Detecting Automatically Generated Sentences with Grammatical Structure Similarity

Nguyen Minh Tien, Cyril Labbé

Univ. Grenoble Alpes, CNRS, Grenoble INP [**], LIG, F-38000 Grenoble, France
Minh-tien.nguyen@univ-grenoble-alpes.fr
Cyril.labbe@univ-grenoble-alpes.fr

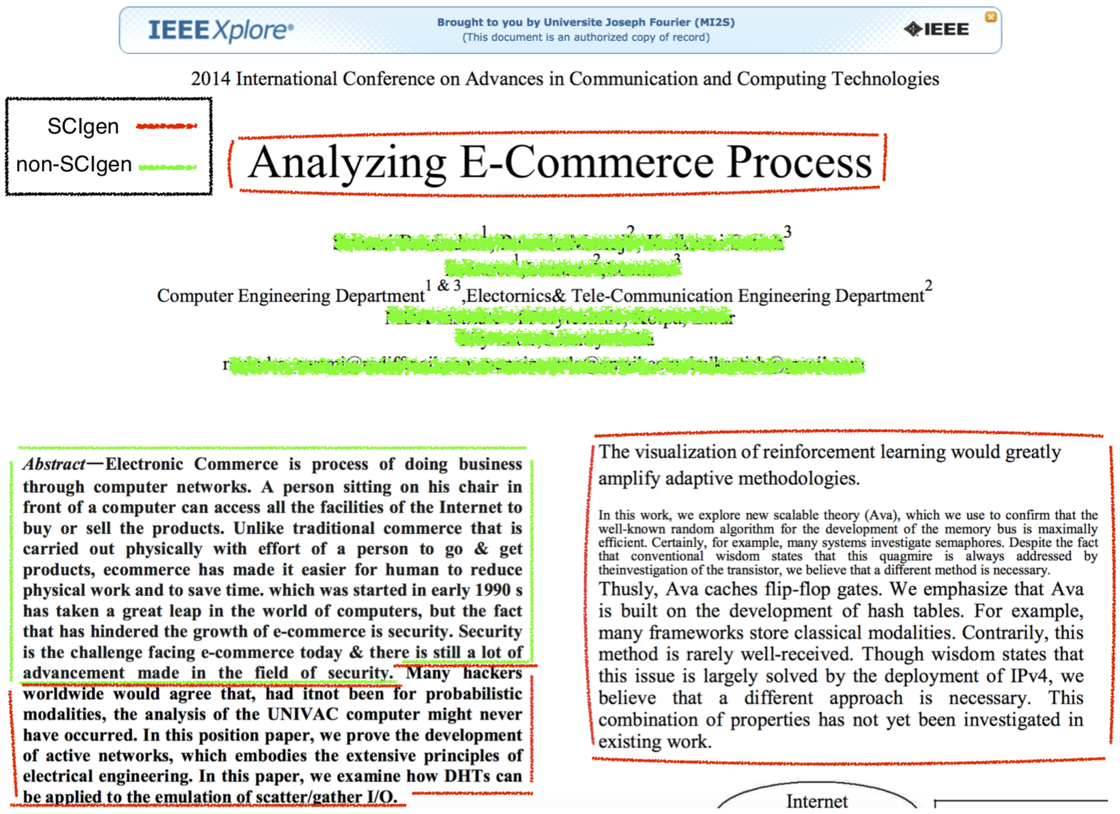
Abstract. Detection of automatically generated papers has been a new field of research. However, all current approaches are working at the document level and are unable to detect a small amount of generated text inside a large body of genuine written text. This paper will present the Grammatical Structure Similarity (GSS) measurement to detect sentences or short fragments from known generators. The proposed approach is tested against common machine learning methods, the ability to detect a modified generator is also tested.

1 Introduction - Problems

Detection of automatically generated fake papers has become an important matter [9] as well as a case study [8,12,14,6]. For example, automatically generated papers have been used to manipulate index score when [11] became one of the highest cited author on Google Scholar or when [2] published nonsensical papers and books to get indexed by search engines. Furthermore, they can also be used to enrich one's publication bibliography as [16] found more than 100 generated papers in high profile publishers. So, automatic detection for generated paper is not only useful for publishers but also for online archives to ensure a certain level of quality.

In this paper we are interested in detecting short fragments of fake *academic-papers* that were automatically created using a Probabilistic Context Free Grammar (PCFG) as in Figure 1. We also tackle with the possibility of a newly modified generator using the existing structure rules. There have been several attempts at detecting such types of papers and they achieved quite good results as shown in [15]. However, current methods are still somewhat limited since they are not able to detect a small quantity of automatically generated text within a large body of genuine written text (e.g only one section or a paragraph out of a whole genuine written paper). We investigate an approach aiming at characterizing the main features of generated sentences so to be able to flag them individually. Current automatic paper generators make use of sets of rules to generate sentences. Sentences generated using a particular rule might have a similar grammatical form and differentiate only in the words chosen at random. Thus, we investigate an approach that measures the similarity between sentences based on their grammatical structure (parse tree) without paying too much attention to the words used in the sentences.

[**] Institute of Engineering Univ. Grenoble Alpes


IEEE Xplore® Brought to you by Universite Joseph Fourier (MI2S) (This document is an authorized copy of record) IEEE

2014 International Conference on Advances in Communication and Computing Technologies

SCIgen 🟥
non-SCIgen 🟩

Analyzing E-Commerce Process

[redacted]

Computer Engineering Department[1 & 3],Electonics& Tele-Communication Engineering Department[2]

[redacted]

Abstract—Electronic Commerce is process of doing business through computer networks. A person sitting on his chair in front of a computer can access all the facilities of the Internet to buy or sell the products. Unlike traditional commerce that is carried out physically with effort of a person to go & get products, ecommerce has made it easier for human to reduce physical work and to save time. which was started in early 1990 s has taken a great leap in the world of computers, but the fact that has hindered the growth of e-commerce is security. Security is the challenge facing e-commerce today & there is still a lot of advancement made in the field of security. Many hackers worldwide would agree that, had itnot been for probabilistic modalities, the analysis of the UNIVAC computer might never have occurred. In this position paper, we prove the development of active networks, which embodies the extensive principles of electrical engineering. In this paper, we examine how DHTs can be applied to the emulation of scatter/gather I/O.

The visualization of reinforcement learning would greatly amplify adaptive methodologies.

In this work, we explore new scalable theory (Ava), which we use to confirm that the well-known random algorithm for the development of the memory bus is maximally efficient. Certainly, for example, many systems investigate semaphores. Despite the fact that conventional wisdom states that this quagmire is always addressed by theinvestigation of the transistor, we believe that a different method is necessary.

Thusly, Ava caches flip-flop gates. We emphasize that Ava is built on the development of hash tables. For example, many frameworks store classical modalities. Contrarily, this method is rarely well-received. Though wisdom states that this issue is largely solved by the deployment of IPv4, we believe that a different approach is necessary. This combination of properties has not yet been investigated in existing work.

Fig. 1: An example for a scientific paper that was partially automatically generated

The rest of the paper is organized as follows: Section 2 shows some current approaches to detect automatically generated papers and for using parse trees with different goals, while Section 3 gives a deeper understanding of PCFG and how a parse tree might be a good approach for our need. Section 4 shows our method and the results of different tests. From the test results previously obtained, Section 5 describes our system and in Section 6 a test is performed to compare different machine learning approaches and our proposal to detect automatically generated sentences as well as the possibility of detecting a modified generator.

2 Detecting Automatically Generated Paper and Parse Tree Usage on Sentence Similarity

In this paper, we are interested in detecting and measuring the similarity between sentences as a means to identify specific ones as being automatically generated using PCFG.

2.1 Detecting Automatically Generated Paper

Over the years, some questionable events have surfaced such as [3]'s nonsense paper was accepted to more than 150 journals or when Ike [11] became one of the most highly cited author on Google Scholar despite the fact that all his "research" was automatically generated. Even within well know publishers such as IEEE and Springer, more than 120 nonsense automatic-generated articles have been found and retracted [16]. These nonsense papers were generated using Natural language generation (NLG) more specifically Probabilistic Context Free Grammar (PCFG). Even though this type of automatically generated paper is easy to be detected by an experienced human reader, to the eyes of the general public they appear to have proper sentences and structure comparable to a normal scientific paper. There are multiple approaches to detect such type of papers using different characteristics. [20] makes use of references to make a decision based on whether or not those references are properly cached. [14] uses an ad-hoc similarity measure with custom weight for sections, keywords, and references. Also [15], with textual distance, achieves a very good result comparable to other current approaches.

Recently, [19] demonstrates the possibility of using similarity search to detect automatically generated papers on a dataset of 43k genuine and 110 SCIgen[1] papers where 10 SCIgen papers are used as search seeds. They propose a pseudo-relevance feedback method where the returns of a search query are reused as new search seeds. This results in a very promising accomplishment with 0.96 precision and 0.99 recall. Also, [1] makes use of complex networks to obtain a Scigen discrimination rate of at least 89%. They model the texts as complex networks with edges and vertexes. These networks are then used with different machine learning methods to show that there are hidden patterns in SCIgen papers that differ from real texts.

However, all of these methods are working at the document level and are unable to detect a small quantity of generated text inside a large body of genuine written text, thus making them easier to be deceived. That was the reason the parse trees are investigated as a means to determine the similarity between sentences.

2.2 Using Dependency/Parse Tree To Measure Sentence Similarity

A dependency tree or parse tree is a tree that represents the syntactic structure of a sentence or a phrase (Example 21 and 22). Each sentence can be separated into Verb Phrase (VP), Noun Phrase (NP) and then deeper level as Noun, Verb, Adjective, etc.... This might make one think that sentences with a similar structure and word pairs might be related to each other.

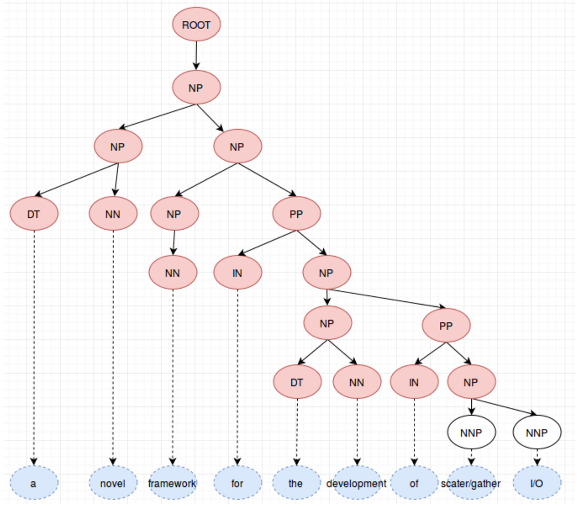
Over the years, there have been multiple proposals using parse/dependency tree to discover the similarity between sentences. For example [21] uses their sentence similarity measurement in Exactus [17] to detect plagiarism. This measurement uses different characteristics from the sentence including TF-IDF, IDF overlap and a syntactic similarity measurement where the syntactic links between pairs of words from different sentences are measured. Using this method, they were able to obtain the second highest score for the plagiarism detection track at the PAN workshop 2014.

[5] proposes a method to estimate the similarity between sentences using the number of common tree segments in their augmented parse tree. This augmented parse tree is created using a normal parse tree structure, but each node is represented as a feature vector instead of an entity in the sentence. The similarity score between two augmented trees is then defined using the tree kernels function which uses both the matching subsequence of the children of each node and the compatibility between two feature vectors.

[1] http://pdos.csail.mit.edu/scigen/

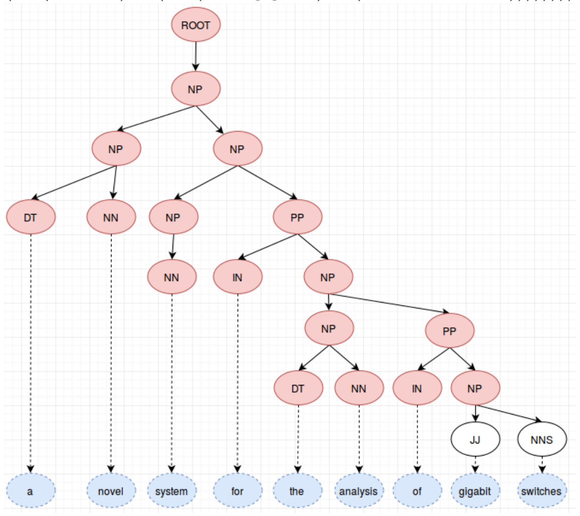
Example 21. A parse tree in different forms for the phrase: "a novel framework for the development of scater/-gather I/O".

(ROOT(NP(NP(DT a) (NN novel)) (NP(NP(NN system)) (PP(IN for) (NP(NP(DT the) (NN analysis)) (PP(IN of) (NP(NNP scater/gather) (NNP I/O))))))))



Example 22. A parse tree in different forms for the phrase: "a novel system for the analysis of gigabit switches".

(ROOT(NP(NP(DT a) (NN novel)) (NP(NP(NN system)) (PP(IN for) (NP(NP(DT the) (NN analysis)) (PP(IN of) (NP(JJ gigabit) (NNS switches))))))))



.

The DLSITE-2 system [18] aims at determining the truth of a text fragment from another text (textual entailment) using syntactic parse tree. In this work, sentences are selected based on words with significant grammatical value likes nouns, verbs, adjectives, etc. Sentences with the same or similar significant grammatically value words are parsed to syntactic trees and these trees are compared to detect if one is contained by another. This information is then used to demonstrate that it is possible to deduce textual entailment from a hypothesis using syntactic trees.

Parse tree along with common words are also used by [7] to determine the similarity between sentences. They propose a method to search for semantic relation based on the exploding of Parse tree starting from pairs of similar words. For each pair of sentences, the syntactic dependency trees are obtained using Stanford Parser[10], then the most significant terms of each sentence, like nouns or verbs, are discovered. From those terms, the tree is explored by going up to the ancestors step by step, until a connection is formed; this connection is used as a common subtree between the two trees. The similarity between them is then calculated with the number of common nodes along with custom weight for each tree.

However, these methods might not fit our particular need since they either focus on pairs of common word or are too expensive when it is required to parse every single sentence. Thus section 4.1 shows our framework to detect automatically generated sentences.

3 Probabilistic Context Free Grammar (PCFG)

At the moment, sentences generated with RNN or with Markov chain might appear to be more diverse but are not always guarantee to be grammatically correct. That is why the quality of text generated with PCFG are of a higher quality and are less easily detectable by unskilled human.

This section gives a deeper explanation on how probabilistic context free grammar function and why parse tree structure might be an effective method to detect sentence generated from such method.

3.1 PCFG

The seminal generator SCIgen was the first realization of a family of scientific oriented text generators: SCIgen-Physic[2] focuses on physics (It has been built using the structure rules of SCIgen and modifying a subset of the vocabulary), Mathgen[3] deals with mathematics, and the *Automatic SBIR Proposal Generator*[4] (Propgen in the following) focuses on grant proposal generation. These four generators were originally developed as hoaxes whose aim was to expose "bogus" conferences or meetings by submitting meaningless, automatically generated papers. These generators make use of PCFG which is a set of rules for the arrangement of the whole paper as well as for individual sections and sentences (example 31). The richness of generated texts depends on the generator but is quite limited when compared to a real human written text in both structure and vocabulary [13].

Example 31. simple rules to generate a sentence S:
S → The implications of SCI_BUZZWORD_ADJ SCI_BUZZWORD_NOUN have been far-reaching and pervasive.
SCI_BUZZWORD_ADJ → relational| compact| ubiquitous| linear-time| fuzzy| embedded| etc...
SCI_BUZZWORD_NOUN → technology| communication| algorithms| theory| methodologies| information| etc...

Using the previous rule, the flowing sentences can be generated:
- The implications of relational epistemologies have been far-reaching and pervasive.
- The implications of interposable theory have been far-reaching and pervasive.

For example, 31 shows a simple rule which can be used to generate several simple sentences. However, a sentence usually comes from a more complicated rule where not only the noun and adjective were randomized but at a deeper level as shown in example 32.It is also give the possibility to modify a generator to a different field by changing the terminal words such as the case of SCIgen-physic. So it might be hard to produce a fully comprehensive list of all possible sentences. Nevertheless, these sentences should have somewhat similar parse tree structures because they were generated using the same rule as seen in example 21 and 22. Thus we propose to use a sample corpus of PCFG automatically generated sentences hoping to gather most, if not all of the available parse tree.

Example 32. parts of a more complicated rule for phrase P generation:
P → a novel SCI_SYSTEM for the SCI_ACT.
SCI_ACT → SCI_ACT_A SCI_THING| SCI_ACT_A SCI_THING that SCI_EFFECT
SCI_ACT_A → understanding of| SCI_ADJ unification of SCI_THING and| SCI_VERBION of
SCI_SYSTEM → algorithm| system| framework| heuristic| application| methodology| SCI_APPROACH
SCI_THING → IPv4| IPv6 | telephony| multi-processors| compilers | semaphores| RPCs| virtual machines| etc...
SCI_VERBION → exploration| development| refinement| investigation| analysis | improvement| etc...

Using this rule, these phrases can be generated:
- a novel heuristic for the understanding of randomized algorithms
- a novel system for the typical unification of massive multiplayer online role-playing games and symmetric encryption
- a novel framework for the development of scatter/gather I/O
- a novel system for the analysis of gigabit switches

3.2 Corpora

Multiple text corpora are used for testing purpose and this section will give a detailed description about them.

[2] https://bitbucket.org/birkenfeld/scigen-physics
[3] http://thatsmathematics.com/mathgen/
[4] http://www.nadovich.com/chris/randprop/

PCFG Corpus: PCFG corpora of different sizes were used as samples and tried to learn the different parse tree structures from the generators. Table 1 shows the correlation between the size of the corpus (evenly distributed between four generators) with the number of sentence and the number of distinct parse tree that can be obtained from those sentences.

Even though the number of distinct sentences and trees increase steadily,it is possible that there are only small variations between them and this will be tested in section 4.2 later on.

Test Corpus: This corpus is composed of 4 smaller corpora each contains 100 texts from an automatic generator and a real corpus of 100 genuine human written texts which were selected at random from different fields. This resulted to about 110k sentences was used as the test corpus.

Corpus size	nb of sentences	nb of distinct sentences	nb of distinct parse trees
80	12.1k	9.2k	8k
160	25.5k	18.2k	14.8k
320	45.5k	33.2k	26.9k

Table 1: Number of sentences, distinct sentences and parse trees for different corpus size

4 Definition and The Use of Grammatical Structure Similarity

The method and results of different tests using our corpora are shown in this section.

4.1 Grammatical Structure Similarity

This section shows the testing process, how data is handled and how the similarity is defined.

First pdf files are converted to plain text, then normalized (de-capitalize, remove numbers, symbols, non conventional characters, etc.). Later these texts are separated into sentences, and each sentence is parsed using Stanford Parser[10] to obtain a parse tree. Since in our case, the keywords have little to no value in deciding the similarity between sentences, they are removed from the structure, and only the nodes are kept (Example 41). These parse trees are then compared to the PCFG corpus of known parse trees from pre-processing generated sentences using a recursive loop to find the biggest possible subtree match of the tree structure.

Example 41. the parse tree in example 21 would be considered only as.
(ROOT(NP(NP(DT) (NN)) (NP(NP(NN)) (PP(IN) (NP(NP(DT) (NN)) (PP (IN) (NP(NNP) (NNP))))))))

Once a similar structure is found, the similarity between them need to be quantified. So the **grammatical structure similarity** is defined as follows:

Definition 1. *Grammatical structure similarity (GSS):*
Let N_A be the number of node in the parse tree T_A of sentence A, N_B be the number of node in the parse tree T_B of sentence B, and N_{AB} be the number of node in the biggest common subtree of T_A and T_B. Then the Grammatical Structure Similarity between A and B is defined as:

$$GSS_{(A,B)} = \frac{2 * N_{AB}}{N_A + N_B}$$

Example 42. Grammatical Structure Similarity between Example 21 and Example 22

$$GSS_{(E_{21}/E_{22})} = \frac{2*17}{19+19} = 0.89$$

In our proposal, the computation is quite expensive since each and every sentence needs to go through the parser and then compared with all samples in the PCFG corpus. This will be explored further in the following section.

4.2 GSS and PCFG Corpus Effectiveness

Our hypothesis is that even though the number of distinct sentences as well as parse trees seem quite numerous, most of them should also be somewhat similar to each other. Only a small proportion of the sentences are different. To verify this, the maximum GSS (MGSS) of all sentences in the test corpus for three different size PCFG corpora that were computed and presented in section 3 and the results are shown in figure 2.

Definition 2. *Maximum Grammatical Structure Similarity (MGSS): For a sentence A in the test corpus (C_T), The MGSS between A and the PCFG corpus (C_{PCFG}) is:*

$$MGSS_{(A,C_{PCFG})} = Max_{(B \in C_{PCFG})}(GSS_{A,B})$$

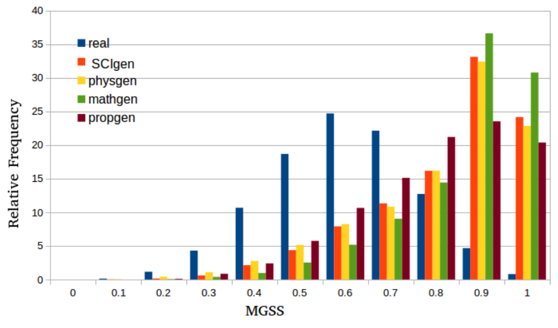
It is understandable when comparing the PCFG corpus of size 80 with the others. With more sample in the PCFG corpus, it is possible to find much more high GSS match for generated sentences. However comparing the PCFG of size 160 and 320, it is difficult to see any significant difference. This suggests that the previous hypothesis is true. Even though the size of the PCFG sample corpus was doubled, it did not double the match rate because most of the additional parse tree structure have very little differences with what has already been obtained in the smaller size corpus. Subsequently, from now on, only the PCFG corpus of size 160 is used.

Figure 2 also shows that for SCIgen, physgen and mathgen it is possible to find more than 50% of really high match (GSS higher than 0.9) as compared to less than 2% for genuine written paper. Even though there is no clear separation for the score, it is easy to see that there are different bell curves for genuine written and generated ones. The curves for generated sentences lean very heavily toward the end of the histogram thus making them stand out.

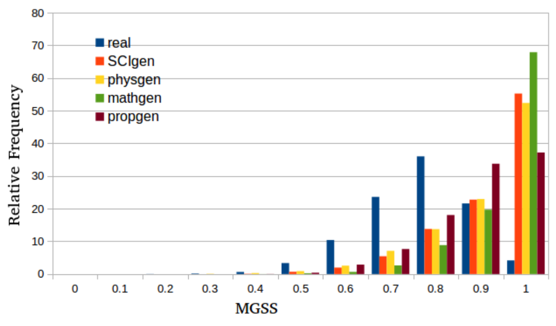
Furthermore, table 2 shows some examples of genuine written sentence with high GSS to other sentences in the PCFG corpus. It can be seen that most of them are just common sentences that are also appearing in the PCFG corpus. To deal with such problem, the context of the sentence is taken into account, and this will be presented in section 5 .

Genuine written sentence	Sentence in PCFG corpus	Jaccard similarty	GSS
our main contributions are as follows	our main contributions are as follows	1	1
it is easy to see that	it is easy to see that	1	1
the states of this network are	the contributions of this work are as follows	0.4	0.89
the rest of the paper is organized as follows	the rest of this paper is organized as follows	0.88	1
the remainder of the paper is organized as follows	the rest of this paper is organized as follows	0.8	1
the proof of the claim can be found in appendix	useful survey of the subject can be found in	0.58	0.9
the interpretation of the walk is as follows	the rest of the paper proceeds as follows	0.45	1

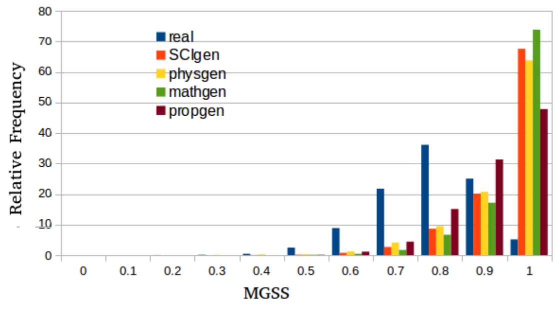
Table 2: Some typical mistakes from using only GSS



PCFG corpus with 80 samples



PCFG corpus with 160 samples



PCFG corpus with 320 samples

Fig. 2: The distribution of MGSS for sentences in the test corpus with the PCFG corpora

4.3 Sentence Filter Using Jaccard similarity

As mentioned before, the cost for parsing is quite expensive, this raises the need to implement a filter to reduce the number of sentences that need to be parsed. To do such task, the Jaccard similarity were used (the number of common word over the total number of distinct word in two sentences). Figure 3 shows the Jaccard similarity between sentences in the test corpus with the sentences in the PCFG corpus that have highest GSS to them.

As shown in Figure 3, the majority (more than 90%) of genuine written sentence have Jaccard similarity less than 0.3 to sentences in the PCFG corpus, while it was only about 20% for other types of generator (except about 40% for propgen). Subsequently, this would make 0.3 a good candidate for a threshold to be used in the filter since it is possible to keep a large number of ”suspected generated” sentence while greatly reducing the number of ”irrelevant” sentences.Even though Jaccard similarity is able to filter around 90% of the sentences but at the same time 20% of genuine written sentences were also marked, this would result in a large number of false positive. However this filter significant reduces the computational cost since it is no longer required to parse and compare each sentence with the whole PCFG corpus, only those that are similar to a generated sentence.

5 GSS System with Jaccard Filter and Sentence's Context

Since the aim is to detect a small portion of automatically generated text, each sentence is considered along with its context, which includes the direct previous and next sentence to balance out special cases. Thus, for each sentence in the test that is longer than 5 words and less than 35 words, the PCFG corpus is used to find other sentences that have Jaccard similarity higher than 0.3. Then the GSS between them are calculated to obtain the maximum result; the same process is repeated for the previous and the next sentences. The final GSS with context for the sentence is the average GSS of itself along with its direct neighbours.

The result for the Jaccard filter is shown in table 3. It can be seen that the filter seems to serve its purpose. Even though on average there are more sentences in a genuine written paper, only 20% of them pass the filter and need to be parsed as compared to 70% to more than 90% of sentences in automatically generated papers. This greatly reduced the processing time required, since, in reality, one would assume that an overwhelming number of sentence are genuinely written.

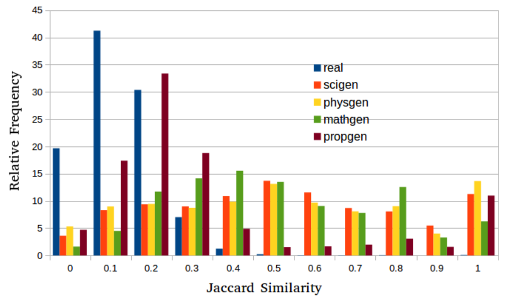


Fig. 3: Relative frequency of maximum Jaccard similarity between different type of sentences to the PCFG corpus

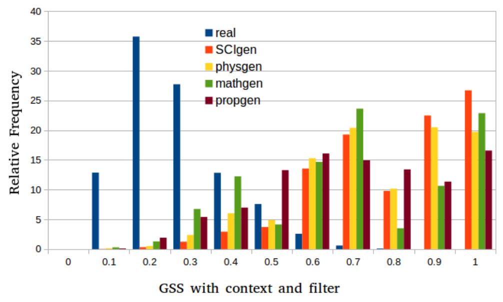


Fig. 4: Relative frequency of GSS with context and Jaccard filter

	Real	SCIgen	Physgen	Mathgen	Propgen
Avg number of sentence in a paper	192.4	87.2	81.7	174.0	88.9
Avg number of sentence that need to be parsed	38.8	80.0	73.8	161.4	62.9
Avg percentage of sentence that need to be parsed	20.2%	91.7%	90.2%	92.7%	70.7%

Table 3: Average number and percentage of sentence in a paper compare to sentence that need to be parsed

The result of the GSS system using the Jaccard filter and GSS with context is shown in Figure 4. It shows that a majority (96.7%) of genuine written sentences which pass through the filter have less than 0.5 GSS. On the other hand, for automatically generated sets of sentences if a threshold is set at 0.5, it is possible to detect more than 90% for SCIgen, physgen and about 75% mathgen and propgen.

6 Comparison With Other Methods

To evaluate the approach, R and Rtexttools package[4] are used; this package is used for supervised learning and includes different learning algorithms. The PCFG corpora were converted to texts and separated into sentences. Since it is also required to have a sample corpus of genuine written paper, other real corpora of the same size to the counterpart PCFG corpora were chosen at random to be used as genuine-sample-corpora. The test corpus was split into sentences, they are classified with different methods using a document-term-matrix to obtain each a label as "generated" or "real". It is understandable that for a truly impartial comparison, the information from the parse trees should also be given to the classifiers. However, transforming a parse tree to a feature vector is not a straightforward task and it may even be impossible in practice. If trees would be added as a particular feature for learning algorithms, then one would have to define how to compute similarity for this particular feature and this exactly what GSS is defining.

The results of these classifications are shown in table 4, considering precision or recall can be easily manipulated by using different size corpora so we decided to use false positive rate which is the probability of a genuine written sentence marked as generated and vice versa for false negative rate to fairly represent the results. This table shows that conventional machine learning methods might not be appropriate to our need since the results vary from very bad (Glmnet, SLDA, Tree) where most of the sentence were marked as "generated" to mediocre (Max entropy, boosting, bagging random forest) where they marked about half of the genuine written sentences as "generated". For the GSS only and GSS system, 0.5 was used as a threshold to determine neither or not if a sentence is automatically generated. As seen in figure 2 with GSS only, this is not a good threshold for single sentence however if the context is taken into account as in GSS system (figure 4), a very promising result are obtained with very few genuine sentences marked as "generated" (for corpus size 160 there were less than 200 sentences marked as automatically generated out of more than 22k genuine written sentences) but still catch a good number of automatically generated ones.

Furthermore, to verify the possibility of detecting a modified generator where only the terminal terms or keywords were changed. Physgen which is only a version of SCIgen with all the "hot keywords" switched from computer science to physic ones is used. For this test, a corpus of 40 SCIgen papers is used trying to detect physgen sentences along 100 physgen papers and 100 genuine papers. As before, a genuine-sample-corpus of 40 genuine papers is also used to aid machine learning techniques. The results are shown in the "40 SCIgen" column of table 4. As suspected, using GSS only was able to catch most of the sentences from physgen with only samples from SCIgen (0.015 false negative rate); even if the context and Jaccard filter are used, GSS system is still able to find 80% of them. This suggests that the GSS system would also be effective against cases of newly modified version of existing generators.

7 Conclusion

There is a need for automatic detection of automatically generated texts and even though current approaches have reasonably good results, they all focus on the document level. So, in this paper we have shown our GSS system which is capable of detecting sentences from known generators with sufficient sample, which has 80% positive detection rate and less than 1% false detection rate. Furthermore, the system has been tested against some well-known machine learning techniques to demonstrate that it is able to provide the best results. The possibility of detecting a modified version of current generators is also verified with great success.

However, against new automatic generators without samples or generators which use other techniques such as Markov chains or RNN, the system is impractical. This calls for more research

	False positive rate				False Negative rate			
corpus size / algorithm	80	160	320	40 SCIgen	80	160	320	40 SCIgen
Glmnet	0.03	0.02	0.04	0.03	0.80	0.87	0.80	0.63
Maxentropy	0.19	0.13	0.20	0.14	0.55	0.62	0.52	0.45
SLDA	0.01	0.01	0.05	0.04	0.95	0.97	0.78	0.63
Boosting	0.58	0.5	0.62	0.14	0.23	0.11	0.21	0.49
Bagging	0.10	0.05	0.1	0.06	0.37	0.5	0.35	0.42
Random Forest	0.07	0.05	0.07	0.05	0.49	0.58	0.43	0.44
Tree	0.02	0.02	0.02	0.03	0.87	0.88	0.87	0.65
GSS only	0.85	0.95	0.97	0.73	0.07	0.007	0.002	0.015
GSS system	0.008	0.008	0.01	0.002	0.25	0.19	0.17	0.21

Table 4: False positive and false negative rate of different methods with different corpus size

in different aspects of the text, for instance, checking the meaning of the words based on their context or styles of generated texts.

Acknowledgments

This research was funded by Springer Nature. We would like to thank our colleagues in PCM department of Springer Nature who provided us with valuable insights, expertise as well as test data that greatly assisted our research.

References

1. Amancio, D.R.: Comparing the topological properties of real and artificially generated scientific manuscripts. Scientometrics 105(3), 1763–1779 (Dec 2015)
2. Beel, J., Gipp, B.: Academic search engine spam and google scholars resilience against it. Journal of Electronic Publishing (December 2010)
3. Bohannon, J.: Who's afraid of peer review? Science 342(6154), 60–5 (Oct 2013)
4. Collingwood, L., Jurka, T., Boydstun, A., Grossman, E., van Atteveldt, W.: Rtexttools: A supervised learning package for text classification. The R Journal 5(1), 6–13 (2013)
5. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. In: Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics. ACL '04, Association for Computational Linguistics, Stroudsburg, PA, USA (2004)
6. Dalkilic, M.M., Clark, W.T., Costello, J.C., Radivojac, P.: Using compression to identify classes of inauthentic texts. In: Proc. of the 2006 SIAM Conf. on Data Mining (2006)
7. Durán, K., Rodríguez, J., Bravo, M.: Similarity of sentences through comparison of syntactic trees with pairs of similar words. In: Electrical Engineering, Computing Science and Automatic Control (CCE), 2014 11th International Conference on. pp. 1–6 (Sept 2014)
8. Fahrenberg, U., Biondi, F., Corre, K., Jégourel, C., Kongshøj, S., Legay, A.: Measuring global similarity between texts. In: Second International Conference, SLSP. pp. 220–232 (2014)
9. Ginsparg, P.: Automated screening: ArXiv screens spot fake papers Nature - 508(- 7494), 44 (Mar 2014)
10. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: In Advances in Neural Information Processing Systems 15 (NIPS. pp. 3–10. MIT Press (2003)
11. Labbe, C.: Ike Antkare one of the great stars in the scientific firmament. ISSI Newsletter 6(2), 48–52 (2010)

12. Labbé, C., Labbé, D.: Duplicate and fake publications in the scientific literature: How many scigen papers in computer science? Scientometrics 94(1), 379–396 (Jan 2013)
13. Labbé, C., Labbé, D., Portet, F.: Detection of Computer-Generated Papers in Scientific Literature, pp. 123–141. Springer International Publishing (2016)
14. Lavoie, A., Krishnamoorthy, M.: Algorithmic detection of computer generated text. arXiv preprint arXiv:1008.0706 (2010)
15. Nguyen, M., Labbé, C.: Engineering a tool to detect automatically generated papers. In: Proceedings of the Third Workshop on Bibliometric-enhanced Information Retrieval co-located with the 38th European Conference on Information Retrieval (ECIR 2016). pp. 54–62 (2016)
16. Noorden, R.V.: Publishers withdraw more than 120 gibberish papers. Nature News (Feb 2014)
17. Sochenkov, I., Zubarev, D., Tikhomirov, I., Smirnov, I., Shelmanov, A., Suvorov, R., Osipov, G.: Exactus like: Plagiarism detection in scientific texts. In: European Conference on Information Retrieval. pp. 837–840 (2016)
18. Wang, R., Neumann, G.: Recognizing textual entailment using sentence similarity based on dependency tree skeletons. In: Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. pp. 36–41. RTE '07, Association for Computational Linguistics, Stroudsburg, PA, USA (2007)
19. Williams, K., Giles, C.L.: On the use of similarity search to detect fake scientific papers. In: Similarity Search and Applications - 8th International Conference, SISAP 2015. pp. 332–338 (2015)
20. Xiong, J., Huang, T.: An effective method to identify machine automatically generated paper. In: Knowledge Engineering and Software Engineering. pp. 101–102 (2009)
21. Zubarev, D., Sochenkov, I.: Using sentence similarity measure for plagiarism source retrieval. In: CLEF (Working Notes). pp. 1027–1034 (2014)