corpus (2010), `http://www.psych.ualberta.ca/~westburylab/downloads/westburylab.wikicorp.download.html`
41. Velásquez, J.D., Covacevich, Y., Molina, F., Marrese-Taylor, E., Rodríguez, C., Bravo-Marquez, F.: DOCODE 3.0 (DOcument COpy DEtector): A System for Plagiarism Detection by Applying an Information Fusion Process from Multiple Documental Data Sources. Information Fusion **27**, 64–75 (2016). https://doi.org/10.1016/j.inffus.2015.05.006
42. Weber-Wulff, D.: False Feathers. Springer Berlin Heidelberg (2014). https://doi.org/10.1007/978-3-642-39961-9
43. Weber-Wulff, D.: Plagiarism detectors are a crutch, and a problem. Nature (2019). https://doi.org/10.1038/d41586-019-00893-5
44. Yokoi, T.: Sentence-Based Plagiarism Detection for Japanese Document Based on Common Nouns and Part-of-Speech Structure. In: Proceedings Int. Conf. on Intelligent Software Methodologies, Tools and Techniques (SoMet). vol. 513 CCIS, pp. 297–308 (2015). https://doi.org/10/ggdv2p
45. Zhang, Q., Wang, D.Y., Voelker, G.M.: Dspin: Detecting automatically spun content on the web. In: Proceedings Network and Distributed System Security (NDSS) Symposium. pp. 23–26 (2014). https://doi.org/10.14722/ndss.2014.23004

Citation for this Paper

Foltýnek, T. & Ruas, T. & Scharpf, P. & Meuschke, N. & Schubotz, M. & Grosky, W. & Gipp, B., "Detecting Machine-obfuscated Plagiarism," in Proceedings of the iConference, 2020.

BibTeX:

```
@inproceedings{Foltynek2020,
title = {Detecting {Machine}-obfuscated {Plagiarism}},
booktitle = {Proceedings of the {iConference}},
author = {Folt{\'y}nek, Tom{\'a}{\v s} and Ruas, Terry and Scharpf, Philipp and
Meuschke, Norman and Schubotz, Moritz and Grosky, William and Gipp, Bela},
year = {2020}
}
```

RIS:

```
TY  - CONF
TI  - Detecting Machine-obfuscated Plagiarism
AU  - Foltýnek, Tomáš
AU  - Ruas, Terry
AU  - Scharpf, Philipp
AU  - Meuschke, Norman
AU  - Schubotz, Moritz
AU  - Grosky, William
AU  - Gipp, Bela
C3  - Proceedings of the iConference
DA  - 2020
ER  -
```

Related Publications: www.gipp.com/pub