Innovative: International Multi-disciplinary Journal of Applied Technology
(ISSN 2995-486X) VOLUME 02 ISSUE 11, 2024

Unmasking Search Engine Spam: Machine Learning Techniques for Detecting Algorithmically Generated Content

Yuldasheva Khurshida Timurovna

Abstract:

We propose a new way to spot search engine spam created by computer programs. Our method looks at how often certain writing styles and genres appear in a text. We use machine learning to automatically identify spam based on these patterns. Experiments show that our method can successfully detect spam created by a common type of text generator called a Markov chain.

Keywords:
Search Engine Spam, Spam Detection, Machine Learning, Text Generation, Markov Chains, Spam Filtering Algorithms, Automated Text Generation, Natural Language Processing, Language Patterns, Duplicate Content Detection, Non-Dictionary Features

Introduction

The internet is full of information, and search engines are our go-to tool for finding it. But there's a big problem: search engine spam. Spammers try to trick search engines into ranking their pages higher than they deserve. This can make it hard to find the information you really want. In fact, about 22% of all web content is estimated to be spam [11]. Even today, spam is a popular way to get a website to the top of search results. Spam makes it harder to find what you're looking for and can mess up how search engines work. The easiest way to create spam is to use computers to generate lots of spam pages automatically. A popular way to automatically create lots of text is by using something called Markov chains. First, you train a computer model on a bunch of existing text. Then, the model can generate new text that might not make complete sense but sounds pretty natural when you look at small parts of it. If the original texts were all about a certain topic, the new text will also be about that topic.

We don't yet have a complete theory of how to create text that sounds and makes sense like human-written text. So, there's no perfect way to make a computer write something that we can't tell apart from what a person wrote. However, we know a lot about how human language works. For example, we know that good writing has a consistent style and follows certain rules. Markov chains can only copy these patterns on a small scale. Our goal is to find ways to spot computer-generated text by looking at how it breaks these rules.

2 Review of existing methods

2.1 Search engine spam detection methods

Paper [14] looked at how well simple statistics could be used to spot search engine spam. They mostly focused on links pointing to a page, but didn't pay much attention to the actual text on the page.

Paper [19] looked at using language features to spot search engine spam. They used a special dictionary to calculate over 200 different statistics about the text. However, if you use a dictionary to detect spam, spammers could just change their methods to avoid it.

Paper [9] suggests a new way to spot search engine spam by looking at whether the text is trying to sell something. They do this by using some special statistics, mostly based on what people search for online or what ads are shown. But if these statistics are based on a limited set of words, it's easy for computers to create spam that can trick this method.

Papers [18] and [21] try to find spam in a way that doesn't rely on knowing specific words or topics. The first one looks at blogs and uses things like comments to spot spam, but this only works for blogs. The second one focuses on the way web pages are built (the HTML code), ignoring the actual text on the page.

2.2 Using non-dictionary characteristics in text analysis

People who study language are always looking for ways to measure things about writing that authors don't really think about. One thing they look at is how easy it is to read something. Studies like [12] and [13] have looked at this a lot. In the US, it's common to use simple things like how long the sentences are and how long the words are to figure out how easy something is to understand. There's even a study [10] that shows you can guess what kind of writing something is, like a news article or a story, just by looking at a few simple numbers about the text.

You can also figure out who wrote something by looking at the overall patterns in the text. A lot of research on this, like in paper [6], looks at things like how often certain words are used, and how long the sentences and words are. By studying these patterns, you can start to see what makes one writer different from another.

2.3 Identifying duplicates

Finding copies of text is a lot like finding text that's been created by a computer program called a Markov chain. If you make the program long enough, it can copy big parts of other texts. A good overview of how to find copies of text is in paper [3]. Paper [15] shows how we can use a technique called shingling to spot spam that's been made by copying bits of real text.

3 Text generators based on Markov chains

3.1 Text generation methods

Making spam for search engines means creating lots of text really quickly. There are a few ways to do this [16]:

- Writing it by hand;
- Copying it from other websites;

- Using computers to create new text;
- Changing existing text with a computer.

Writing it by hand is too much work and too expensive, so it's not very common. Copying from other sites is really popular, but there are good ways to catch it now, like using something called “shingling” [3].

So, the best ways to create spam today use computers to make new text that looks real but doesn't actually mean anything. These programs can make words and sentences that seem okay, but they don't make sense. Spammers also try to trick search engines by making their fake text seem like it's about things people are searching for.

3.2 Markov chains

A popular way to make computer-generated text is to use something called a Markov chain. Basically, it's a way of making a computer choose the next word based on the word that came before it. So, the computer kind of guesses what word should come next based on what it's already said.

A Markov chain is like a set of rules for moving from one place to another. These places are called “states”. The rules tell you how likely it is to move from one state to another. If these rules never change, we call the Markov chain "homogeneous". When we want a computer to make up text, we often use these unchanging rules because they're simple and easy to use.

3.2 Generating Search Spam Using Markov Chains

When we use Markov chains to make computer-generated text, we treat each word and punctuation mark as a “state.” We look at a bunch of existing text to figure out how often one word comes after another. This information helps us create a set of rules for choosing the next word. When we follow these rules, we get a string of words that kind of looks like real text, even though it might not make perfect sense.

A big thing about these text generators is something called the “order” of the Markov chain. It's basically how many words the computer looks at before guessing the next one. The more words it looks at, the longer and more connected the sentences can be. But if you make it look at too many words, it starts copying big chunks of the original text word for word.

Texts made by Markov chains have some cool tricks that make them great for spamming search engines. First off, the new text uses the same words as the original text you fed into the computer. So, if you want to make text about a specific topic, you can just feed the computer some text that's already really good at getting found by search engines, like the little snippets you see when you search for something online. The new text will be really similar and will also do a good job of getting found. Secondly, the new text is almost always completely different from anything else on the internet, which makes it hard to catch.

Here's an example of what I mean:

“Creating fake text from real documents. We wanted to see if we could tell if a piece of text was made by a computer program that copies from real documents. We looked at lots of words and marked them with a special code to show where they came from. Then we used a special kind of computer program called a support vector machine to decide if the text was fake or real.”

Computer programs that use Markov chains are often used to make fake websites called doorway pages. These fake websites are designed to trick people into clicking on them and then sending them to a real website that the person running the fake website wants them to see. The fake websites don't have any useful information, but they're good at tricking search engines into thinking they're important. Finding and getting rid of these fake websites can help make search results better.

4 The proposed method

The idea behind this approach is based on methods we've used before to figure out what kind of text something is and who wrote it.

We've found some special features in text that are hard for writers to change.

We used these features and a computer learning system to build a tool that can spot fake text.

This tool is based on the idea that text written by a computer using a simple rule, like a Markov chain, will look different from text written by a human.

4.1 Characteristics under consideration

The considered characteristics can be conditionally divided into the following overlapping groups:

- Readability features;
- Stylistic features of the text;
- Genre features of the text [10];
- Global statistical characteristics;
- Morphological features of words in the text (the mystem parser [4] was used for morphological analysis);
- Punctuation usage statistics.

Features were selected based on the assumption that it is difficult for a human author to control their values. Each author and genre has a unique style that is reflected in the values of certain characteristics. It is assumed that text generators based on Markov chains are unable to emulate some of these characteristics. Thus, search engine spam can be singled out as a separate document genre, and the task of identifying this genre can be considered.

An important property of the selected features is that they are not based on the topic or lexicon of the texts, thus allowing the identification of artificially generated texts regardless of their subject matter. Below is a complete list of the extracted features, with some items corresponding to several characteristics:

1. Average number of words per sentence;
2. Average number of characters per word;
3. Average number of syllables per word;
4. Proportion of words longer than 7 characters;
5. Proportion of words with more than 7 syllables;
6. Proportion of one-syllable words;
7. Proportion of two-syllable words;
8. Minimum number of syllables in a sentence;
9. Maximum number of syllables in a sentence;
10. Number of particles “would”;
11. Number of particles “well”, “that's it”, “after all”;
12. Average number of punctuation marks per sentence;
13. Average number of expressive punctuation marks (“!”, “?”, “…”);
14. Average number of words starting with a capital letter;
15. Proportion of different parts of speech:

    a. Proportion of verbs among words;

    b. Proportion of adjectives among words;

c. Proportion of nouns among words;
d. Proportion of numerals among words;
e. Proportion of ordinal numbers among words;
f. Proportion of adverbs among words;
g. Proportion of particles among words;
h. Proportion of prepositions among words;
i. Proportion of particles among words;
j. Proportion of interjections among words:

16. Variance of the quantity of different parts of speech (from point 15) across sentences;
17. Proportion of first-person pronouns;
18. Proportion of second-person pronouns;
19. Proportion of verbs by tense:
    a. Proportion of present tense verbs
    b. Proportion of past tense verbs
    c. Proportion of non-past tense verbs
20. Proportion of nouns by gender:
    a. Proportion of masculine nouns among all words and among nouns
    b. Proportion of feminine nouns among all words and among nouns
    c. Proportion of neuter nouns among all words and among nouns
21. Text compression using various algorithms:
    a. bz2 compression
    b. zlib2 compression
22. Word frequency distribution (evaluation by frequency histogram): Measures how often different words occur in the text.

In total, 61 text characteristics are being investigated. Each characteristic is represented by a positive real number. Thus, each document is associated with a feature vector of 61 elements.

4.2 Machine learning methods

A proposal is made to develop a classifier, leveraging machine learning algorithms, to distinguish between human-written and AI-generated texts. This classifier would be trained on a dataset comprising authentic, human-crafted content and texts suspected to be AI-generated. By identifying correlations between specific characteristics and the method of text creation, the classifier could potentially be employed to detect search engine spam.

We looked at two popular machine learning algorithms in this study:

- A support vector machine classifier with a linear kernel, using the SVMLight implementation [17];
- A decision tree classifier, Dtree.

4.2.1 SVMLight Machine Learning Algorithm

Support vector machines (SVMs) are classification algorithms that create hyperplanes to optimally separate data points belonging to different classes in a feature space. By maximizing the margin between these classes, SVMs enhance classification performance.

In this work, we utilized SVMLight [17], a popular implementation of the SVM algorithm.

4.2.2 DTree Machine Learning Algorithm

The foundation of our approach is the C4.5 decision tree algorithm [20]. A decision tree in this context is a binary tree structure where each internal node represents a test on an attribute, such as a specific word or phrase. The outcome of the test determines the branch to follow. Leaf nodes, at the end of each branch, represent the class labels (spam or not spam) and assign a probability to each class.

The tree-building process begins at the root node, which initially contains a subset of the training data. At each leaf node, the algorithm searches for the best split based on a feature and a threshold value. The best split is the one that results in the lowest information entropy in the resulting child nodes. If the entropy is reduced, the leaf is expanded into two child nodes, and the feature and threshold are associated with the parent node. The data is then partitioned according to the split criterion.

Once the decision tree is built, the probability of a document being classified as spam or not spam is determined for each leaf node.

	DTree			SVMLight		
	Rosadult	Completeness	F-measure	Rosadult	Completeness	F-measure
Markov2	91,30%	93,27%	92,27%	72,9%	74,83%	73,85%
Doorway_Su	92,11%	89,98%	91,03%	69,41%	68,24%	68,82%
Rusadult	99,19%	99,26%	99,23%	89,65%	89,83%	89,74%

Table 1. Accuracy and completeness of detection of texts generated by different text generators

This is achieved by assigning each document in the training set to the appropriate leaf based on the tree's decision rules. The proportion of spam and non-spam documents within each leaf is then calculated and stored. To prevent overfitting, a holdout method is employed where the tree is constructed using one portion of the training data, and the probabilities are calculated using a separate, unseen portion.

To improve the model's generalization ability, we employ an ensemble method where multiple decision trees are constructed. For each tree, the training data is randomly partitioned into two subsets. The first subset is used to train the tree, and the second subset is used as a validation set to estimate the class probabilities at the leaf nodes.

The ensemble of decision trees is combined into a single classifier using a majority voting scheme. When classifying a new document, it is passed through each tree in the ensemble. For each tree, the document is assigned to a leaf node, and the associated class probabilities are obtained. The final classification is determined by summing the probabilities for each class across all trees and assigning the document to the class with the highest total probability.

5 Experiments

The ROMIP By.Web web page collection [1] served as the dataset for the experiment. All HTML tags, including those containing script elements, were stripped from the documents before analysis. The first experiment focused on assessing the system's capability to identify text generated by Markov chains and commonly used search engine spam generation techniques. The second experiment explored the system's ability to detect real-world search engine spam.

5.1 Text generators under consideration

The initial experiment involved three text generation models:

- a custom Markov chain model with a chain length of 2 (dubbed markov2),
- the Doorway.su doorway generator [2] (labeled doorway_su),
- and the Rusadult doorway generator [5] (labeled rusadult).

Each document was transformed into a feature vector and assigned to one of two classes: spam or good. Documents generated by the text generators were labeled as spam, while those from the ROMIP By.Web collection were labeled as good. The goal was to evaluate the system's capacity to distinguish between texts generated by these models and genuine content.

5.2 Construction of training sets

The training data for the classifiers was created in the following manner:

1. A random sample of 20,000 documents was drawn from the By.Web dataset and labeled as “good” (GoodSet);
2. A separate random sample of 20,000 documents (ExampleSet) was also extracted from the By.Web dataset;
3. The ExampleSet was used as a basis to generate synthetic text using three different algorithms:
   a. 20,000 texts were generated using the markov2 algorithm and labeled as “spam” (SpamMarkov2);
   b. 20,000 texts were generated using the Doorway.Su doorway generator and labeled as “spam” (SpamDoorwaySu);
   c. 20,000 texts were generated using the Rusadult doorway generator and labeled as “spam” (SpamRusadult).

The GoodSet, SpamMarkov2, SpamDoorwaySu, and SpamRusadult datasets were each split into two equal halves. The first 10,000 documents from each set formed the training data, and the remaining 10,000 documents served as the test data. To assess the effectiveness of detecting each generator individually, three separate training sets were constructed. Each training set comprised 10,000 "good" documents and 10,000 documents generated by a specific generator. The corresponding classifier was then evaluated on a test set of 10,000 "good" documents and 10,000 documents generated by the same generator.

5.3 Application of Machine Learning Methods

The SVMLight classifier was trained using the default parameter settings. Following training, a threshold was determined for the classification decision that aimed to optimize the balance between precision and recall on the training data.

	DTree		SVMLight	
	Completeness of real spam	F-measure	Completeness of real spam	F-measure
Markov2	65,66%	91,57%	44,33%	71,69%
Markov2+RealSpam	87%	92,61%	49,33%	72,46%

Table 2. Completeness of detection of real spam

When using decision trees for classification, each tree was limited to a maximum depth of 50 nodes. An ensemble of 100 such trees was created by combining their predictions through a voting process. The experiment evaluated the performance of these classifiers in terms of precision, recall, and F1-score for spam detection. Results for three text generators and two classification algorithms are summarized in Table 1.

5.4 Experiment on detecting real spam

The second experiment aimed to evaluate the effectiveness of the classifiers in detecting real-world spam, specifically those generated using Markov chains. To achieve this, 600 documents were manually collected from Yandex Blog Search [8] using commercial search queries. These documents,

classified as "RealSpam," were assumed to be generated using Markov chain techniques and were designed to target low-frequency search queries.

If you're looking for a car, Deo has a wide range of vehicles for sale. Our price list offers competitive deals. We also have a great selection of car audio accessories at affordable prices. For those interested in the Daewoo Nexia, it's worth noting that this model is a modern interpretation of the classic Opel Kadett E. Production began in 1986 in [country]. And if you're in Tyumen, check out our car classifieds for new and used vehicles, including car DVRs.

The spammers seem to have combined various automotive-related content to create this text. Their intention was to draw visitors to this page and encourage them to click on the embedded links.

The experiment involved two distinct training sets:

- The first set comprised 10,000 documents from the ROMIP By.Web collection and 10,000 documents artificially generated using the markov2 algorithm.
- The second set was an extension of the first, augmented with an additional 300 real-world spam examples.

The test dataset included 10,000 documents from the ROMIP By.Web collection, 10,000 synthetic documents generated using the markov2 algorithm, and 300 real spam examples. Two machine learning algorithms, DTree and SVMLight, were used to evaluate the models. The experiment measured the recall (sensitivity) of real spam detection and the overall F1-score (harmonic mean of precision and recall) for the classification task. The results are summarized in Table 2.

6 Conclusions

The experiments indicate that analyzing features that are difficult for an author to manipulate can be a successful approach to detecting search spam produced by different text generation algorithms. A decision tree classifier proved to be the most effective when applied to a dataset with a wide variety of features.

The experimental findings allow us to make certain inferences regarding the text generation techniques employed by the spam generators examined. It appears that Doorway.Su likely utilizes Markov chains for generating its content, given the close similarity in detection accuracy between spam produced by Doorway.Su and our benchmark Markov chain model.

The findings of the second experiment suggest that training on synthetic search spam data can lead to accurate detection of real-world spam. This implies that the creation of a suitable training dataset for spam detection can be made more efficient by utilizing artificially generated samples.

The inclusion of a relatively small number of real spam examples (300 documents) led to a substantial improvement in the recall of detecting this specific type of spam, while maintaining the overall spam detection performance. Moreover, our findings suggest that it is possible to detect texts generated by Markov chains, even when the original training data for these models is not available.

7 Conclusion

This work proposes a novel method for detecting search spam that leverages text features traditionally employed in genre classification and authorship attribution tasks.

The method was tested on texts generated by a Markov chain-based text generator as well as two widely used Russian search spam generators. Experimental results showed that the proposed approach is effective in detecting search spam.

Future work will focus on exploring alternative text analysis techniques and investigating the feasibility of detecting other prevalent types of search spam.

Literature

[1] Веб коллекция BY.Web, http://romip.ru/ru/collections/by.web-2007.html.

[2] Генератор дорвеев Doorway.Su, http://doorway.su/.

[3] Зеленков Ю.Г., Сегалович И.В., Сравнительный анализ методов определения нечетких дубликатов для Web-документов // Труды 9ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL'2007, Переславль, Россия, 2007. – Том 1, С. 166-174.

[4] Парсер mystem http://company.yandex.ru/technology/mystem/.

[5] Серверный генератор дорвеев от RUSADULT.com, http://doorways.rusadult.com/ru/.

[6] Фоменко В.П., Фоменко Т.Г., Авторский инвариант русских литературных текстов, 1981.

[7] Чжун Кай-лай, Однородные цепи Маркова. Перев. с англ. — М.: Мир, 1964. — 425 с.

[8] Яндекс.Поиск по блогам, http://blogs.yandex.ru/.

[9] Benczúr, A. A., Bíró, I., Csalogány, K. and Sárlós, T. Web Spam Detection via Commercial Intent Analysis. In Proceedings of the 3rd international workshop on Adversarial Information Retrieval on the Web, Banff, Alberta, Canada, May 8th, 2007. Pages: 89–92.

[10] Braslavski P. Document Style Recognition Using Shallow Statistical Analysis. In Proceedings of the ESSLLI 2004 Workshop on Combining Shallow and Deep Processing for NLP, Nancy, France, 2004, p. 1–9.

[11] Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M., Vigna, S. A Reference Collection for Web Spam. ACM SIGIR Forum Volume 40, Issue 2 (December 2006) Pages: 11–24.

[12] Dale, E. and J. S. Chall. 1949. "The concept of readability." Elementary English 26: 23.

[13] Dubay, W.H.. 2004. The Principles of Readability. Costa Mesa, CA: Impact Information

[14] Fetterly, D., Manasse, M., Najork, M. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In Proceedings of WebDB'04, New York, USA, 2004.

[15] Fetterly, D., Manasse, M., Najork, M. Detecting phrase-level duplication on the World Wide Web. In Proceedings of SIGIR'05, pages 170–177, New York, NY, USA, 2005. ACM.

[16] Gyöngyi, Z. and Garcia-Molina H., Web Spam Taxonomy. In Proceedings of AIRWeb 2005, May 2005.

[17] Joachims, T. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schцlkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[18] Mishne, G., Carmel, D., and Lempel, R. Blocking blog spam with language model disagreement. In Proceedings of AIRWeb 2005, May 2005.

[19] Piskorski, J., Sydow, M., Weiss, D., Exploring Linguistic Features for Web Spam Detection: A Preliminary Study. In Proceedings of the 4th international workshop on Adversarial Information Retrieval on the Web, Beijing, China, Pages 25-28.

[20] Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[21] Urvoy T., Chauveau E., Filoche, P. Tracking Web Spam with HTML Style Similarities. ACM Transactions on the Web,