Detection of Generated Text Reviews by Leveraging Methods from Authorship Attribution: Predictive Performance vs. Resourcefulness

Manfred Moosleitner[1] Günther Specht[1] Eva Zangerle[1]

Abstract: Textual reviews are an integral part of online shopping, provided the reviews are authentic. To this end, pre-trained large language models have been shown to generate convincing text reviews at scale. Therefore, a critical task is the automatic detection of reviews not composed by humans. State-of-the-art approaches to detect generated texts use pre-trained large language models, which exhibit hefty hardware requirements to run and fine-tune. Previous work has shown that texts generated by the same language model show a coherent writing style. We propose to leverage this property to identify whether a text was indeed automatically generated. In this paper, we investigate the performance of features prominently used for authorship attribution, using classifiers with substantially lower computational resource requirements. We show that features and methods from authorship attribution can be successfully applied for the task of detecting generated text reviews, leveraging the consistent writing style exhibited by large language models like GPT-2. We argue that our approach achieves similar performance as state-of-the-art approaches while providing shorter training times and lower hardware requirements, necessary for, e.g., ad-hoc detection tasks.

Keywords: Text Classification, Stylometric Text Features, Generated Text Detection

1 Introduction

User-created reviews are often available on online e-commerce platforms. Such reviews allow users to express their satisfaction or disappointment with products or services in the form of numeric ratings or written text reviews. While user ratings provide a quantitative view of user experiences, text reviews allow for providing a more detailed report about the perceived quality of the product or service. Such reviews may inform other users in the process of comparing products, finding viable alternatives, or eventually, purchasing a product. However, this assumes that reviews are authentic and not fictitious and therefore, fake. Fake reviews can be a threat to businesses [La12, LSV16, LZ16], regardless of whether a counterfeit review is written by a human or generated with the help of an algorithm.

Recently, machine learning approaches like pre-trained large language models (LLM), such as BERT [De18] or GPT-2 [Ra19], are prevalent in many natural language processing and text generation tasks. To this end, GPT-2 was used to generate convincing text reviews [Ip20, Sa22], substantiating that we need to find ways to differentiate between human-written and algorithmically generated texts. Ott et al. [Ot11] investigated differentiating

[1] Universität Innsbruck, Department of Computer Science, Austria; firstname.lastname@uibk.ac.at

doi:10.18420/BTW2023-11

between genuine and fabricated human-written reviews based on psycho-linguistic, lexical and n-gram features. They used these features and trained a Naïve Bayes and a linear Support Vector Machine as classifiers. Notably, in their evaluation, they also show that automated classification achieves better results than human judges. Ippolito et al. [Ip20] show that a large BERT model, fine-tuned for classification, can classify texts as either machine-generated or written by a human with an accuracy of up to 0.88, but also note less diversity for words used, due to preferably choosing words with a higher likelihood from the prevailing word distribution. Shahid et al. [Sh17] argue that if texts are generated by the same algorithm, these texts share the same author. Along these lines, we propose formulating the task of fake review detection as an authorship attribution task. Therefore, we leverage stylometric text features prominently used in the field of authorship attribution [Zl18, St21, TMS19, MS22] to detect text reviews that are generated by an LLM. Our contributions can be summarized as follows: (1) We propose modeling the task of detecting generated text reviews as an authorship attribution task. (2) We investigate and compare the performance of different stylometric text features and state-of-the-art authorship attribution approaches. (3) We show that Support Vector Machines and Decision Trees achieve predictive performance (F_1, precision, recall) comparable to those of LLM-based classifiers while having much lower requirements in terms of training time and hardware requirements. Particularly given the recent trend towards greener IT and more energy-efficient computing [Mu08], we argue that this is a pivotal dimension that needs to be considered when evaluating and comparing potential approaches. (4) To ensure reproducibility, we publish the code of our experiments and analysis at `https://git.uibk.ac.at/c7031305/btw23_textreviewdetection`.

2 Related Work

Ott et al. [Ot11] used part-of-speech (POS) tags, psycholinguistic and statistical text features, and *n*-grams to differentiate if human written reviews are genuine or fictive. They used 400 genuine reviews from Tripadvisor and 400 fabricated reviews, created by crowd workers. The authors used classifiers based on Naive Bayes, and on Support Vector Machines (SVM). Shahid et al. [Sh17] aim to separate Wikipedia articles and texts generated by content-spinning tools, using said articles as seed documents, by using an SVM-based classifier. They employed an assortment of stylometric features like *n*-grams, vocabulary richness, readability, and others. Salminen et al. [Sa22] evaluated the predictive performance of a RoBERTa [Li19] based model, a GPT-2 based model[2], and an SVM based classifier. They created a balanced data set, consisting of approximately 40,000 text reviews, based on the Amazon Customer Review data set[3]. From the ten most occurring product categories in this data set, they sampled customer reviews to fine-tune an LLM to generate approximately 20,000 artificial text reviews. The authors additionally drew roughly 20,000 reviews from the Amazon data set as reviews written by humans.

[2] `https://github.com/openai/gpt-2-output-dataset/blob/master/detection.md`

[3] `https://s3.amazonaws.com/amazon-reviews-pds/readme.html`

Shahid et al. [Sh17] note that the seed documents and the generated texts differ in the way the texts are composed (i.e., their grammatical structure). Therefore, methods and features from the field of authorship attribution and plagiarism detection should be able to catch these differences. Zlatkova et al. [Zl18] proposed to use various frequencies on word and sentence level, lexical richness, readability metrics, and other features. They reached first place in the multi-author analysis shared challenge at the PAN[4] workshop in 2018. Strøm [St21] used a similar configuration for this challenge in 2021, reaching high performance in this authorship attribution task. Murauer et al. [MS22] proposed Dependency Tree-grams (DT-grams) for the task of authorship attribution. DT-grams capture the writing style of authors by using the grammatical structure of sentences. Substructures, extracted from the dependency trees, are used to represent the grammatical style of the text and author.

In this work, we put our focus on identifying whether a review was written by a human, or was generated by an LLM. Along the lines of Shahid et al. [Sh17], we argue that texts originating from the same language model share the same author, which in turn should exhibit a similar writing style across all generated texts. Therefore, contrary to previous works, we model the task of detecting whether a review was manually written or automatically generated as an authorship attribution problem. We particularly investigate the use of state-of-the-art authorship attribution models to the task and particularly investigate the feature sets by Zlatkova [Zl18]/Strøm [St21], and Murauer et al. [MS22].

3 Methodology

In the following, we first detail the different features employed and subsequently, describe the experimental setup used for the evaluation.

3.1 Features

The main goal of this work is to investigate generated review detection by leveraging features and models from the field of authorship attribution. Therefore, we rely on features that have been shown to capture the writing style of authors well for authorship attribution tasks, specifically in the sub-tasks of style change detection [Zl18, St21], intrinsic plagiarism detection [TMS19], and cross-language authorship attribution [MS22], leveraging the consistent writing style exhibited by texts generated by LLMs [Sh17, Ip20]. As baselines, we rely on word and character n-grams for comparison, along the lines of previous work [JL08, Ot11, Sh17]. We propose three types of features to represent reviews: (1) *textfeatures* (frequencies on word, phrase, and sentence level, and readability metrics); (2) *dtgrams* (substructures extracted from dependency trees); and (3) *ngrams* (word and character n-grams of varying lengths).

[4] https://pan.webis.de/shared-tasks.html

As *textfeatures*, we use the same collection of features as Zlatkova et al. [Zl18] and Strøm [St21] to represent the writing styles of the corresponding authors. Our implementation is based on the work by Strøm[5], in which the following five feature sets are extracted from the text reviews: (1) count metrics (for instance, number of sentences and words, count of English POS tags, capitalized words, and others); (2) number of occurrences of function words and function phrases; (3) the number of uses of digits (0, 1, ..., 9) and their alphabetical counterpart (zero, one,..., nine), use of UK English (e.g., colour) and US English words (e.g., color), and the use of contractions and non-contractions; (4) the frequency of punctuation marks and other special characters; and (5) nine readability metrics (e.g., Flesch reading ease [Fl48]). This provides us with a total number of 487 features. For the *dtgrams* features, we rely on dependency-tree-based features, aiming to find texts that are composed with a similar grammatical style, therefore, attributing it to the generating LLM as the single author. In the following, we briefly introduce DT-grams. At first, we create the dependency tree for a given sentence (cf. Figure 1 for an example tree of the sentence "The quick brown fox jumped over the lazy dog."). In the resulting tree, each node holds the English POS tag for the corresponding word. Next, specific substructures of the tree are grouped together, where the substructures can be of different shapes. For our work, we used the shape of *pq-grams* with $p = 2$ and $q = 3$, which means that we use the parent-child relation of two nodes ("jumped" and "dog" as one example from Figure 1) and three sibling nodes ("over", "the", and "lazy" as one example from Figure 1). In the next step, the DT-grams are constructed using a sliding window, similar to n-grams. Here the grouped substructures are traversed (from parent to child, and siblings from left to right) and their corresponding POS tags are concatenated using the underscore as delimiter and the asterisk for when nodes are absent in the sliding window. The sentence in the given example results in a total of 18 strings, with "VBD_NN_IN_DT_JJ" being an example with no absent node and "VBD_NN_*_*_IN" being an example with absent nodes.

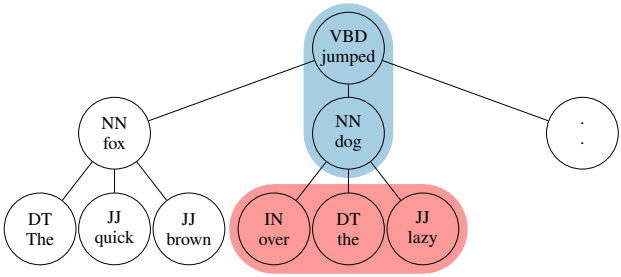

Fig. 1: Tree representation for the given sentence, based on the dependency tree. In each node, we display the word and its corresponding POS tags. Highlighted in blue is the parent-child relationship between *jumped* and *dog*, and marked in red is the sibling relationship between *over*, *the*, and *lazy*. Here, "VBD_NN_IN_DT_JJ" and "VBD_NN_*_*_IN" are two examples of the 18 dtgrams.

For the calculation of the dt-grams, we used two Python libraries from Murauer et al. [MS22]:

[5] https://github.com/eivistr/pan21-style-change-detection-stacking-ensemble

tuhlbox[6] parses the texts and calculates the dependency trees using the Python library Stanza [Qi20] from the Stanford NLP group, and *treegrams* [7], is used to extract the dt-grams from these dependency trees. For the *n-grams* feature set and as a baseline, we employ word and character n-gram TF-IDF vectors, using the scikit-learn library[8]. We utilized similar parameters for the vectorization as [Zl18, St21], further details are given in Section 3.2.

3.2 Experimental Setup

We evaluated the classification performance of the proposed approach in multiple experiments. Specifically, we employed stratified ten-fold cross-validation and used F_1, precision, and recall, as evaluation metrics. For the binary classification, the generated reviews were used as the positive class. Furthermore, we also measured classification runtimes and memory usage to compare the efficiency and required resources for each of the feature sets and classifiers. Memory usage was recorded using the Python library memory-profiler; the experiments were executed on a general-purpose processor.

3.2.1 Dataset

We relied on the data set provided by Salminen et al. [Sa22] for the evaluations. The data set is balanced in the amount of original and generated text reviews, featuring approximately 20,000 text reviews per class. We computed all features proposed and normalized them by removing the mean and scaling the values to unit variance (using scikit-learn's StandardScaler).

rating	class	text
1	CG	Editor was too busy watching "Duck Dynasty" and not paying attention to his work!!!
5	CG	I loved this book. The characters were believable and the plot was interesting. I really enjoyed this book
1	OR	Just a bit strange and different for me. Probably excellent for others.
5	OR	A truly riveting page turner. All three in the series were fantastic.

Tab. 1: Example reviews from the category Book_5 for the classes Computer Generated (CG) and Original Review (OR), one with the highest and lowest rating each.

3.2.2 Classification Algorithms and Evaluation Strategy

For the classification, we rely on five established classifiers: Support Vector Machine (SVM) with linear kernel, SVM with Radial Basis Function kernel, k-Nearest Neighbor (kNN),

[6] https://pypi.org/project/tuhlbox/
[7] https://pypi.org/project/treegrams/
[8] https://scikit-learn.org/

Gradient Boosting Decision Tree (gbdt), and Random Forest (rf). For the two SVM-based and the kNN-based classifiers, we used the corresponding scikit-learn implementations. For the tree-based classifiers, we utilized the gradient boosting framework LightGBM[9]. Following the example of Strøm[St21], we also used the hyperparameter optimization framework Optuna[10] to efficiently tune the hyperparameters for gbdt. The hyperparameters for the SVM-based classifiers were optimized using a grid-search approach.

This optimized hyperparameters[11] were then used for the final set of experiments, where all the necessary data and results were collected. Since we used the data set from Salminen et al. [Sa22], we reproduced their setup using a basic RoBERTa [Li19] model and added it as a state-of-the-art baseline to our experiments. As a more lightweight LLM, we added a DistilBERT [Sa19] model to the list of classifiers.

Furthermore, we tested the values for F_1, precision, and recall, per ten-fold cross-validation, for normal distribution by performing a Shapiro-Wilk [SW65] test. This showed $p > .05$ for all cases, therefore, we assume that the determined results all stem from a normal distribution. This allowed us to perform statistical significance tests using paired t-tests, where the pairs were built within feature set and also within classifier.

3.2.3 Preliminary Experiments

In preliminary experiments, we conducted a coarse grid search for the classifiers. Regarding the feature set ngrams, the texts were used as input without any further pre-processing. The following parameters were supplied with the stated values: maximum number of n-grams (5,000, 25,000, no limit), word and character n-grams, range of n-grams (uni- to six-grams). The classifier hyperparameters used for the grid search are shown in Table 2.

linear SVM	C: 0.1, 1.0, 10; dual: False; tol: 0.001
rbf SVM	C: 0.1, 1.0, 10; kernel: rbf; tol: 0.001
kNN	n_neighbors: 5, 40, 100
gbdt & rf	objective: binary learning_rate: 0.1, 0.01, 0.001, 0.0001, 0.00001 bagging_freq: 40; bagging_fraction: 0.85

Tab. 2: Hyperparameters used for the initial grid search.

These experiments showed that the linear SVM, the SVM with a radial basis function kernel, and the Gradient-boosted decision tree constantly outperformed kNN and Random Forest classifiers. Therefore, we excluded kNN and Random Forest classifiers from further experiments.

[9] https://github.com/microsoft/LightGBM

[10] https://optuna.org/

[11] Details can be found at https://git.uibk.ac.at/c7031305/btw23_textreviewdetection

Likewise, we also experimented with different ranges for n-grams, using word and/or character n-grams, and the maximum number of features. Again, we performed a grid-search approach to get the individual F_1-scores for word and character uni- to six-grams, varying values of 5,000, 25,000, 50,000, and unlimited number of maximum features. The results have shown that character four-, five-, and six-grams performed best with all three classifiers. Regarding the maximum number of features, the SVM with radial basis function kernel performed best with a limit of 25,000 features, while the linear SVM and the Gradient-boosted decision tree performed best without limiting the number of features.

Based on the findings of these preliminary experiments, we performed a final optimization of the hyperparameters for the classification algorithms. Both SVM variants were tuned again using a grid-search approach, to find the best-performing value for the regularization parameter. For the SVM with radial basis function kernel, a value of 10 showed the best performance across the three feature sets. Regarding the linear SVM, a value 0.1 performed best for the feature set dtgrams, and 0.001 for the feature sets textfeatures and ngrams.

4 Experimental Results

Based on the collected predictions and measurements from our experiments, we compared the performance of the combinations of the feature sets with classifiers, in terms of prediction, runtime, and memory consumption.

4.1 Predictive Performance

We present the average F_1, precision, and recall from the stratified ten-fold cross-validations per feature set for the three classification algorithms and the two LLMs in Table 3.

For the textfeatures feature set, the gradient-boosted decision tree achieved the highest performance with an F_1-score of 90.83%, followed by the SVM with radial basis function kernel and the linear SVM with a slightly lower performance (89.46% and 89.04%, respectively). The best performance for the feature set dtgrams was achieved by the SVM with radial basis function kernel with an F_1-score of 91.86%. Here, the F_1-score of the linear SVM achieved a 1.85% and the Gradient-boosted decision tree a 2.82% lower F_1-score. For the ngrams feature set, the SVM with radial basis function kernel obtained an F_1-score of 95.45%. This outperformed the F_1-scores of the other two classifiers, with the linear SVM achieving a 1.35% and the gradient-boosted decision tree a 2.16% lower F_1-score.

The results of the paired t-tests within the evaluated feature sets and classifiers showed $p < .05$ for all pairs. The paired t-test with the results of RoBERTa and DistilBERT also showed $p < .05$. We also compared the results of the best non-LLM model with the RoBERTa model in a paired t-test, which also showed significant differences ($p < .05$).

		linear SVM		SVM rbf		GBDT	
		μ	σ	μ	σ	μ	σ
F_1	textfeatures	0.8904	0.0037	0.8946	0.0044	0.9083	0.0027
	dtgrams	0.9028	0.0036	0.9186	0.0042	0.8904	0.0046
	ngrams	0.9410	0.0039	**0.9545**	0.0019	0.9329	0.0039
	RoBERTa	**0.9794**	0.0027				
	DistilBERT	0.9670	0.0035				
PRECISION	textfeatures	0.9011	0.0050	0.8910	0.0056	0.9078	0.0053
	dtgrams	0.8765	0.0044	0.8998	0.0054	0.8832	0.0060
	ngrams	**0.9639**	0.0034	0.9446	0.0020	0.9244	0.0050
	RoBERTa	**0.9946**	0.0015				
	DistilBERT	0.9882	0.0022				
RECALL	textfeatures	0.8799	0.0053	0.8982	0.0053	0.9089	0.0037
	dtgrams	0.9308	0.0066	0.9384	0.0052	0.8977	0.0091
	ngrams	0.9193	0.0078	**0.9646**	0.0039	0.9415	0.0054
	RoBERTa	**0.9647**	0.0055				
	DistilBERT	0.9466	0.0065				

Tab. 3: Mean and standard deviation for F_1, precision, and recall over the values from the ten-fold cross-validation. Results are reported per classifier per feature set. The best results for the non-LLM and LLM approaches are marked in boldface.

From these results, we conclude that RoBERTa, as expected, achieves the best F_1-scores regarding the predictive performance. The proposed text features, on the other hand, achieve only slightly lower F_1-scores, with the best classifier achieving a less than 2% lower F_1-score. However, we argue that the differences are subtle and, as we will show in the following experiments, the non-LLM approaches are able to outperform RoBERTa w.r.t. resource consumption and runtime.

4.2 Runtime Performance

Given the ever-increasing need for more energy-efficient computation, another important aspect is the runtime performance of the different approaches. We analyzed the time needed to fit the algorithms on average for the ten-fold cross-validation. All the experiments regarding runtime were conducted on the same virtual machine (Debian GNU/Linux 10 (buster) OS with 12 cores and 128GiB of memory). We utilize the grid-search functionality of scikit-learn. We present the average time to fit a classifier and the corresponding standard deviation in Table 4. The reported runtimes for ngrams, RoBERTa, and DistilBERT, include the time to convert the text reviews into their respective representations. The feature sets textfeatures and dtgrams were pre-computed, with a runtime of $\approx$ 380 seconds for the former and 73439.468 seconds for the latter.

The time to fit the models increases with the number of features, with the exception of the linear SVM when trained with ngrams. This constellation was about 2.4 times faster than

	linear SVM		SVM rbf		GBDT	
	μ	σ	μ	σ	μ	σ
textfeatures	357.69s	54.09s	1,259.42s	107.04s	**54.59s**	2.17s
dtgrams	**490.26s**	52.40s	4,020.95s	145.32s	2,135.30s	145.16s
ngrams	**146.29s**	26.00s	17,605.94s	439.07s	9,695.23s	137.98s
RoBERTa	28149.44s	243.10s				
DistilBERT	8293,96s	104.49s				

Tab. 4: Average (μ) and standard deviation (σ) of the measured runtimes in seconds per classifier per feature set over the ten-fold cross-validation. The best values per feature set are marked in boldface.

when paired with textfeatures, and about 3.3 times faster when using dtgrams. Most notable are the runtimes for the gradient-boosting decision tree with textfeatures, the SVM with radial basis function kernel and ngrams, and RoBERTa. The first two displayed the shortest and the longest mean time to fit, respectively, from the non-LLM models, and RoBERTa the overall longest average time to fit. These experiments showed that the proposed simple approaches clearly outperform RoBERTa.

4.3 Memory Usage

Besides runtime, a second important cornerstone when evaluating and comparing these methods is their memory usage and hence, resource consumption. Particularly with LLMs, the amount of memory required has increased substantially. Therefore, we are also interested in comparing memory usage among the proposed approaches. One run of the final experiments was used to measure the amount of memory that was allocated during the execution of the cross-validation (cf. 4). We present the results of these analyses in Table 5. As expected, the memory profiler reported memory usage numbers for the non-LLM classifiers that are only a fraction of the memory needed by the RoBERTa model. Particularly, the recorded memory requirements of RoBERTa are in the range of 245.62 to 778.79 times higher compared to the non-LLM models with the lowest memory consumption.

	SVM linear	SVM rbf	GBDT
textfeatures	42.516 MiB	17.027 MiB	19.520 MiB
dtgrams	17.301 MiB	16.961 MiB	14.152 MiB
ngrams	13.312 MiB	19.543 MiB	13.910 MiB
RoBERTa	10,364.988 MiB		
DistilBERT	4,880.535 MiB		

Tab. 5: Memory usage was acquired with cross-validation using the memory profiler.

4.4 Discussion

RoBERTa achieved the highest F_1-scores, followed by ngrams with rbfsvm, and ngrams with linsvm. When comparing these three, the highest F_1-score comes at the price of 1.6 and 192.2 times longer training times, for a difference of 1.8% and 3.15% in F_1-score, respectively. In terms of memory requirements, RoBERTa's memory usage is about 530 and 778 times higher when compared to rbfsvm with ngrams and linsvm with ngrams. In comparison, textfeatures with gbdt showed the shortest training time. Compared to RoBERTa, training times of ngrams with rbfsvm, ngrams with linsvm, and textfeatures with gbdt, were around 515, 322, and 3 times faster, and with a difference in F_1-scores of 6.42%, 4.62%, and 3.27%, respectively. The lowest memory requirement was recorded for ngrams with linsvm, which was 779 and 1.47 times lower than RoBERTa and ngrams with rbfsvc, with a difference in F_1-score of 3.15% and 1.35%.

Our experiments show it is possible to train classifiers quicker or with lower hardware requirements while sacrificing at most 6.42% of F_1-score, which is still higher than the performance of human raters [Ot11, Sa22]. Therefore, we argue that the proposed features borrowed from authorship attribution tasks are a valid option for the task of generated text detection.

5 Conclusion

We proposed using text features borrowed from the field of authorship attribution for the task of detecting generated product reviews. Related work suggests that generated texts differ in writing style, grammatical structure, and the diversity of words used from their input texts. Therefore, we used statistical textfeatures, features based on dependency-tree-grams, and *n*-grams to train a linear Support Vector Machine, a Support Vector Machine with radial basis function kernel, and a gradient-boosting decision tree as classifiers. We utilized a balanced dataset to evaluate their predictive performance using F_1-score, investigated their runtimes and memory requirements, also in comparison with a state-of-the-art RoBERTa LLM model. Our results show that classification algorithms like Support Vector Machines and Decision Trees can be trained using different stylometric features and achieve F_1-scores that come close to the performance of a basic RoBERTa model. While these non-LLM models show slightly lower performance, they can reach up to 515 faster training time and need up to 779 times less memory—two factors that become more and more central when choosing an approach.

In future work, we aim to investigate the importance and impact of individual features and extend the experiments to further datasets and text domains.

Bibliography

[De18] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805, 2018.

[Fl48] Flesch, Rudolph: A new readability yardstick. Journal of applied psychology, 32(3):221, 1948.

[Ip20] Ippolito, Daphne; Duckworth, Daniel; Callison-Burch, Chris; Eck, Douglas: Automatic Detection of Generated Text is Easiest when Humans are Fooled. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Online, pp. 1808–1822, July 2020.

[JL08] Jindal, Nitin; Liu, Bing: Opinion Spam and Analysis. In: Proceedings of the 2008 International Conference on Web Search and Data Mining. WSDM '08, Association for Computing Machinery, New York, NY, USA, p. 219–230, 2008.

[La12] Lappas, Theodoros: Fake Reviews: The Malicious Perspective. In (Bouma, Gosse; Ittoo, Ashwin; Métais, Elisabeth; Wortmann, Hans, eds): Natural Language Processing and Information Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 23–34, 2012.

[Li19] Liu, Yinhan; Ott, Myle; Goyal, Naman; Du, Jingfei; Joshi, Mandar; Chen, Danqi; Levy, Omer; Lewis, Mike; Zettlemoyer, Luke; Stoyanov, Veselin: RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692, 2019.

[LSV16] Lappas, Theodoros; Sabnis, Gaurav; Valkanas, Georgios: The Impact of Fake Reviews on Online Visibility: A Vulnerability Assessment of the Hotel Industry. Information Systems Research, 27(4):940–961, 2016.

[LZ16] Luca, Michael; Zervas, Georgios: Fake It Till You Make It: Reputation, Competition, and Yelp Review Fraud. Management Science, 62(12):3412–3427, 2016.

[MS22] Murauer, Benjamin; Specht, Günther: DT-grams: Structured Dependency Grammar Stylometry for Cross-Language Authorship Attribution. In: Proceedings of the 32nd GI-Workshop Grundlagen von Datenbanksysteme (GvDB'21). CEUR-WS.org, Aachen, 2022.

[Mu08] Murugesan, San: Harnessing green IT: Principles and practices. IT professional, 10(1):24–33, 2008.

[Ot11] Ott, Myle; Choi, Yejin; Cardie, Claire; Hancock, Jeffrey T.: Finding Deceptive Opinion Spam by Any Stretch of the Imagination. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. HLT '11, Association for Computational Linguistics, USA, p. 309–319, 2011.

[Qi20] Qi, Peng; Zhang, Yuhao; Zhang, Yuhui; Bolton, Jason; Manning, Christopher D.: Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 2020.

[Ra19] Radford, Alec; Wu, Jeffrey; Child, Rewon; Luan, David; Amodei, Dario; Sutskever, Ilya: Language Models are Unsupervised Multitask Learners. OpenAI blog, 1(8):9, 2019.

[Sa19] Sanh, Victor; Debut, Lysandre; Chaumond, Julien; Wolf, Thomas: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019.

[Sa22] Salminen, Joni; Kandpal, Chandrashekhar; Kamel, Ahmed Mohamed; gyo Jung, Soon; Jansen, Bernard J.: Creating and Detecting Fake Reviews of Online Products. Journal of Retailing and Consumer Services, 64:102771, 2022.

[Sh17] Shahid, Usman; Farooqi, Shehroze; Ahmad, Raza; Shafiq, Zubair; Srinivasan, Padmini; Zaffar, Fareed: Accurate Detection of Automatically Spun Content via Stylometric Analysis. In: 2017 IEEE International Conference on Data Mining (ICDM). IEEE, pp. 425–434, 2017.

[St21] Strøm, Eivind: Multi-label Style Change Detection by Solving a Binary Classification Problem—Notebook for PAN at CLEF 2021. In (Faggioli, Guglielmo; Ferro, Nicola; Joly, Alexis; Maistro, Maria; Piroi, Florina, eds): CLEF 2021 Labs and Workshops, Notebook Papers. CEUR-WS.org, Aachen, pp. 2146–2157, 9 2021.

[SW65] Shaphiro, S; Wilk, MBJB: An analysis of variance test for normality. Biometrika, 52(3):591–611, 1965.

[TMS19] Tschuggnall, Michael; Murauer, Benjamin; Specht, Günther: Reduce & Attribute: Two-Step Authorship Attribution for Large-Scale Problems. In: Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL). Association for Computational Linguistics, Hong Kong, China, pp. 951–960, November 2019.

[Zl18] Zlatkova, Dimitrina; Kopev, Daniel; Mitov, Kristiyan; Atanasov, Atanas; Hardalov, Momchil; Koychev, Ivan; Nakov, Preslav: An Ensemble-Rich Multi-Aspect Approach for Robust Style Change Detection. In (Cappellato, Linda; Ferro, Nicola; Nie, Jian-Yun; Soulier, Laure, eds): Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum. CEUR-WS.org, Aachen, 9 2018.